An Investigation of Feature Weighting Algorithms and Validation Techniques using Blind Analysis for Analogy-Based Estimation

A thesis submitted as partial fulfilment of the requirement of
Doctor of Philosophy

by

BOYCE B. SIGWENI

College of Engineering, Design and Physical Sciences
Brunel University London

January 2016

Abstract

Context: Software effort estimation is a very important component of the software development life cycle. It underpins activities such as planning, maintenance and bidding. Therefore, it has triggered much research over the past four decades, including many machine learning approaches. One popular approach, that has the benefit of accessible reasoning, is analogy-based estimation. Machine learning including analogy is known to significantly benefit from feature selection/weighting. Unfortunately feature weighting search is an NP hard problem, therefore computationally very demanding, if not intractable.

Objective: Therefore, one objective of this research is to develop an efficient and effective feature weighting algorithm for estimation by analogy. However, a major challenge for the effort estimation research community is that experimental results tend to be contradictory and also lack reliability. This has been paralleled by a recent awareness of how bias can impact research results. This is a contributory reason why software effort estimation is still an open problem. Consequently the second objective is to investigate research methods that might lead to more reliable results and focus on blinding methods to reduce researcher bias.

Method: In order to build on the most promising feature weighting algorithms I conduct a systematic literature review. From this I develop a novel and efficient feature weighting algorithm. This is experimentally evaluated, comparing three feature weighting approaches with a naïve benchmark using 2 industrial data sets. Using these experiments, I explore blind analysis as a technique to reduce bias

Results: The systematic literature review conducted identified 19 relevant primary studies. Results from the meta-analysis of selected studies using a one-sample sign test ($p = 0.0003$) shows a positive effect — to feature weighting in general compared with ordinary analogy-based estimation (ABE), that is, feature weighting is a worthwhile technique to improve ABE. Nevertheless the results remain imperfect so there is still much scope for improvement. My experience shows that blinding can be a relatively straightforward procedure. I also highlight various statistical analysis decisions which ought *not* be guided by the hunt

for statistical significance and show that results can be inverted merely through a seemingly inconsequential statistical nicety. After analysing results from 483 software projects from two separate industrial data sets, I conclude that the proposed technique improves accuracy over the standard feature subset selection (FSS) and traditional case-based reasoning (CBR) when using pseudo time-series validation. Interestingly, there is no strong evidence for superior performance of the new technique when traditional validation techniques (jackknifing) are used but is more efficient.

Conclusion: There are two main findings: (i) Feature weighting techniques are promising for software effort estimation but they need to be tailored for target case for their potential to be adequately exploited. Despite the research findings showing that assuming weights differ in different parts of the instance space ('local' regions) may improve effort estimation results — majority of studies in software effort estimation (SEE) do not take this into consideration. This represents an improvement on other methods that do not take this into consideration. (ii) Whilst there are minor challenges and some limits to the degree of blinding possible, blind analysis is a very practical and an easy-to-implement method that supports more objective analysis of experimental results. Therefore I argue that blind analysis should be the norm for analysing software engineering experiments.

Publications List

The publications listed below are results of work based on this thesis:

- Sigweni, B., (2014, May). Feature weighting for case-based reasoning software project effort estimation. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (EASE) (p. 54). ACM.

- Sigweni, B. and Shepperd, M., (2014, September). Feature weighting techniques for CBR in software effort estimation studies: a review and empirical evaluation. In *Proceedings of the 10th International Conference on Predictive Models in Software Engineering* (PROMISE) (pp. 32-41). ACM.

- Sigweni, B. and Shepperd, M., (2015, April). Using blind analysis for software engineering experiments. In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering* (EASE) (p. 32). ACM.

- Ayoung, A.D., Sigweni, B. and Abbott, P., (2015, December). Case-Based Reasoning System for Predicting the Sustainability of a Telecentre. In *Proceedings of the 10th International Conference for Internet Technology and Secured Transactions* (ICITST-2015). IEEE.

- Sigweni, B., Shepperd, M., & Turchi, T. (2016, June). Realistic assessment of software effort estimation models. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering* (EASE) (p. 41). ACM **(best short paper award)**

Contents

List of Figures

List of Tables

Acronyms

ABE Analogy-Based Estimation

CBR Case-Based Reasoning

COCOMO Constructive Cost Model

FSS Feature Sequential Selection

FSW Feature Sequential Weighting

FWT Feature Weighting Technique

ISBSG International Software Benchmarking Standards Group

MAR Mean Absolute Residual

MdMRE Median of Magnitude Relative Error

MMRE Mean Magnitude of Relative Error

MRE Magnitude of Relative Error

PROMISE Software Effort PRedictOr Models In Software Engineering

SA Standardised Accuracy

SEE Software Effort Estimation

SLR Systematic Literature Review

Acknowledgments

I would first and foremost like to thank my supervisor Professor Martin Shepperd for his dedicated supervision and unwavering support, not only for his academic guidance and assistance, but also for his patience and personal support — without which this thesis would never have got off the ground. Thanks Prof, I am truly grateful.

Thanks to Dr. Simon Kent for his work as my second supervisor. Many thanks to Tommaso Turchi for his practical assistance in the investigations carried out for this thesis.

A great deal of thanks must also go to my parents, siblings, and Bareneetse Memo who have supported me throughout, putting up with my thoroughly unsociable behaviour and late nights.

The research contained within this thesis has been supported by Botswana International University of Science & Technology (BIUST).

This thesis is dedicated to the memory of my father, Themba Ntatiwa Sigweni. Thank you for believing in me.

Chapter 1

Introduction

Given the importance of timely and accurate software cost prediction, it is unsurprising that there is a large body of published research work, the majority of which has focused on effort prediction since effort is normally the dominant and hardest to predict component of overall cost. Unfortunately, estimates are often substantially wrong and specifically most projects encounter effort overruns.

Software effort estimation with resounding reliability, productivity and development is a demanding and arduous task. "*Anyone who expects a quick and easy solution to the multifaceted problem of resource estimation is going to be disappointed.*" — Alfred Pietrasanta 1968 cited in [99]. Three decades later Briand et al. observed that "*Despite the large number of cost factors collected and the rigorous data collection, a lot of uncertainty in the estimates can be observe*d" [27]. The current situation is not any better; there are still problems associated with effort estimation, whose errors have devastating consequences. An example is the cancelled NASA's Check Out Launch Control System (CLCS) project in 2003, where approximately 400 developers lost their jobs when the project was cancelled after the initial estimate was increased by more than $250 million dollars [60]. Worldwide software revenue totalled $3.5 trillion in 2015 according to Gartner, Inc. (NYSE: IT) [47]. Considering that software effort estimates are often wrong [25], therefore this highlights the importance of very good estimates. Good estimates have potential not only to save livelihoods but to help other sectors of the economy grow.

Software effort estimation research is focused mainly on issues associated with errors in estimation and seeking to identify better estimation methods. These efforts have resulted in varying levels of support for different approaches [60].

Although many approaches have been proposed, a widely used technique is based upon Case-Based Reasoning (CBR) [98] and is usually referred to as Analogy-Based Estimation (ABE) [121, 150]. CBR essentially proceeds by using knowledge of past episodes of interest called cases that are encoded as vectors of features that describe the case state and the case solution. New, or target cases, are solved by utilising solutions from past cases that exhibit similarity, i.e., are proximal in the feature space.

For effort prediction a case is usually a project, is a fixed vector of features values. A feature, sometimes called an attribute, describes some characteristic of a case. Two examples of features types: categorical (development_environment $\in$ {Java, C++, python}) and continuous (for example cyclomatic complexity). Continuous features are used when the values can be linearly ordered, even if the values are not truly continuous.

The case state will be a vector of features such as size, development environment, client experience, etc., and the solution is the actual amount of effort utilised. Thus the case-base is conceptually an $n \times p$ matrix where there are n cases and p features. Often the features that are included in the case-base are more due to happenstance and availability rather than because it is known that there are well defined relationships with the case solution. Moreover, there may exist multicollinearities amongst these features. Consequently, as is common with the majority of machine learning techniques, it is widely acknowledged that not all features are of equal importance [51, 157]. Thus CBR systems will benefit from optimisation of the feature sets. In order to improve CBR estimation accuracy, many researchers have proposed Feature Weighting Techniques (FWTs) or feature set optimisation.

Feature set optimisation can be accomplished by means of feature weighting which has the effect of stretching or compressing the feature space thus impacting the proximity of cases (projects) and thus modifying the set of neighbour projects that are used to donate solutions. Such problems are NP-hard and for non trivial numbers of cases and feature sets present significant computational challenges. A slightly less daunting approach, although

still NP-hard, is feature subset selection where features are assigned weights of $\{0, 1\}$. Until recently this has been the dominant approach within software effort prediction.

In recent years there has been growing concern about conflicting experimental results in empirical software engineering. This lack of agreement is shown amongst the many empirical studies conducted in the various branches of empirical software engineering [145] including defect prediction [118, 175] and Software Effort Estimation (SEE)) [72, 149]. Closer investigation suggests that a contributory reason — though only one of many — is selective reporting and partial analysis [71, 146].

The focus of this thesis is on the analogy-based or case-based software effort estimation techniques. Other software effort estimation techniques are not the primary focus of this research, nevertheless, a brief introduction of other algorithmic models are discussed in Chapter 2 to put the research theme into context. This thesis also focuses on blind analysis as mechanism to reduce researcher bias (analysis bias) in an effort to improve consistency of experimental results in empirical software engineering.

As it is common practice within the software engineering literature, the terms effort and cost will be used interchangeably during the course of this thesis, as will the terms prediction and estimation, case and project.

1.1 Motivation for Thesis

Although there is widespread consensus that some form of feature weighting technique is beneficial there has been no empirical investigation on how offering different weight sets affects estimation accuracy. Nor has there been an experimental evaluation of CBR FWTs on the interaction between performance and data set based on time series or data set size. The motivation for this thesis is to provide the software effort estimating community with a more effective approach to the feature weighting problem, complementing current practices, and blind analysis for software engineering experiments. The main reasons for this are:

- *Much of research effort has been spent on developing different feature weighting tech-*

niques to improve estimations accuracy, however predictions are still far from perfect. The suggestion, therefore, is to apply research effort on how these diverse feature weighting techniques might address the problems of effort estimation for each new case in a more effective way.

- *There has been growing concern about conflicting experimental results in empirical software engineering [146].* Numerous empirical studies into the consistency of results of software experiments have been published in the literature (for example [72, 118, 145, 149, 175]). Unfortunately, the over-riding trend is inconsistency. Closer investigation suggests that a contributory reason — though only one of many — is selective reporting and partial analysis [71, 146] known as analysis bias. Exploring techniques that help reduce analysis bias makes it possible to build effort prediction systems that produce consistent results.

- *Validation approaches currently used may not be representative of industry practice.* Validation must reflect the industrial practice where a project's effort can only be predicted based on past projects. More appropriate approaches need to be adopted that do validation using only earlier projects. Normally validation is done using any project in the case base even if it was completed after the test project.

1.2 Research Objectives

The work described within this thesis is a practical investigation of offering different weight sets to 'local' regions (different parts of the instance space) of data set, which has received little to no attention from the software engineering community. The software community is also concerned about the contradictory results from software engineering experiments. Therefore I will look at techniques to reduce bias, this will in turn improve consistency.
This work is an attempt to partially redress this imbalance with the following objectives:

1. systematically evaluate *all* relevant empirical studies of feature weighting using analogy-based estimation in the domain of software engineering.

2. research bias in order to mitigate contradictory results in software engineering.

3. propose an efficient feature weighting algorithm for analogy-based estimation.

4. investigate if there is any difference between existing traditional validation techniques such as jackknifing and pseudo time-series — a technique that better reflects industry practice

The overall objective is to investigate in an unbiased way the efficacy of offering different weight sets for analogical reasoning for the purpose of estimating the required effort to complete software projects.

1.3 Scope of Work

Whilst this work could be applied to other software measurement problems such as the prediction of defect density or estimation of project duration, the focus of this thesis is exclusively on the estimation of software project effort. Admittedly effort is by no means the only driver of software project costs, it is commonly the most important. The definition of what effort constitutes varies between development environments but is expected to include at least, the effort expended during the requirements definition, design, coding and testing phases [141]. In reality it is the cost rather than effort that we are trying to predict. However, since software production costs are often too sensitive to be made public, we therefore substitute effort for cost. Cost is largely informed by effort, therefore effort can be used as a convenient proxy for cost with the assumption that there is a linear relationship between cost and effort. Also effort is also much easier to compare across different companies. We use data sets from different companies.

Effort estimated in this study is not restricted to the effort for developing new software but can also refer to effort for maintenance or enhancement of existing projects, which reflects the large amount of effort required for such activities on software projects. Data sets such as Finnish has a mixture of new, maintenance and enhancement projects. This thesis only uses industrial projects data sets, therefore hypothetical or educational projects such as those

commonly carried out by students for their coursework are not considered.

1.4 Thesis Structure

Following on from the introduction which outlined the importance of software effort estimation, Chapter 2 presents a background of analogy-based software effort estimation and related work. Analogy-based estimation is a form of case-based reasoning in the machine learning field therefore it is important to first discuss machine learning and in general feature weighting techniques available. This enables the discussion on various types of software estimation methods before detailing the analogy-based estimation which the is main technique of this thesis. To obtain a better understanding of analogy-based estimation, the following are discussed; tools, systems, advantages and disadvantages as well as issues with analogy-based estimation.

Machine learning is known to significantly benefit from feature selection/weighting. Chapter 3 reports on a comprehensive systematic literature review focussing on how feature weighting has been dealt with in the empirical software engineering community for effort estimation using case based reasoning. I specifically focus on feature weighting in software effort estimation (SEE) because of the distinctive characteristics of the SEE data sets. Feature weighting in other domains may not be as representative of interrelationship of features in software effort estimation. Systematic literature review (SLR) findings show that feature weighting does improve estimation accuracy. However, experimental results tend to be contradictory therefore lack reliability.

In an effort to address the issue of experimental results unreliability in software engineering, I looked at experimental techniques to reduce experimental bias in Chapter 4. Specifically I used the little known technique of blind analysis and I also incorporated robust statistics as appropriate and reported my experiences. Feature weighting does improve estimation accuracy. However, all such approaches assume a uniform set of weights across the entire case base. In Chapter 5, I investigate whether local feature weighting schemes may offer benefit over more conventional approaches which use a single weight set for the entire data set. A

new algorithm that finds more local sets of feature weights is presented. In addition, I address this question in the more realistic setting of a dynamic data set, i.e., one that grows over time.

Empirical evaluation of feature weighting for analogy-based effort estimation in chapter 6 is based on two industrial data sets. The results show that blind analysis can be used to reduce analysis bias and suggest that multiple weight sets should be used in effort prediction for *all* future models using analogy-based effort estimation that employ feature weighting to improve estimation accuracy.

The objective of this work was to improve SEE. In Chapter 7 as well as providing a list the contributions of this thesis I outline limitations of the work. Finally I provide pointers to possible future work in the field of feature weighting in empirical software engineering.

Chapter 2

Background and Related Work

2.1 Introduction to Machine Learning

Machine Learning is the field of scientific study that explores induction algorithms, that is, algorithms that can be said to "learn" from data (definition provided by Kovahi and Provost in the glossary of terms [94]). Learning algorithms are presented with data that represent a task so that they can learn from it, and therefore predict a solution for a new task when provided with new data. These algorithms make use of *inputs* to build models that could be used to make predictions or decisions [21]. Machine learning can be employed in computing processes where creating and explicitly programming instructions is not practical or easily achieved.

Chapter 1 provided an introduction to the topic of analogy-based estimation with a brief discussion about the importance of software effort estimation and the issue of feature weighting. This chapter continues the software effort estimation discussion from Chapter 1 by first addressing the machine learning process. The term CBR effort estimation, also known as analogy-based effort estimation, is usually used as a synonym for case-based reasoning in machine learning. More precisely, analogy-based reasoning is an application of case-based reasoning in the machine learning field, both sharing many similarities in the application process and algorithms involved and their differences are also discussed in this chapter.

There are three broad categories of learning determined by the type of "feedback" available to a learning system [136]. These are: *Reinforcement learning, Unsupervised learning* and *Supervised learning*. These will be discussed in the next three subsections. My focus will be on supervised learners because they take advantage of the fact that in classification problems, the class label can provide additional information regarding the usefulness of a feature to the problem at hand [42] and I know what class I'm interested in.

2.1.1 Reinforcement learning

Reinforcement learning is where an algorithm learns through a series of rewards or punishments inspired by behaviourist psychology [78, 136]. Reinforcement learning was introduced by Christopher Watkins [166] in 1989 using Q-learning. Q-learning is a form of model-free

reinforcement learning. Q-learning provides the capability of learning — acting optimally in Markovian domains by experiencing the consequences of actions. An algorithm tries different actions at different states, then evaluates its consequences of each action in terms of the immediate reward or penalty it receives. By trying all actions, the algorithm learns which action is best overall, based on long-term discounted reward [165].

In reinforcement learning the algorithm interacts with a changing environment in which it must perform a certain goal without any feedback explicitly telling it whether it is close to its goal or not. The environment can be formulated as a Markov decision process [20] as many reinforcement learning algorithms for this context utilize dynamic programming techniques. While reinforcement learning may not be very appropriate for static data sets of SEE, a form of reinforcement learning is often used for unsupervised learning [13], where the algorithm bases its actions on the past rewards and punishments without necessarily even learning how its actions affect the environment.

2.1.2 Unsupervised Learning

In unsupervised learning the labels of training data are not available. The algorithm *learns* patterns in the input even though no explicit feedback is made available. Data clustering [68] is one of the common unsupervised learning tasks. The data clustering algorithm detects potentially useful clusters of input examples. A good representational process of input objects is essential because the level of accuracy the learned model achieves depends intensely on how the input object is represented. Unsupervised Learning is advantageous where the problem can not be clearly specified e.g., Data mining. Data sets used in software effort estimation are often fully labelled, for an example effort data sets in PROMISE [117]. Therefore unsupervised learning may not be very beneficial for effort estimation.

2.1.3 Supervised Learning

In Supervised Learning the algorithm studies training data examples of input-output pairs and learns a function that maps output to input. The difference between supervised learning and unsupervised learning, is that the labels of training data must be known. The task of

supervised learning can be represented as follows [136]:

> Given N examples of input-output pairs as a `training set`
>
> $(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$,
>
> where an unknown function $y = f(x)$ generates each y_j,
>
> the algorithm discovers a function h that approximates the true function f

where x and y need not be numbers; they can be any value, h is the hypothesis function. Searching through the space of possible hypotheses for one that will perform well, even on unseen targets beyond the training set for example $\hat{y}_{N+1}$, the algorithm learns.

Supervised learning algorithms are trained using labelled examples, such as an input where the desired output is known [139]. Input data is called training data. A prediction model is adjusted through a training process where it is required to make predictions. The learning algorithm receives a set of inputs along with their corresponding correct outputs, and the algorithm learns by comparing its predicted output with correct outputs to find differences. Figure 2.1. These predictions are corrected if wrong. The training process stops when the model achieves a desired level of accuracy on the training data [28].

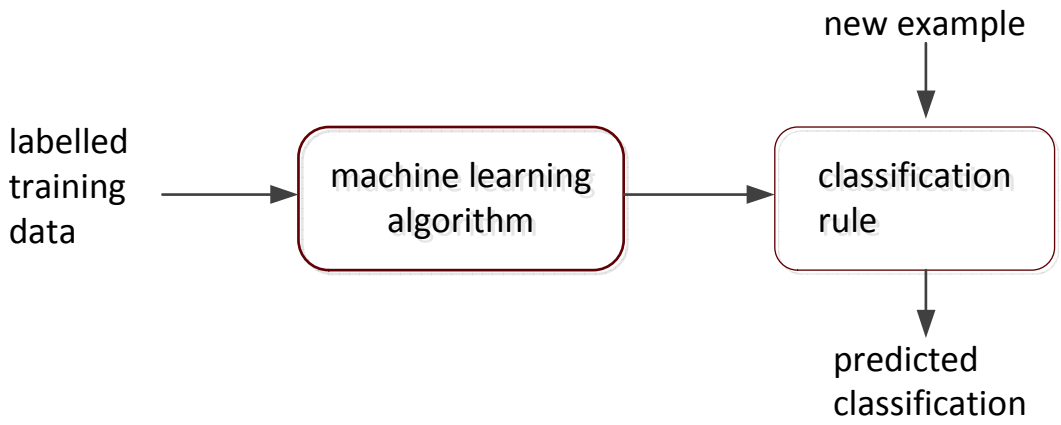


Figure 2.1: The machine learning approach to classification adapted from [140]

The learning problem is called *classification* when the output y is one of a finite set of values. And when the output is a continuous number the learning problem is called *regression* [136]. Supervised learning is usually used in applications where historical data can be used to predict likely future events. Some of the approaches and algorithms proposed for supervised learning

include:

Random forests - A random forest can be defined as a classifier made up of a collection of tree-structured (decision trees learning [133]) classifiers such that $\{h(\mathbf{x}, \Theta k), k = 1, ...\}$ where the $\{\Theta k\}$ are independent identically distributed random vectors [26]. While random forests are more computationally intensive than CBR, studies do not show that they are consistently better than CBR in SEE [113].

Artificial neural networks [53], Neural networks (NNs) are composed of simple elements operating in parallel, inspired by natural biological nervous systems. The connections between elements are used to determine the network function. Neural networks are trained by adjusting the values of the connections between elements in such a way that a particular input leads to a specific target output. Many input/target pairs are needed to train a network [36]. The advantage of using neural networks for prediction is that after learning from examples only, they may be able to find hidden and strongly non-linear dependencies, even in presence of significant noise in the training set. The disadvantage is that the error of prediction cannot be generally estimated [127] and also NNs can learn the dependency valid in a certain period only.

Naive Bayes classifiers [46] - is based on the Bayes' Theorem and the maximum posteriori hypothesis. The naive assumption of class conditional independence is often made to reduce the computational cost [101]. Studies do not show that naive Bayes classifiers are consistently better than CBR in SEE.

Case-based reasoning (CBR) [1], is a well-established methodology with broad applications such as medical science, finance, mechanics, and electronics [104]. CBR offers some distinct advantages in SEE over the previously discussed ML techniques. CBR needs only to deal with problems that actually occur in practice, while generative systems must handle all possible problems. This enables CBR to deal with poorly understood domains (such as SEE) since solutions are based upon what actually happened rather than chains of rules in the case of rule based systems. CBR systems are also able to identify failed cases, enabling practitioners to discover potentially high-risk projects [150].

The fundamental principle of CBR is as follows (see Figure 2.2): when a new case for estimation is provided, the most similar past cases are selected to predict the target unit of the new case by utilising a *similarity measure*.

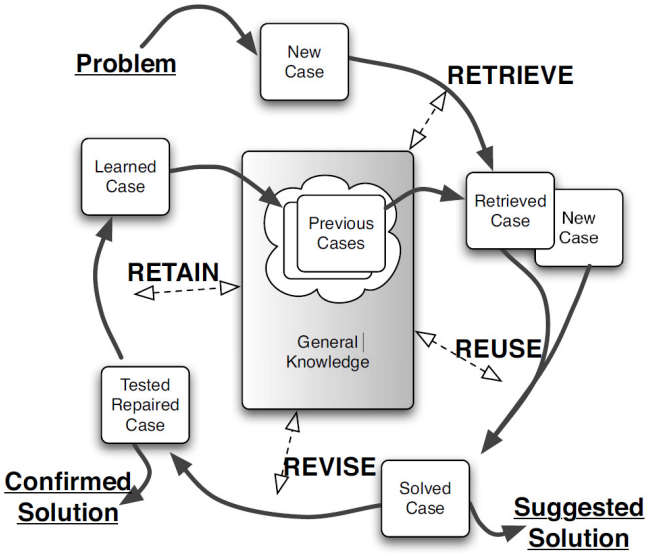


Figure 2.2: CBR process life-cycle [81]

Figure 2.2 shows the 4-stage CBR process which is also sometimes referred to as the R^4 model (Shepperd [143]). The 4-stage cycle described by Aamodt and Plaza [1], consists of:

1. `RETRIEVE` the case or cases most similar to the target problem
2. `REUSE` the past solution and information to solve the new problem
3. `REVISE` the proposed solution and to better fit the target problem
4. `RETAIN` the parts of current solution and information in the case-base for future problem solving

The main activities of the 4-stage cyclical CBR process are: the identification of a `problem` as a new case, then `retrieve` similar cases from case base repository, the `reuse` of knowledge derived from previous cases and the suggestion of a solution for the new case. It may be neccessary to `revise` the predicted solution in the light of actual events and the outcome `retain`ed to augment the repository of completed cases [150].

2.1.4 The choice of classifier

Having looked at different learning styles and classifiers, how do we choose from among multiple inconsistent options? One alternative is to select the simplest option consistent with the data [136], also parsimony helps avoid over-fitting. This principle is called `Ockham's razor`, used to argue against all sorts of complications. Defining simplicity is not easy, but it could be argued that a degree-1 polynomial is simpler than a degree-7 polynomial. The term "simplicity" closely related to prior expectations while "complexity" could be measured by the number of input parameters [140]. Therefore the choice of classifier or learning style may depend on reasons such as:

- Do you have enough training examples, that is, do you have sufficient data sets and are these data sets labelled or not,
- does the classifier show good performance on the training set,
- and the classifier is not too complex, that is, classifiers should be "as simple as possible, but no simpler"

Looking at preceding points: Available software effort datasets are fully labelled therefore a supervised learning is the best option. The next item to consider is a "simple" classifier under supervised learning that shows good performance on training sets. CBR has shown consistent good results on different datasets for example experiments such as using ESTOR [125] and ANGEL [150]. The "simplicity" of CBR is due to the fact that it bases its estimates on real life past examples. This is a familiar mode of human problem solving [98]. Users often find it easy to understand and accept solutions from analogy based systems [112] since these are obtained via a form of reasoning similar to human problem solving, as opposed to the somewhat arcane chain of rules or neural nets[144, 150]. It is for these reasons that CBR is chosen.

Analogy-based software effort estimation is an example of a CBR strategy for problem solving. In effort estimation each project is represented as a case. Each case is characterised by features such as: *the number of interfaces, function points, customer type.* These features in supervised learning would be represented by $(x_1, x_2, \ldots x_p)$ and completed projects also have extra target features, for example total effort (person-days) represented by y. Thus the case-base is

conceptually an $n \times p$ matrix where there are n cases and p features. Often the features that are included in the case-base are more due to happenstance and availability rather than because it is known that there are well defined relationships with the case solution. Features may be collected for all kinds of reasons. Moreover, there may exist multicollinearities amongst these features. Consequently, as is common with the majority of machine learning techniques, it is widely acknowledged that not all features are of equal importance [51, 157]. Thus Analogy-based estimation (ABE) systems will benefit from optimisation of the feature sets.
Feature set optimisation can be accomplished by means of feature weighting which has the effect of stretching or compressing the feature space thus impacting the proximity of cases (projects) and thus modifying the set of neighbour projects that are used to donate solutions. Such problems are NP-hard and for non trivial numbers of cases and feature sets present significant computational challenges.

2.2 Feature Weighting

A slightly less daunting approach, although still NP-hard, is feature subset selection where features are assigned weights of $\{0, 1\}$. Until recently feature subset selection has been the dominant approach within software effort prediction. Effort estimation methods using feature weighting tend to outperform feature subset selection methods [153]. This is also reflected in other domains other that SEE, where results indicate that hard removal of features employed by feature subset selection is sometimes undesirable and better results can be obtained by simply down-weighting the less important features [42, 95]. Therefore it is worthwhile to investigate the more difficult general case of feature weighting instead of feature subset selection. The different feature weighting techniques available to SEE are discussed in the next chapter where I present a literature review on how feature weighting has been employed in SEE. Further discussions about feature weighting are presented in the following sections.

Feature weighting is a process commonly employed in machine learning to deal with the

problem of high dimensionality. For small data sets, large dimensionality of feature space may lead to over-fitting [96]. Features are assigned different weights to reflect their relevance to the output (predicted value). For example, irrelevant or redundant features are assigned a very low weight value. Classification performance is improved since the focus is on most important aspects of data.

The feature weighting techniques are usually embedded in ABE algorithms employing different variants of the similarity distance function shown in Equation 2.1.

$$\text{Similarity}(T, S) = \sum_{k=1}^{p} f(T_k, S_k) \times w_k \tag{2.1}$$

where T is the target case, S is the source case, f is the similarity function, p is the number of features and w_k is the k^{th} feature weight where $1 \leqslant k \leqslant p$. Typically, but not necessarily, f is some variant of Euclidean distance.

There are many objectives for feature weighting, but the most important ones are[137]:

a) to improve model performance and avoid over-fitting, i.e in the case of clustering better cluster detection and improve prediction performance in the case of supervised classification,

b) to provide more cost-effective and faster models,

c) to gain a deeper insight into the underlying processes that generated the data, and

d) to obtain a subset of features through feature weighting for improving prediction accuracy or decreasing the size of the structure without significantly decreasing prediction accuracy of the classifier built using only the selected features [97].

There are three general categories to finding feature weights for a particular case base. The categories are based on how the the feature weight search combine with the construction of the classification model. These are – the *Filter methods*, *Wrapper methods* and *Embedded methods*. The next three subsections provide a common taxonomy of feature weight search methods, and for each technique highlighting the advantages and disadvantages, as well as some examples.

2.2.1 Filter Method

The main idea behind the filter approach is shown in Figure 2.3. The induction algorithm is considered as a black box. The dataset is usually partitioned into training data and test data. Filters use statistical or other general information that can be extracted from the data set alone to attempt to determine the important features. This importance is then reflected in the feature weights. Filters have lower complexity but at the expense of accuracy [86]. An example would be to use principal components analysis.

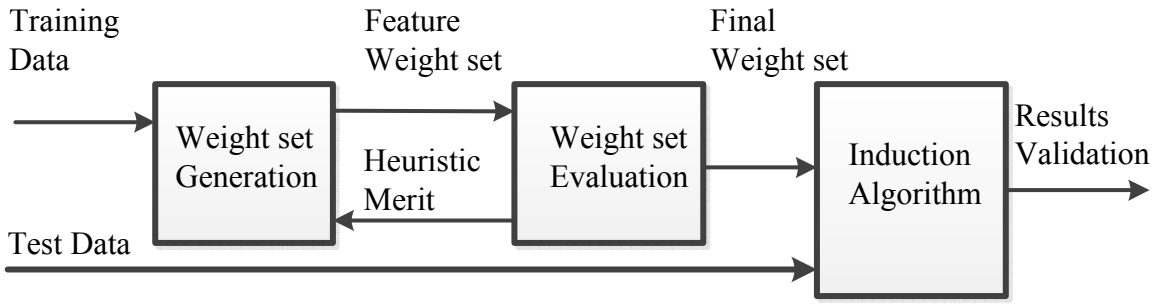


Figure 2.3: The filter approach to feature weight set selection adapted from [93]

The main disadvantage of the filter approach is that the filtering process totally ignores the effects of the selected features weight set on the performance of the estimation algorithm. Filters do not take into account the biases of the algorithms and features that are independent of the induction. In terms of supervised learning which is the focus of this thesis, a study by Kohavi and John [93] reported several issues with filters that define feature relevance independently of the learning algorithm. They highlighted the following problems: filters exclude less relevant features even though inclusion of these features may actually aid performance, filters include correlating features but inclusion of such features may actual hurt performance. It is for these reasons that filters are not pursued any further for this research.

2.2.2 Wrapper Method

An alternative to filter method, but far more computationally demanding method, is the wrapper approach [69]. The wrapper methodology, popularized by Kohavi and John [93], offers a way to address the problem of variable selection. The optimal features weight set

should depend on the specific heuristics and biases of the induction algorithm [162]. Hence, wrapper methods employ a specific induction algorithm to evaluate the usefulness of selected features weight set, and offer a powerful and yet simple way to address the problem of features weight set selection, irrespective of the chosen learning machine method [93]. In simplified terms the induction algorithm is considered as a black box. The dataset is usually partitioned into training data and test data. The feature weight set selection algorithm searches for a good weight set using the induction algorithm itself as part of the function evaluating feature subsets. The final weight set is the feature weight set with the *best* evaluation, and is run on the induction algorithm. The resultant classifier is then evaluated on the test data.
Given a predefined induction algorithm, a typical wrapper model will perform the following steps:

Step 1: searching a weight set of features,

Step 2: evaluating the selected weight set of features by the performance of the induction algorithm,

Step 3: repeating *Step 1* and *Step 2* until the desired evaluation accuracy is achieved.

The main idea behind the wrapper approach is shown in Figure 2.4. A general framework for wrapper methods of contains three major components:

- Feature weight set search - how to search the weight set of features from all possible feature weight set,
- Weight set evaluation - how to evaluate the performance of the chosen classifier, and
- Induction Algorithm.

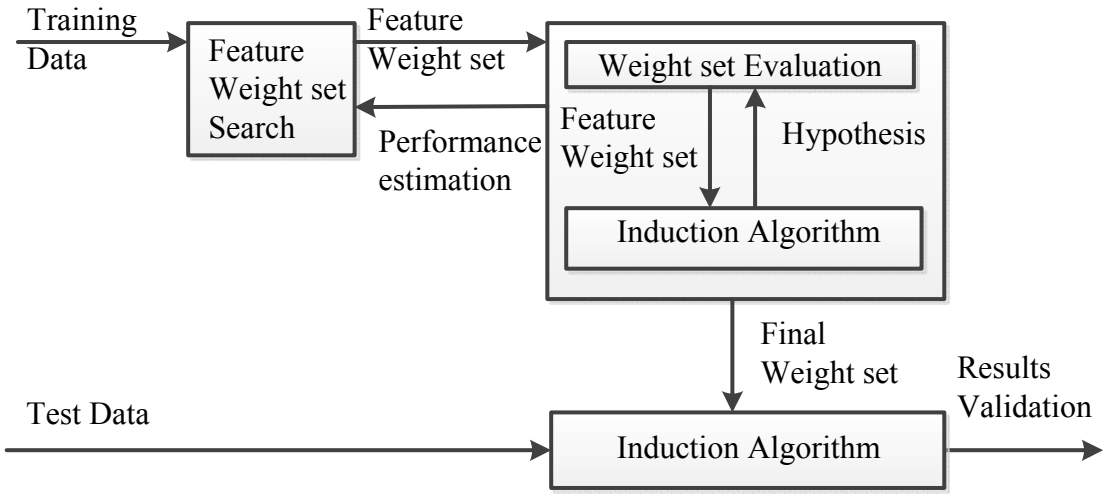


Figure 2.4: The wrapper approach to feature weight set selection adapted from [93]

The feature weight set search component produces a set of weights for features and the weight set evaluation component uses the induction algorithm to estimate the performance, which is returned back to the feature weight set search component for the next iteration of feature weight set selection. The feature weight set with the *best* estimated value will be chosen as the final weight set. The resultant induction algorithm is then evaluated on test data that is not used in during the training process [93]. Performance assessments are usually carried out using a validation set or by cross-validation [51].

In practice, one needs to define: (i) how to search the space of all possible variable subsets; (ii) how to assess the prediction performance of a learning machine to guide the search and halt it; and (iii) which predictor to use [51].

An exhaustive search may be performed, if the number of features is small. This is true for feature subset selection. The size of search space for p features is $O(2^p)$, thus an exhaustive search is impractical if p is large; feature weighting is even worse, depending upon how many weight values you allow per feature. The problem is known to be NP-hard and the search becomes quickly computationally intractable. A wide range of search strategies can be used, including: heuristic search, hill-climbing, best-first, and greedy search algorithms [137].

The hill-climbing strategy expands the current search set by adding weights and moves to the

weight set with the highest accuracy, terminating when no weight improves over the current set [162]. The best-first strategy is to select the most promising set that has not already been expanded and is a more robust method than hill-climbing [93]. Greedy search strategies seem to be particularly robust against overfitting and computationally advantageous [51, 162]. Greedy search strategies come in two flavours - *forward selection* and *backward elimination*. Forward selection refers to a search that begins at the empty set of weights, then most promising are progressively added into the weight set, — stopping when all available weights have been tried — weights that do not improve accuracy are discarded (not added to the set), whereas backward elimination begins with the full set of weights and the least promising weights are progressively eliminated, until we end with the best weight set. The search component aims to find a weight set with the highest/best evaluation, using a heuristic function to guide it. Since the actual accuracy of the classifier is unknown, accuracy estimation is used as both the heuristic function and the valuation function in the weight evaluation phase [51].

Generally wrappers are found to perform better than filters (see Kohavi and John [93]). If the learning machine is considered as a black box, wrappers are remarkably simple and universal. Popular predictors that use wrappers include decision trees, naïve Bayes, CBR predictors, and support vector machines [51]. These are widely used in analogy-based effort estimation.

2.2.3 Embedded Method

The third approach, termed embedded techniques, is hybrid of wrapper and filter methods. The search for an optimal subset of features is embedded into the classifier construction, and as such can be seen as a search in the combined space of feature weight set and hypotheses. Embedded approaches are specific to a given learning algorithm. An example would be feature weight set search using the weight vector of support vector machines (SVM) [52]. Embedded methods have the advantage that they take into account the interaction between the classification model and selected feature weight set, contemporaneously being far less computationally intensive than wrapper methods since some features are already "filtered" out. The disadvantage of embedded method is that feature selection is classifier dependent

[137].

The focus of this thesis is on estimation by analogy and wrappers have been found to perform better than filters and used by popular CBR predictors. Other estimation approaches are not in the scope. Discussion of analogy-based estimation techniques is continued in the next section.

2.3 Software effort estimation techniques

Software effort estimation can be defined as approximating the quantity of work required to develop a software project from start to finish. There is a plethora of software effort estimation models and methods in practice, having different strengths and limitations. There are different schemas for classifying software effort estimation techniques, which can be grouped into three main categories [24]:

Expert judgement-based: Expert judgement methods are techniques whereby the estimates are generated based on a considered opinion process. Experienced estimators familiar with the project development environment are required to make estimates. This makes the estimation result highly dependent on the expert's abilities and decision making therefore is not repeatable in another environment for different projects [75]. Although expert judgement based methods are the most used technique [123] in practice, they have several limitations [70]. Experts can easily be influenced by external factors, such as being biased because of: client's expectation of cost [77], misleading or irrelevant information during estimation request stage [73] and by cultural related characteristics [73]. Also Expert judgements in most instances involve over-optimism [72]. The strength of expert judgement methods is that when quantified, empirical data is absent. For example if a new software project's effort is required but no similar past projects exist, data driven methods would fail but an expert could use his knowledge of past seemingly dissimilar projects and his experience in effort estimation to come with a reasonable estimate. Some of the examples of expert judgement methods include: Work Breakdown Structure (WBS) [24], Wideband Delphi [24, 25] and Planning Poker [124].

Model based techniques: algorithmic cost models such as Function Points Analysis [5] and Constructive Cost Model (COCOMO) [25] by Boehm (1981). These model based techniques require a certain amount of completed historical project data. Therefore for early software life-cycle where many of required parameters may be unavailable these models would not be suitable. Incomplete or partial requirement specifications would prevent these algorithmic models from producing the required estimates [80]. Often models come with parameters. The problem is that they need calibration and generally independent studies show poor performance e.g., Kemerer 1987 [79] onwards.

Learning Oriented techniques: New developments in modelling and pattern recognition encouraged researchers to investigate effectiveness of techniques previously used in domains other than SEE. These techniques attempt to automate improvements achieved in the estimation process by constructing models that "learn" from previous experience [24]. Some examples include: starting in 1995, Sirnivasan and Fisher looked at Regression trees [159], in 1997 Shepperd and Schofield investigated Case-based reasoning [150], Genetic programming [29] by Burgree and Lefley in 2001, Pedrycz looked at Rule-based expert systems [130] in 2002, Fuzzy systems [174] in 2004 by Xu and Khoshgoftaar, in 2006 Oliveira investigated Support vector regression [128] and Neural networks by Idri et al. [66] in 2007. Learning oriented techniques have two distinct advantages: First, being, the capability to model a complex set of relationships between features and the output value (value being predicted). Second, learning oriented techniques are able to learn from historical datasets therefore use these datasets to predict new cases. Amongst these techniques, case based reasoning (or synonymously estimation by analogy) is the most commonly used technique by researchers. Further discussions about estimation by analogy and its research directions are presented in the following sections.

2.4 Analogy-based Software effort estimation

Analogy-based software effort estimation is a typical example of a CBR strategy for problem solving. CBR is the process of solving new problems based on the solutions of similar past

problems see Figure 2.5 adapted for effort estimation. When a new problem is entered into the CBR system, studies such as [143] considered the new problem as a case that comprises two parts (pretty much inherent to the approach). The basic data structure of the system is formed by two parts — the description part and the solution part forming . The description part consists of a set of features that describe the case state at the point at which the problem is presented. The solution part describes the solution for the specific problem (the problem description part). Figure 2.5 illustrates the problem description part and the solution part forming the basic data structure of a typical CBR or analogy-based system. The fundamental principle of ABE is as follows; when a new project for estimation is provided, the most similar past projects are selected to predict the effort/cost of the new project by utilizing similarity measure.

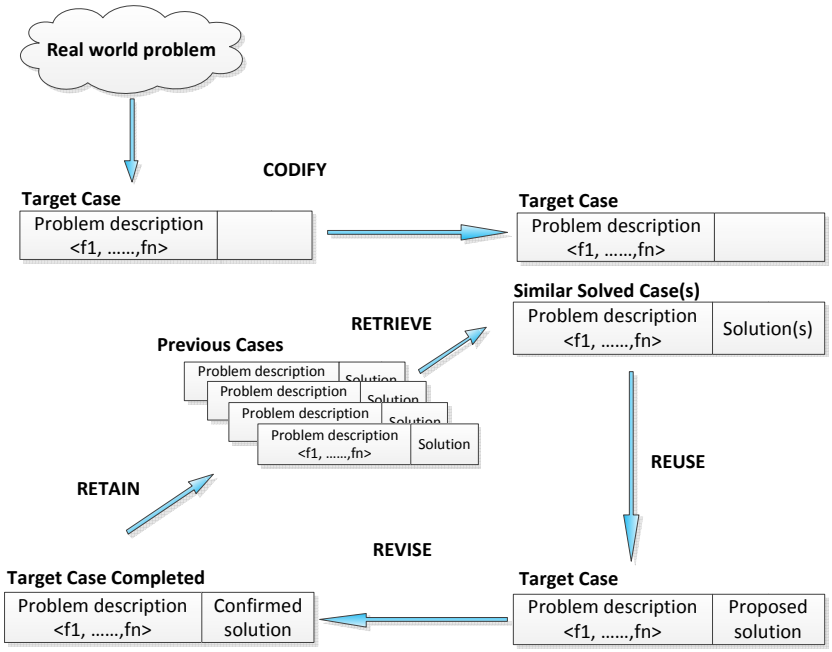


Figure 2.5: CBR Process adapted from Shepperd [144]

The main principle of ABE is that *the more similar the projects are, the more similar their effort will be* [84]. Generally, there are three key parameters for ABE: the similarity measure function, number of nearest neighbours (denoted as k-NN) and the adaptation rules [104]. The effectiveness of analogy based estimation may depend on the similarity measure function

and the number of nearest neighbours (k-NN) used . Each of these parameters has crucial impact on the estimation accuracy of CBR. In the next subsequent subsections I introduce some of the similarity measures available for ABE, the choice of k-NN and performance metrics.

2.4.1 Similarity Measure

The similarity measurement is an essential part of software effort estimation by analogy. A similarity function is used to measure the degree of similarity between software projects. Similarity measurement plays significant role in the identification of projects closest to the project being estimated (Nearest Neighbour k-NN — k is the number of nearest neighbours that are used as donor projects to predict effort of target project), therefore having an influence on estimation accuracy. A study by Mendes et al. [115] comparing different types of distance metrics in analogy software estimation, reported that using different distance metrics may yield divergent results. There are many different types of similarity functions, such as the Minkowski distance, but Euclidean similarity and Manhattan similarity based similarities measures are widely used in software effort estimation by analogy [104]. The Euclidean distance metric is presumably the most frequently used in software effort estimation by analogy for distance measures [81]. Euclidean distance metric is based on the principle of Pythagorean Theorem deriving a straight line distance between two points in m-dimensional space. Standardisation is used to overcome problems of units. The figure 2.6 illustrate the concept of Euclidean distance measure between two objects in 3-dimensional space ($x_1, x_2,$ and x_3).

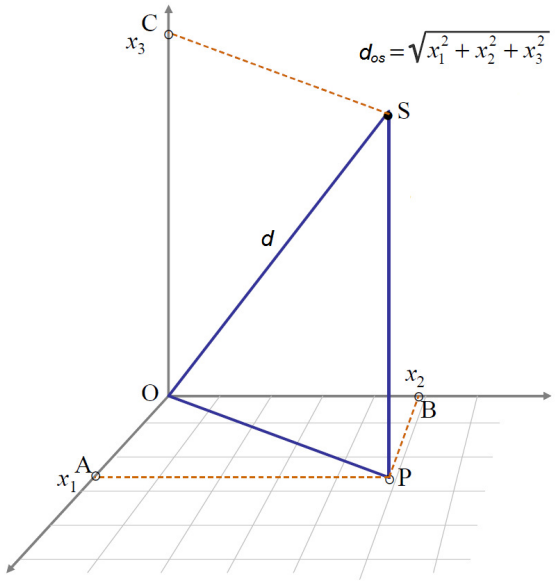


Figure 2.6: The Euclidean distance in 3-dimensional space [49]

In general, the un-weighted Euclidean distance between two points X = $(x_1, x_2, \ldots, x_n)$ and Y = $(y_1, y_2, \ldots, y_n)$, in Euclidean n-dimensional space, is defined and calculated as:

$$d_{x,y} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.2}$$

Alternatively weights can be applied to individual features to reflect their relative importance, because individual features may have different contributions to the overall project cost. In the literature there are several approaches focusing on feature weighting. Sigweni and Shepperd [153] identified more than 12 distinct approaches. For example, given w_i and w_p as the weights of ith and p^{th} project features. The weighted Euclidean distance would be calculated as:

$$d_{x,y} = \sqrt{\sum_{i=1}^{n} w_i (x_i - y_i)^2} \tag{2.3}$$

The Euclidean similarity (SIM) based on the Euclidean distance between two projects is calculated as follows:

$$\mathrm{SIM}(\mathbf{c_1}, \mathbf{c_2}, p) = 1/\left[\sqrt{\sum_{j \in p} w_j \mathrm{DIS}(\mathbf{c_{1j}}, \mathbf{c_{2j}})} + \delta\right] \tag{2.4}$$

$$\mathrm{DIS}(\mathbf{c_{1j}}, \mathbf{c_{2j}}) = \begin{cases} (\mathbf{c_{1j}}, \mathbf{c_{2j}})^2 & \text{if the features are numeric} \\ 0 & \text{if the features are categorical and } \mathbf{c_{1j}} \neq \mathbf{c_{2j}} \\ 1 & \text{if the features are categorical and } \mathbf{c_{1j}} = \mathbf{c_{2j}} \end{cases} \tag{2.5}$$

where c_1 and c_2 represent the cases, $\mathbf{c_{1j}}$ and $\mathbf{c_{2j}}$ represent the jth feature values of selected cases, p is the total number of features and DIS is the feature dissimilarity and δ is a small constant to deal with the situation when the denominator equals 0, Li et al. [104] set $\delta = 0.0001$.

Shepperd and Schofield [150] identified other different forms of similarity measure that could be used, such as:

Manually guided induction. Key features are manually identified by an expert.

Template retrieval. A range of values for ranges is supplied by the user and all cases that match the criteria are retrieved (similar to database queries).

Goal directed preference. Cases that have the same goal as the target case are selected.

Specificity preference. Cases that match features exactly are selected over those that match generally.

Frequency preference. Cases that are most frequently retrieved are selected.

Recency preference. Select cases that have been recently matched over cases that have not been matched for a period of time.

Fuzzy similarity. Implemented based on concepts such as at-least-as-similar and just-noticeable-difference.

Among many different distance metrics available for non-continuous variables, Jaccard distance [161] for binary distance and Gower distance [48], based metrics are some of the widely accepted. Only the Euclidean distance measure is considered in this thesis, since it is the

most used distance measure in software cost estimation studies [81, 85]. Also the Java shell (ArchANGEL) used for this research adopted the Euclidean distance measure.

2.4.2 Choice of *k* for *k*-NN

k refers to the number of analogies sought, close to the target project being estimated [104]. In order to estimate the cost of a new project in software effort estimation by analogy, one or more historical software projects are required. The decision on the number of analogies (k) that is required to predict the cost of target case may be one of the important issues in software effort estimation by analogy. See Figure 2.7 for different scenarios where a different k value would benefit different target cases (A, B and C). For example for Case A $k = 5$ may not yield the best result, this is due to a_5 being required to make number of analogies equal 5 to estimate Case A. Donor a_5 is clearly to not very *similar* to $a_1 \ldots a_4$ but $k = 5$ may be perfect for Case C since all donors are all almost equidistant from C. Case B represents a situation where either $k = 2$ or $k = 4$ will yield similar results, therefore shows that there are situations where different k values would be suitable for the same target case.

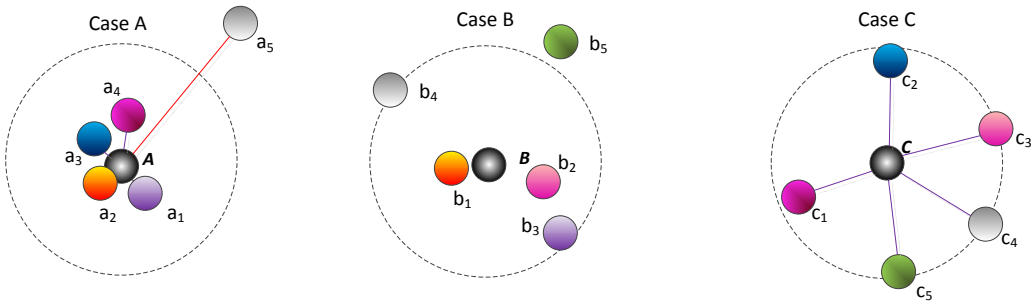


Figure 2.7: Nearest neighbours (k-NN) example

The optimal value of k has been the subject of debate in literature [81], also begs the question is there an optimal value? Studies such as [12, 31, 164] suggested $k = 1$, while other studies [74, 116, 150] and [61] recommend k equals to two or three. Other authors investigated other approaches to determine the suitable value of k. Keung [82] proposed a dynamic approach in an attempt to determine the best possible value of k. Idri et al.'s approach involves selecting all projects that fall within a set similarity limit [64]. Therefore any study using $k = \{1, 2,$

3, 4, 5} would cover most of the suggested k values in the literature.
So, in this thesis, experiments will evaluate the performance of the estimating model when to use the different k values from the set {1,2,3,4,5}. This will cover most of the recommended k value in existing research work to enable acceptable comparison.

2.4.3 Adaptation Rules

After the k nearest neighbours are selected, a case adaptation strategy is then applied to the selected donor cases in order to obtain a prediction of effort for the target case. Tthe target case is the new case being predicted. Adaptation rules are mechanisms used to derive a new estimate in order to minimize the differences between donor cases and target case [129]. Various techniques of adaptation have been proposed, such as;

- The *mean* adaptation rule uses the average effort of nearest neighbours. This is a classical measure of central tendency and assumes that similar projects have equal influence with each other.
- The *median* adaptation rule is another measure of central tendency. It takes the median of the effort of selected neighbours and hence more robust than mean as k increases [7].
- The *inverse distance weighted* mean allows neighbouring cases to have different influence on the estimation. Therefore cases more similar to the target case have more influence on the estimate.

This is a very important step in estimation by analogy because it reflects structure of target case on retrieved cases. Adaptation techniques cam be a lot more sophisticated than the above mentioned examples but tend to be hand developed so there is a significant danger of over-fitting e.g., see Srinivasan, K., & Fisher, D. (1995) [159]. Some estimation techniques such as Estor [125] adopt rule based adaptation founded on the rules that are hand crafted (selected via expert judgement [80]) rather than induced. The next subsection explores analogy-based tools and systems, describing how they apply the three key parameters for ABE: the distance function, the number of nearest neighbours and the adaptation rules.

2.5 Analogy-based Tools and Systems

There are various implementations of analogy-based tools and systems for software project effort estimation, such as: Analogical and Algorithmic Cost Estimator (ACE) [164], Finding Analogies for Cost Estimation (FACE) [22], TEAK [91], BRACE [160], AQUA+ [102], `Estor` and ArchANGEL. These tools and systems apply the same principles of CBR reasoning approach discussed in the previous sections. Therefore they have the same design goal, which is automating the estimation process and enabling data-intensive analogy for software effort estimation [81]. However they have different mechanisms for case adaptation and the choice of value for k (i.e. number of analogies). In this section related literature (implementations, performance and limitations) is surveyed from the earliest (`Estor`) to probably the most popular [65] (ArchANGEL) analogy-based systems for software effort estimation. ArchANGEL is the focus of this thesis and its performance on wide range of different data set sizes is well established [153]. Secondly, I have access to the Java shell and its developers.

2.5.1 Estor

In the early 90's Mukhopadhyay et al. [125] developed `Estor` as a proof-as-concept system to evaluate the feasibility of case based reasoning in software effort estimation. Estor is one of the early implementations of an analogy-based to estimate software project effort. Figure 2.8 shows the logical architecture for Estor.

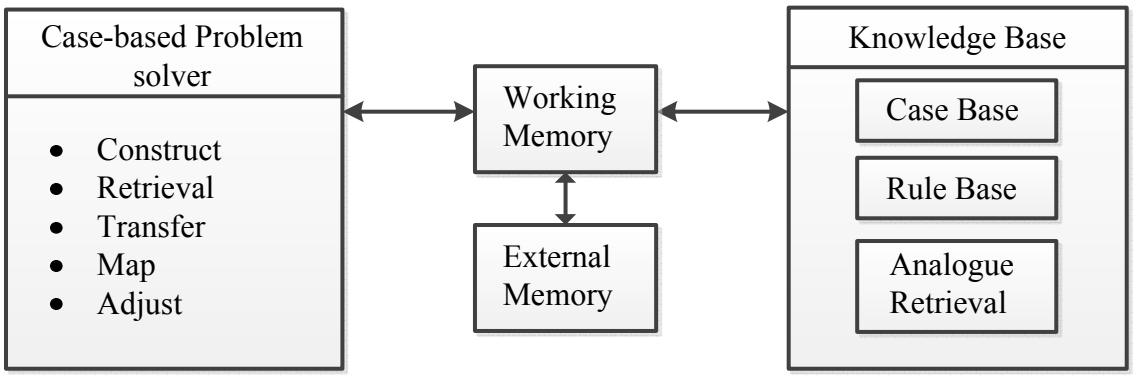


Figure 2.8: Estor Logical Architecture [125]

The Estor framework involves five basic processes (Figure 2.8): *construct, retrieve, transfer, map,* and *adjust*, outlined as follows:

1. The *construct* process is used to develop a representation of the target problem.
2. Estor invokes a domain-specific case selection heuristic such as "similarity distance" to *retrieve* an appropriate source analogies.
3. *Transfer* the solution that achieved the goal in the base case to the target case by referencing the effort attribute of the source project.
4. Estor identifies the differences between the target case and source case and then *maps* them. Bringing their individual attributes one by one into working memory, and all non-corresponding attributes are kept in a list in memory.
5. Based on the differences identified between the attributes Estor will *adjust* the estimate for the initial solution, using a set of rules.

For effort estimation Estor employs $k = 2$ (i.e. 2 analogies), selected via expert judgement [80]. Estor's accuracy performance is assessed via magnitude of relative error or (MRE).
The accuracy of the estimation is inversely proportional to the MRE of a project. To evaluate the accuracy of Estor, the mean MRE (MMRE) for a set of projects is calculated. In a study by Mukhopadhyay et al. [125], based on Kemerer dataset [79], they demonstrated through statistical based results that there is no difference between Estor and Expert judgement approach, both being more consistent than either COCOMO model [25] and Function points [5]. Unfortunately, Estor has limitations: as already stated in section 2.5.1, Estor requires an expert to choose the number of analogies and to derive rules for adaptation. Another point is that since the rules are derived for features of a particular data sets, this limits their generalisability [143].

2.5.2 ArchANGEL

ArchANGEL CBR tool[143] is the latest version of ANaloGy softwarE tooL (ANGEL) [151] developed by Shepperd, Schofield and Kirsopp. ArchANGEL (or ANGEL) is the dominant automated software effort estimation based on analogy [65]. Many researchers such as [15, 84, 85, 102, 104, 143, 147, 148, 150, 151, 164] on analogy-based research published findings

based on the results of ArchANGEL. It is a tool based on a k-NN system to estimate software projects effort based analogy. ArchANGEL is an implementation based on the case-based learning algorithm in [3], where it adopts a similarity function and the normalization strategy for the different data types of feature values [81].

ArchANGEL separates the process of effort estimation into three key components (Fig 2.9):

i The `template` is like a data dictionary so that the tool can have a shell architecture and is not restricted to a particular data set or features.

ii Information captured in a template is used to build `project case base`, which is the repository for project data.

iii Utilising source projects in the project case base, `estimate generation` allows the user to generate estimates for target projects.

The interaction between the key components of the ANGEL tool are shown in Figure 2.9. Prior to project case data being stored, a template editor is used to outline the environment from which the data will be collected, creating a template. The subsequent template is saved as a text file. The project case editor then uses this template file to create a project case base. Finally the estimate generator can use the project data to generate estimates for any new projects. Preferably, an estimate should then be added to the project case base so that it may be compared to the final estimate, (this is not mandatory in ANGEL)[141].

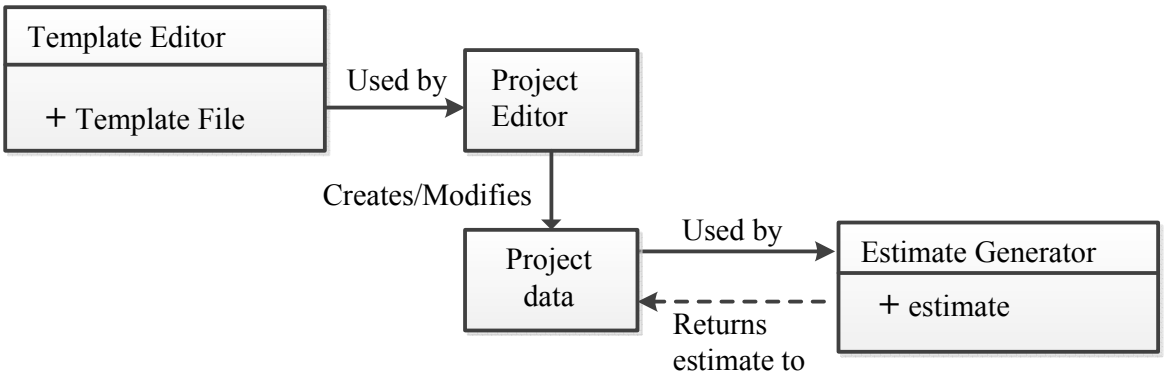


Figure 2.9: ANGEL schematic[141]

ArchANGEL employs the Euclidean distance metric to measure the similarity between target project and all potential source analogues from the case base. It implements various feature subset selection and case subset selection algorithms such as Random search, Exhaustive search, Hill climbing, Forward and backward sequential selection algorithms. These algorithms are evaluated using the performance indicators used in the estimation process such as bootstrapping, MMRE and PRED(p) but MAR is also available enabling evaluation based on SA [149]. Therefore, avoiding issues related to MMRE and PRED(p) discussed in the previous sections. For validation, techniques such leave-one-out cross-validation (LOOCV) procedure are employed [39]. Although LOOCV is computationally intensive for larger datasets when using a wrapper, the advantage is that the results are deterministic. By comparison, $m \times n$ fold cross validation also available in ArchANGEL will depend upon the random allocation of cases to the individual folds and so there is often some variability in the results [154].

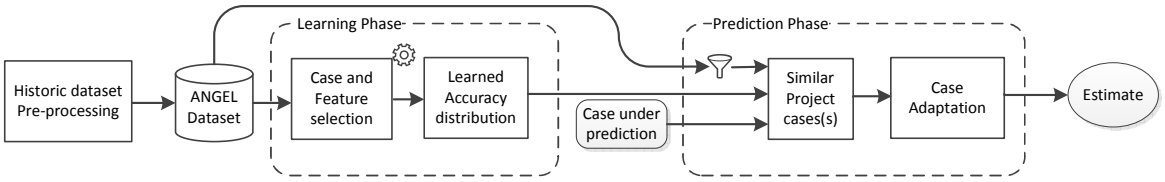


Figure 2.10: ArchANGEL Logical Architecture

ArchANGEL is very popular with researchers due to the completeness of its implementation:

- Supporting various kinds of validation techniques.
- Uses the wrapper approach whose advantages have been discussed previously
- Multiple search heuristic algorithms to perform feature and case selection on the dataset
- A user-friendly graphical GUI as its primary user-interface.
- Multiple case adaptation rules
- Any project dataset can be used (i.e. it does not assume that estimators will use a particular dataset, therefore the estimator can use any project dataset available to setup an estimation instance in ArchANGEL).

- Normalisation, allowing features to be re-scaled so that the influence of a feature is not related to the choice of unit.

Also, in-terms of feature weighting Kirsopp and Shepperd [87] using forward sequential selection (FSS) algorithm reported improved efficiency and very good prediction results over no weighting versions. Their study used a real life industrial data set, the so-called 'Finnish dataset' which contains 407 cases described by 90 features. The features are a mixture of categorical, discrete and continuous. Prediction made using no feature weighting had an average error of 50.8% while FSS had 13.1%. In-terms of efficiency FSS required only 243 evaluations while hill-climbing required 74433 evaluation during for 113 hill climbs to achieve *similar* accuracy as FSS. Therefore hill climbing can be very computationally intensive (but you can choose to do as many or few restarts with hill climbing). It must be noted that this form of multiple-start hill climbing does not have an in-built end point. Therefore it can run indefinitely, but likely to show diminishing returns as it converges on an exhaustive search [87].

2.6 Advantages of Analogy-based estimation

The objective of software estimation is to provide a very good software effort prediction for project managers. Many researchers explored a wide range of techniques and approaches to software effort estimation (see a systematic review of software development cost estimation studies by Jørgensen and Shepperd [76]). The encouraging accuracy of estimates from experiments such as using ESTOR [125], ANGEL [150] and [113] demonstrates that analogy-based software effort estimation is a viable alternative to other software effort estimation methods. Note I am not arguing that ABE is always the answer, but in addition to its good prediction performance, analogy-based estimation also offers estimators some advantages [80, 144, 150, 164] over other methods, and these are:

1. *The basis for an estimate is easy to understand*
   Analogy-based estimation is different to input-output models. It bases its estimates on real life past examples. This is a familiar mode of human beings problem solving [98],

and this may explain why people are more comfortable estimating based on analogy. Users often find it easy to understand and accept solutions from analogy based systems since these are obtained via a form of reasoning similar to human problem solving, as opposed to the somewhat arcane chain of rules or neural nets [144, 150].

2. *Analogy-based estimation is useful where the domain is difficult to model*
   It is widely accepted that many factors influence the effort needed to complete a software project. There is limited knowledge about how these factors interact with each other, or how best to model the wealth of factors via software metrics. Estimation by analogy can be used effectively without having a clear model of how effort is related to other project factors. It relies primarily on selecting a past project that is similar to the target project, rather than postulating a general relationship between effort and other project characteristics that applies to all projects. Small historical data sets may be sufficient to develop simple algorithmic models, provided the data does not prove too noisy. However noise, unaccounted for variations in dependent variables, is at the crux of domains which are difficult to model. Shepperd et al. (1996) give an example of a data set of 8 projects for which no statistically significant relationships can be found. An algorithmic model based on this data set would be suspect. Nevertheless, the accuracy of analogical estimates for this data set was comparable to that of other much larger data sets.

3. *Analogy-based estimation can be used with partial knowledge of the target project*
   ABE addresses the problem of partial knowledge by allowing estimators to use any information they have available to search for and therefore select relevant analogues, rather than prescribing particular inputs.

4. *Analogy-based estimation has the potential to mitigate problems with calibration*
   ABE has the potential to provide accurate estimates even using a different organisation's data, as long as an appropriate analogue for the target case can be found within the data set used and the features are both appropriate and measured in a consistent manner. This is possible even where the relationships differ for typical projects of each

organisation, therefore mitigating the need for calibration.

5. *Analogy-based estimation has the potential to mitigate problems with outliers*
   Most project data sets have outliers: that is, projects that differ substantially from the typical projects in the case base on values of their metrics and relationships between them. If the target case is a typical example of projects in the data set, it very is likely that one or more appropriate analogies are present to base the estimate on. Therefore, outliers will have no influence on the estimate at all. However, if the target project is itself an outlier, at least the lack of appropriate donors may make this evident to the estimator.

2.7 Summary

ABE is relatively more effective but like any other prediction technique depends upon characteristics of the dataset, and it will tend to be more effective when discontinuities exist in underlying relationships between effort and the independent variables [85]. Unfortunately ABE is also vulnerable to features that are irrelevant, or even misleading, for the particular prediction task [85]. Feature subset selection also known as feature weighting involves determining the optimum (or at least a better) subset of features that would give the most accurate estimation. In order to assess to what extent the empirical software engineering community has addressed feature weighting in analogy-based software effort estimation a systematic literature review was carried out, an its findings are reported in the next chapter (chapter 3).

Chapter 3

Feature Weighting Techniques for ABE in Software Effort Estimation

This chapter describes to what extent the empirical software engineering community has addressed feature weighting in analogy-based software effort estimation. Although feature weighting and techniques are recognised in empirical software engineering literature, the issue of search space is sparsely addressed. Chapters 1 and 2 highlighted the importance of feature weighting for software effort estimation. So although there is widespread consensus that some form of feature weighting technique is beneficial, there has been no systematic literature review[1] (SLR) of all relevant primary studies, Nor has there been an analysis of the extent of feature weighting techniques (FWTs), how they have been experimentally evaluated and the interaction between performance and data set. Most of this chapter is based on my published SLR [153]. In section 3.1 the objectives of this literature review are outlined, and section 3.2 describes the method of literature identification and section 3.3 describes the findings with a more detailed discussion of key papers relevant to the investigations of this thesis. Section 3.4 concludes this chapter summarising the state of affairs for feature weighting in empirical software engineering literature with links elicited to the work imparted in this thesis.

[1] "(also referred to as a systematic review) A form of secondary study that uses a well-defined methodology to identify, analyse and interpret all available evidence related to a specific research question in a way that is unbiased and (to a degree) repeatable"– Kitchenham 2007

3.1 Objective for the Literature Review

Given the importance of timely and accurate software cost prediction it is unsurprising that there is a large body of published research work, the majority of which has focused on effort prediction since effort is normally the dominant and hardest to predict component of overall cost. For an overview of the extent of this research see the mapping study by Jørgensen and Shepperd [72] and more recently the review of machine learning based studies by Wen et al. [168]. Recall from Chapter 2, for effort prediction a case is usually a project, the case state will be features such as size, development environment, client experience, etc., and the solution is the actual amount of effort utilised. Thus the case-base is conceptually an $n \times p$ matrix where there are n cases and p features. It is widely acknowledged that not all features are of equal importance [51, 157].

This systematic literature review (SLR) aims to identify and empirically evaluate existing feature weighting techniques used in analogy-based software effort estimation studies published between January 2000 and April 2014. The SLR characteristics are summarised in Table 3.1 and described in more detail in subsequent sections.

Table 3.1: Systematic Literature Review Summary

Characteristic	**Value**
Review type	Systematic literature review
Research question(s)	The diversity of FWTs (RQ1), their strengths and weaknesses (RQ2), do FWTs improve predictive performance (RQ3)? and the various approaches to the primary study experimental design (RQ4).
Purpose	Guide researchers and practitioners using estimation by analogy techniques.
Audience	Researchers
Search method	Automated and hand search, citation analysis, previously known articles plus approached authors.
Databases used	BESTweb, IEEE Xplore, ScienceDirect, Google scholar, ACM digital library, Springer
Population	Empirical studies relating to FWT in software effort estimation
Setting	Commercial software projects
Study types	Experiments, case studies, observational studies, simulation.
Inclusion criteria	(i) Refereed paper (journal or conference) (ii) Empirical study (iii) Copy of the article available
Language	English language only
Search end date	April 2014
Located articles dates	2000-2014 plus in press articles

3.1.1 SLR Research questions

The principal aim of this SLR is to answer the question of whether feature weighting techniques improve the predictive performance of CBR prediction systems for software effort? In doing so we need to review the range of FWTs that have been proposed, consider how they have been empirically evaluated and using which data sets. This will enable us to provide guidance for researchers and practitioners and identify areas for further research in order to

improve the performances of current CBR models. To achieve this objective four research questions are formulated and presented below.

RQ1: What is the range and diversity of feature weighting techniques used for software development effort estimation? In answering this question we characterise them using the dimensions from a previous review by Wetterschereck [170]
Rationale: Practitioners can take the identified feature weighting techniques as candidate solutions in their practice. For feature weighting techniques that have not yet been employed in CBR, researchers can explore the possibility of using them as potential feasible solutions.

RQ2: What are the strengths and weaknesses of existing feature weighting techniques?
Rationale: This will be helpful for practitioners to better understand the practical issues around deployment.

RQ3: What is the estimation accuracy of each FWT and how do they compare?
Rationale: to enable us to compare techniques and determine which support the most accurate cost predictions.

RQ4: How has the experimental evaluation been conducted e.g., which performance metrics are used?
Rationale: This helps determine the importance we attach to the evidence e.g., some performance metrics such as Mean Magnitude of Relative Error (MMRE) have been shown to be biased by studies such as [43, 89].

The next section describes the methods adopted to aid in identifying relevant literature.

3.2 Method of Identification of Relevant Literature

The aim any literature review is to identify *all* relevant studies. So that the literature review produced is unbiased and repeatable as much as possible. In this case I provide an overall and coherent picture of how feature weighting in analogy-based software effort estimation has been addressed in the empirical software engineering community. Recall that

the main objective for the literature search was to discover studies that explicitly adopt feature weighting for ABE in empirical software engineering, and how these studies addressed this issue. Therefore the next subsections will discuss the search strategy adopted to achieve this objective. This includes a detailed discussion on the: search terms, literature sources, the actual search process employed, and finally outlining the study selection, data extraction and data synthesis.

3.2.1 Search strategy

Previous systematic reviews, for example [72] reported that automated article searches via on-line databases may lead to low recall rates, may not be thorough and also the likelihood of additional work arising from low precision rates. However, this search was relatively focused so there was little danger of that manifesting in significant proportion. The following sections discuss: definition of search terms, selection of appropriate literature sources and the search process used.

Search terms

The following steps were employed to build the search terms based on [2]:

1. Derive major terms from the research questions.
2. Identify synonyms and alternative spellings for major terms.
3. Use the Boolean OR and AND to incorporate and link alternative spellings and synonyms.

The resulting search strings are described as follows: Software AND (effort/OR cost/) AND (“prediction” OR “estimation” OR “forecasting”) AND (Analogy-based OR “Analogy based” OR “case based reasoning” OR “CBR”) AND (“feature subset selection” OR “feature selection” OR “feature weighting” OR “feature weights” OR “weight” OR “feature significance”).

Literature sources

The search included important software engineering journals as per [72] and conferences which publish literature on software development effort estimation. The search also involved following up the references of included papers this is also known as 'backward snowballing'. I then searched the papers that cited included papers known as 'forward snowballing', which was accomplished by means of Google Scholar (*cited by*).

No restriction was placed in terms of the start date of inclusion period. Any published paper that met the inclusion criteria was selected because the intention was to have a broad coverage since this was the first SLR on the topic based on the literature available at the time. The feasibility of CBR in software effort estimation was carried out in the early 1990's by Mukhopadhyay et al. [125] therefore it was expected to find papers published starting from the mid 1990's or early 2000's to current date 2014.

The initial search was performed on the online bibliographic library BESTweb[2] maintained by Simula Research Laboratory. The rest of the search involved searching five electronic databases (IEEE Xplore, ScienceDirect, Google Scholar, ACM Digital Library and Springer). Some other important resources such as CiteSeer, web of science or DBLP were not considered, because they are almost covered by the selected five electronic databases.

The search terms discussed and constructed in section 3.2.1 were used to search for papers in the selected databases. The search covered title, abstract, and keywords.

Search process

As mentioned previously alluded to, a comprehensive search process of '*all*' relevant sources is required for any SLR. Therefore to achieve this objective the search was divided into the following three search phases described in Figure 3.1.

> **Search Phase 1**: Using the in-built filter the online bibliographic library BESTweb database was searched and potential relevant papers located. BESTweb was searched first because it supplies journal and conference papers on software cost and effort esti-

[2](http://www.simula.no/BESTweb)

mation classified according to research topic, estimation approach, research approach, study context and data set. This provides a good starting point and also leads to identification of potential unknown journals. The built in search filter also help to quickly search for only papers on your estimation approach.

Search Phase 2: Using search terms the five electronic databases were individually searched and then merged the located relevant papers with those from BESTweb forming a set of potential candidate papers

Search Phase 3: The references of the included papers ('backward snowballing') are searched. I then searched the papers that cited included papers known as 'forward snowballing' in order to locate further papers.

Zotero[3], Microsoft® Excel and Dropbox[4] were used to manage and store search results. The search process and the number of papers identified at each phase are shown in detail in Figure 3.1.

3.2.2 Study selection and data extraction

A total of 183 potential candidate papers were identified for inclusion in the SLR from search Phases 1 and 2(see Figure 3.1). Since these papers were from independent sources they were checked to eliminate duplicates. A total of 59 unique papers were obtained after removing duplicates.

The next step involved reading the titles, abstracts, key words or full text of these 59 papers to select relevant papers based on the inclusion criteria defined in Table 3.1. From this 17 relevant studies were selected. Then the references of every included study were searched to identify other relevant studies that might have been missed and also examined citing papers. This effort led to the identification of two further studies, resulting in a total of 19 papers. The quality of the selected studies was linked to inclusion criteria, where only papers from demonstrably rigorously refereed sources were included. Therefore no papers were excluded

[3]https://www.zotero.org/

[4]http://www.dropbox.com/

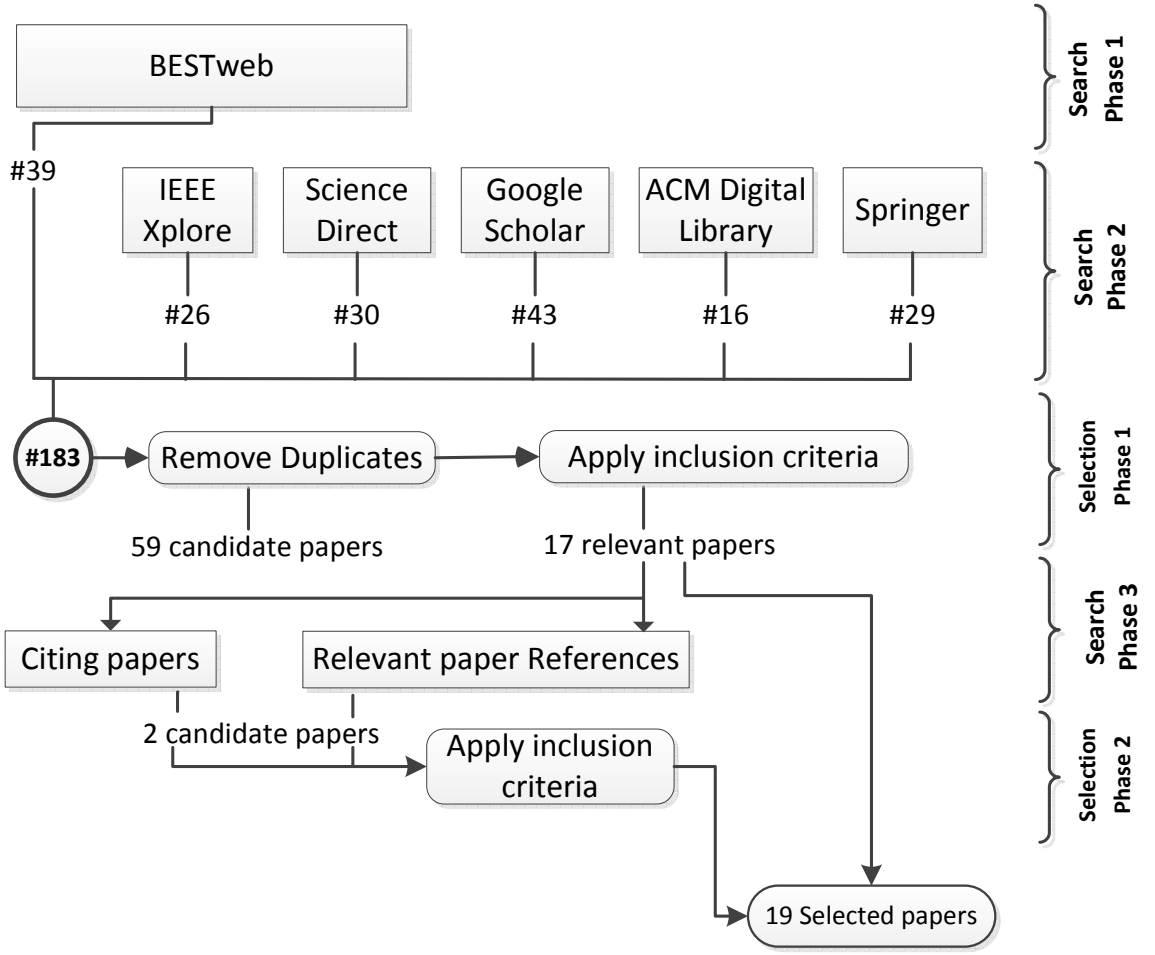


Figure 3.1: The filter approach for feature subset selection

based on quality criteria. i.e., searched only high quality venues.

A master search form was created (see Table 3.2) to manage and keep record of all the candidate papers. The first four items were essentially for bibliometric and housekeeping purposes. The remaining categories of information are cross-referenced to the four research questions. The search form was also used for data extraction. Exploiting the selected papers for data that contributed to addressing the research questions concerned in this review. The initial search and categorising was carried out by a different person from the one who checked reliability via independent checking of the included papers. Any differences in inclusion or categorisation were resolved through discussion.

Table 3.2: The search form for data extraction

Search form fields	RQ(s) addressed	Example article
Article identifier		[S07]
Year of publication		2007
Source		Conference
Article title		Optimising project feature weights for analogy-based software cost estimation using the mantel correlation
Weighting Technique	*RQ1*	Mantel's correlation
Technique framework	*RQ1*	Bias : Preset Weight Space : Continuous Representation : Transformed Generality : Global Knowledge : Poor
Limitations and strengths	*RQ2*	Removing some features before weighting, tested on single dataset and statistically based
Performance metrics	*RQ4*	MMRE, Pred(n)
Statistical testing?	*RQ4*	Yes
Datasets used	*RQ4*	Desharnais
Data quality/issues	*RQ4*	Missing values instances removed, suitability
Performance	*RQ3*	Only one dataset considered, dataset size, morphology maybe a factor

3.2.3 Data synthesis

The goal of data synthesis is to identify different FWTs from the selected studies in order to address the research questions. The extracted data obtained consisted of both quantitative data (e.g., values of prediction accuracy or results) and qualitative data (e.g., strengths and weaknesses of FWTs). Different strategies were used to synthesise the extracted data relating to different sets of research questions. The strategies are explained in detail as follows.

For the data relating to RQ1, the results were tabulated to represent the distribution of

FWTs. The FWTs were categorised based on the dimensions from a review by Wetterschereck et al. [170]. Their dimensions (see Table 3.3) are as follows:

- *Bias*: refers to whether the weight learning utilises feedback from the performance algorithm [69].
- *Weight space*: is used to distinguish feature weighting from conventional feature subset selection algorithms.
- The *Representation*: distinguishes algorithms that transform the given representation (to yield better results) from those that use the 'given' representation.
- *Generality*: is used to distinguish algorithms that assume weights differ among 'local' regions of the instance space form those that do not.
- *Knowledge*: highlights algorithms that use domain specific knowledge to determine feature weights.

Table 3.3: Dimensions For Distinguishing Feature Weighting Methods from [170]

Dimension	Possible Values
Bias	{Performance, Preset}
Weight Space	{Continuous, Binary}
Representation	{Given, Transformed}
Generality	{Global, Local}
Knowledge	{Poor, Intensive}

In RQ2, the limitations of existing techniques were identified from analysis of the text. For RQ3 differing response variables and the absence of detailed results (e.g., for prediction performance indicators typically only measures of centre are given rather than variance) make it difficult to conduct a formal meta-analysis. For our purposes we use a simple vote counting procedure although we recognise this can be problematic as it may bias the results to no effect [57]. For this reason we restrict our analysis to the question of "is there an effect as opposed to how large is the effect" and make no judgement concerning statistical significance when vote counting.

RQ4 considers the accuracy indicators utilised such as MMRE and R-squared, the data sets used for validation and the experimental design in terms of benchmarks and so forth. It turns out that all the primary studies were based on experiments although in theory other forms of empirical study were not excluded from the review.

3.3 Systematic Literature Review Findings

In this section we present and discuss the findings from our systematic review. We start by giving an overview of the selected studies. Then, we present a detailed description of the findings of this review for each research question. We also interpret the review results, not only within the context of the research questions, but also in a broader context related to the effort estimation.

19 papers were identified describing original primary studies and included in this SLR (see Table A1 in appendix A). These papers were published during the time period 2000 - 2014. A total of 14 (74%) papers were published in journals and 5 (26%) papers appeared in conference proceedings, however no papers were located from book chapters. The respective numbers, date published and relative sources of the selected studies are shown in Table Appendix A1. There is some evidence of the topic gaining momentum since six studies were published in the first half of this time period (2000-6) whereas a further 13 were published in the second half (2007-13), essentially doubling the rate of output.

In terms of the actual composition of the selected studies, observe that all the studies are experimental in nature. No case studies or other forms of empirical research were located. All studies, but for one, used a minimum of one project data set from industry to validate the feature weighting techniques. Finally I reiterate that since the quality of the selected studies was linked the inclusion criteria, where only papers from demonstrably refereed sources are included, I believe these should represent good experimental and research practice.

3.3.1 Feature Weighting Techniques

From the 19 selected studies, 12 distinct techniques for feature weighting were identified that have been applied to estimate software development effort to models using CBR. Other techniques are either variations on combinations on these 12 techniques. They are listed in Table 3.4:

Table 3.4: Range and diversity of FWTs

Technique	**Studies**	**#**
Genetic Algorithms (GA)	[S06], [S12],[S17]	3
Exhaustive Search	[S01], [S04],[S05]	3
Weighted Means	[S03], [S04]	2
Particle Swarm Optimisation	[S19],[S18]	2
Mantel's Correlation	[S07],[S10]	2
Principal Component Analysis	[S11],[S13]	2
Kernel Methods	[S15]	1
Heuristic Search (non-population)	[S02]	1
Fuzzy Logic	[S09]	1
Rough Sets Analysis (RSA)	[S08]	1
Grey Relational Analysis (GRA)	[S16]	1
Mutual Information	[S14]	1

It is clear that there is considerable diversity in approach with the majority of studies proposing and evaluating new techniques although the majority of techniques (13 out 19) adopt a general approach to feature weighting as opposed to a binary (included/excluded) view. Search heuristics are non-population searches such as hill climbing, therefore different from GA. We also note that some of the FWTs are combinations, obtained by combining two or more FWTs or by combining FWTs with non-FWTs. Overall we might characterise the area as one that is still in an early stage of development. Since we have a maximum of three observations (genetic algorithms) for any one FWT, our meta-analysis asks a more general question (RQ3). Namely do the techniques generally offer improvement over those not using a feature weighting regime?

3.3.2 Methods for Accuracy Estimation

Having considered the range, diversity, strengths and limitations of the feature weighting techniques I now turn to the central and most actionable aspect of the systematic review. Do FWTs perform better than conventional Analogy-Based Estimation (ABE) where all features have equal weights? In order to make this comparison studies that use conventional ABE as a benchmark are needed. Fortunately 17 out of 19 studies do this (i.e. not S01 and S10) consequently these 17 primary studies are used for meta-analysis (see Table 3.5).

Although the 17 studies all use a comparable benchmark each study has differences in their experimental design, choice of accuracy statistic and the level of reporting. Only counting results that are statistically significant is known to be problematic and indeed the probability of making the correct decision tends to zero as the number of primary study results becomes large (see Hedges and Olkin [57, pp48-52] for a detailed discussion). Mindful of the potential problems of this procedure, I followed the procedure recommended by the Cochrane Collaboration [59]: which recommends to ignore statistical significance and simply classify studies as supporting the intervention (using FWT), neutral (i.e., no difference) and negative (favouring constant feature weights).

Based on Table 3.5 observe that 16 out of 17 studies report a positive effect. As a formality one could use a one-sample sign test which rejects the null hypothesis of no effect ($p = 0.0003$). Thus despite reservations about the adopted meta-analysis procedure of vote counting there is clearly a strong result and one can be reasonably confident that FWTs have the effect of reducing prediction error for software effort when using CBR techniques.

The above analysis does not differentiate between differing classes of FWT. Examining the selected studies more closely one can see that the two most popular accuracy metrics are MMRE (Mean Magnitude of Relative Error) and Pred(25) (Percentage of predictions that are within 25% of the actual value). It can also be noted that the Desharnais data set is the most widely utilised. Therefore to make comparisons between the differing FWTs a subset of eight primary studies that utilise the same accuracy metrics on the same data set (see Table 3.6) are used. For some basic reference the original results reported in Shepperd and Schofield [150] are provided, although I would caution against over-interpretation of the

Table 3.5: Performance of FWTs against ABE benchmark

Study	FeatureWeightingTechnique	Statistical Testing	Bench marking	Improvement wrt EBA
[S01]	Exhaustive Search	No	No	n.a.
[S02]	Search Heuristics	Yes	Yes	Yes
[S03]	Inverse Rank Weighted Mean	Yes	Yes	Yes
[S04]	Exhaustive Dimension Weighting	No	Yes	Yes
[S05]	Exhaustive Search	No	Yes	Yes
[S06]	Genetic Algorithm	No	Yes	Yes
[S07]	Mantel's Correlation	Yes	Yes	Yes
[S08]	Rough Set Analysis	No	Yes	Yes
[S09]	Fuzzy Logic	No	Yes	Yes
[S10]	Mantel's Correlation	Yes	No	n.a.
[S11]	Principal Components Analysis (PCA)	Yes	Yes	Yes
[S12]	Genetic Algorithm	No	Yes	Yes
[S13]	PCA with Correlation Weighting	No	Yes	Yes
[S14]	Mutual Information	No	Yes	Yes
[S15]	Kernel Methods	Yes	Yes	No
[S16]	Grey Relational Analysis	No	Yes	Yes
[S17]	Genetic Algorithm	No	Yes	Yes
[S18]	Particle Swarm Optimisation	No	Yes	Yes
[S19]	Combinations	Yes	Yes	Yes

results. First differing procedures are used and the procedures for the exploration of the number of neighbours to use (k) also vary. As has been extensively discussed elsewhere, there is a lack of confidence in MMRE and Pred(25) as unbiased measures of prediction performance [89]. In addition, using multiple measures can yield contradictory results, so for example, the GAs in study S12 are ranked 3rd for MMRE and worst (9th) for Pred(25). This may be explained by the choice of objective function for the GA.

Notwithstanding the above reservations it would seem that the FWTs generally outperform the benchmark and unsurprisingly those based on exhaustive search tend to do best. The latter observation suggests that it will be fruitful to focus on metaheuristic search as a way of finding good approximations of the optima whenever computational considerations militate

Table 3.6: Performance of non-binary weight-space FTWs on Desharnais data set

Study	Technique	Criteria		
		MMRE(%)	Pred(25)%	k
[150]	Benchmark - no FWT	64	36	1-3
[S04]	Exhaustive Dimension Weighting	30	50	1-5
[S07]	Mantel's Correlation	34.5	49.5	1-5
[S12]	Genetic Algorithm	43	32	1-5
[S13]	Principal Components Analysis	46	51	1-5
[S17]	Genetic Algorithm	46	48	1-5
[S05]	Exhaustive Search	48.7	52.6	1-5
[S08]	Rough Set Analysis	59	42	1-4
[S11]	PCA with Correlation Weighting	64	36	1-5

against exhaustive search.

3.3.3 How was Feature Weighting Dealt With?

The findings and discussions on how feature weighting was dealt with is based on strengths and limitations of FWTs as evidenced on the framework dimensions [170]. The first three dimensions present inconclusive results while the last two dimensions present limitations. In general the greatest strength of FWTs is that they do not assume that features contribute equally to the output, therefore they assign different weights to the features. This potentially results in improved accuracy since theoretically the worst case is where all features do indeed merit equal weights which is a situation that should be discoverable by the FWT. The discussions on the first three dimensions are as follows:

- *Bias*: Most algorithms in the selected studies use performance bias methods therefore their search for feature weights is guided by the efficiency of the performance settings.
- *Weight space*: Thirteen of the 19 studies use continuous weight space and reported improved accuracy when compared with studies which used binary weight space. Therefore the use of continuous weight space by FWTs is a strength.

- *Representation*: All the algorithms in the selected studies transform the set of features used to represent the instances. Transforming the given representation before assigning weights assist to overcome insensitivity to correlated or interacting features. This may lead to improved accuracy [122].

While significant efforts have been invested in developing and improving FWTs, some limitations still exist which require research attention. These limitations are about the algorithms used, which in summary, can be described as follows:

- *Generality*: Despite the findings of the survey by Atkeson et al. [10] showing that assuming weights differ among 'local' regions of the instance space may improve results. FWTs in selected studies use algorithms do not assume weights differ among 'local' regions of the instance space i.e., they use a single set of weights, and employ it globally (i.e., over the entire instance space). Studies such as [4, 9, 45, 56] also reached the same conclusion as Atkeson et al. [10], therefore based on these findings it could be suggested that FWTs may benefit from assuming weights differ among 'local' regions of the instance space.
- *Knowledge*: Several researchers such as [9, 30, 132, 156] and [18] demonstrated the use of domain knowledge to assign weights and that it may lead to improved accuracy. Unfortunately *all* feature weighting techniques in selected studies do not use domain knowledge to specifically assign weights feature and this could be a limiting factor, since domain specific knowledge could assist in identifying the most important features and therefore assign them highest weights first.

3.4 Summary

This chapter presented a systematic literature review investigating how the empirical software engineering community has addressed feature weighting with a specific focus on analogy-based effort estimation. The literature search retrieved 19 papers, which appears a relatively low number considering that the papers only had to mention analogy-based/CBR estimation and feature weighting without actually proposing actions to tackle the issue, and also given the

large volume of papers published in the empirical software domain every year. Despite these potential limitations, I consider the most striking and actionable finding of the review to be that feature weighting techniques are consistently beneficial. Also FWTs may benefit from assuming that weights differ among *local* regions of the instance space. Regions that contribute more to the output may be assigned high weights and vice versa but intermediate regions are also considered. These different regions could be established through means such as: heuristic searches, statistical methods or genetic algorithms. Algorithms could also utilise domain knowledge to determine the different regions which is similar to expert judgement. The results show that feature weighting is an important aspect of CBR, however feature weighting is still under-explored as evidenced by the small number of articles proposing many techniques. There are many published feature weighting techniques which vary and are complex. Prior to this SLR, no up-to-date, comprehensive picture of the current state of FWTs in CBR existed, the closest being the review by Wettschereck et al. [170] also nearly 20 years old!, which was utilised to provide a framework for comparison. The next chapter presents a technique (blind analysis) that may help reduced the contradictory results observed in this SLR.

Chapter 4

Blind Analysis for Software Estimation Experiments

> It has been said: if you torture the data for long enough, in the end they will confess. ..... The data will always confess, and the confession will usually be wrong.
>
> *J.B. Copas, [119]*

> More fields should, like particle physics, adopt blind analysis to thwart bias
>
> *R. MacCoun and S Perlmutter*[a]*, [110]*
>
> [a]shared the 2011 Nobel Prize in Physics

In recent years there has been growing concern about conflicting experimental results in empirical software engineering. This lack of agreement is shown amongst the many empirical studies conducted in the various branches of empirical software engineering [145] including defect prediction [118, 175] and software effort estimation (SEE) [72, 149]. Closer investigation suggests that a contributory reason — though only one of many — is selective reporting and partial analysis [71, 146] known as analysis bias. One technique for reducing the propensity for bias is to conduct blind analysis [58]. Blind analysis entails, as a minimum, the labels of the different treatments being anonymised such that the researchers performing the analysis of the results do not know which result data (i.e., the response variables) relate to which treatment. This chapter explores the practicalities of blind analysis of experimental results to reduce bias and is based on my publication [154] — where Action research[1] was used as vehicle to explore blind analysis as a method to reduce researcher bias in software engineering experiments. Blind analysis can also help reduce analysis bias in Crowdsourced research studies such as [155]. The main purpose here is to find out how blind analysis can work rather than the detailed results of the experiment which are discussed in chapter 6.
Since bias is one of the major factors contributing to conflicting experimental results in empirical software engineering, the next section reviews what is known about bias in scientific research in general and empirical software engineering in particular.

4.1 Sources of Bias in Research

> "[L]et us define bias as the combination of various design, data, analysis, and presentation factors that tend to produce research findings when they should not be produced." John Ioannidis [67]

Researchers have been concerned about the potential impact of unintentional bias upon the part of scientists for at least the past three decades. In considering this it is important

[1] "Action research is a disciplined process of inquiry conducted *by* and *for* those taking the action. The primary reason for engaging in action research is to assist the "researcher" in improving or refining his or her actions." [138]

to distinguish between bias where there are systematic underlying reasons and processes leading to wrong research findings and general randomness. Since confidence limits and null hypothesis testing typically set thresholds at 95% this implies an acceptance of 5% of Type I errors, i.e., wrongly rejecting the null hypothesis or where the true population statistic lies outside the estimated and reported sample confidence limits. Conversely, depending upon the power of the study there is also the random possibility of failing to reject the null hypothesis when we should i.e., a Type II error.

There have been concerns that many areas of research ranging from medicine to social policy and experimental psychology to genomics have been impacted by different sources of bias. Delgado-Rodríguez and Llorca [35] have published a catalogue of more than 70 different types of scientific bias. Moreover these exclude those specifically related to data analysis, reporting and citation behaviours. At a generic level these include:

- *publication bias:* [38], which is the reduced likelihood of publishing certain types of study when the results are not perceived as 'interesting'. Generally results seen as not interesting are typically exemplified by the null hypothesis being retained. This may either be due to the peer review process (some results are seen more favourably by the referees than others) or the "file drawer problem" [135] (when researchers fail to complete or submit papers in a non-random way).

- *selection bias:* selective reporting in that the study only reports a subset of results [63]. Again this process can lead to the over-reporting of 'interesting' results and the under reporting of non-significant results or results with small or no effect size.

- *analysis bias:* where statistical procedures are selected according to their ability to yield 'interesting' results. In passing note that null hypothesis significance testing (NHST) is particularly vulnerable since the logic of this approach leads the researcher to an all or nothing situation, of significance or no significance. More than twenty years ago Dickersin observed how significant results are substantially over-represented in the field of medical research [38].

Unfortunately software engineering does not seem to be immune from these biases. A major

meta-analysis of 600 results derived from 42 primary studies of defect prediction algorithms found that the research team that conducted the work explained approximately 25 times more variance in the performance of the predictor as did the choice of algorithm [146]. Research group was also more important than the data set used to validate the predictor and considerably more so than the choice of metrics or inputs to the predictor. A recent study by Jørgensen et.al. [71] investigating incorrect results in software engineering experiments found that 67% of researchers had statistically tested and reported post hoc hypotheses, 69% only published the best outcome among several measures on the same test, and 55% of researchers had modified or developed outlier criteria after observing the impact of doing so on the results. Such biases also confound meta-analyses since the goal to uncover all relevant studies is thwarted by the systematic non-availability of certain types of result. Thus the entire research community is harmed along with our reduced ability to make reliable recommendations to practitioners.

Of course the question arises as to why scientists may exhibit bias. The first thing that needs to be absolutely clear about is that there is *no* suggestion that this bias is intentional or for morally questionable reasons. Possible explanations include the fact that expertise may not be evenly distributed, moreover some techniques are highly sophisticated and the parameter free-space extensive. As a result it is conceivable that a research group may be able to use Technique A more effectively than Technique B. Conversely a second group might behave in the opposite way. Another explanation is the majority of predictors exploit different machine learning techniques [168]. Such research generally proceeds experimentally and there is little theory to guide. Such research tends also to explore many variants of prediction systems often with many different parameter settings. The consequence is many results. This in itself is not necessarily problematic and there are various statistical procedures for adjusting significance thresholds accordingly. However what is less clear and therefore more difficult is the stopping criterion; at what stage should the researchers stop their experiments and report results? And a related problem is should all results be reported? There may be many intermediate results. These kind of problems mean that selective reporting can be difficult to address. The next section looks at blind analysis as a way to reduce researcher bias.

4.2 Blind Analysis

One approach to combat these biases is blinding the analysis [58]. Blind analysis can only help mitigate where statistical procedures are selected according to their ability to yield 'interesting' results i.e., analysis bias. The principal motivation of blind analysis in this thesis is to reduce researcher's analysis bias.

4.2.1 What is a Blind Analysis?

A blind analysis refers to an analysis in which the final result, and the individual data on which it is based, are kept hidden from the analyst until the analysis is essentially complete [55]. There are several different approaches to blind analysis, the method of choice depends on the type of analysis performed [55]. Blind analysis entails, as a minimum, the labels of the different treatments being anonymised such that the researchers performing the analysis of the results do not know which result data (i.e., the response variables) relate to which treatment. This renders "cherry picking" results more difficult. The idea is that by relabelling the different treatments e.g., as predictor 1, 2, ... , n then the researcher conducting the analysis of the results is no longer aware of which is the new 'pet' technique nor which are the results from benchmarks. Searching for a test or procedure that yields statistical significance is less straightforward since it is more difficult for the analyst to have a view as to what results are 'interesting'. Note that only the response variables are blinded, therefore context descriptors will be unchanged. I am unaware of this approach being used in software engineering prior to my work [154] but there are examples in other disciplines such as physics [8], whilst the concept is hardly used in the software engineering community. By the early 2000s, the technique had widespread use in the areas of particle and nuclear physics [90, 110]. Blind analysis has been used in the past by a number of particle physics experiments including: E791, KTeV, BaBar, and BELLE [55]. Note also that the technique may not be appropriate to other forms of empirical analysis such as case studies and focus groups.

4.2.2 Blinding strategies

Some forms of blinding are widely known: for example, not revealing to either patients or clinicians who receives a placebo or an experimental drug (also know as double-blinding), or hiding names and affiliations on scientific manuscripts to prevent reviewers from being influenced by authors' identities [110]. But these practices are only applicable to collection and source of data, not the analysis which is focus of this thesis.

Blinding strategies depend on the properties of the data, such as; statistical distribution, lower and upper bounds of values presented, whether the values are continuous or discrete, etc. Both data values and labels can be manipulated to develop a suitable strategy. Some blinding techniques used in the fields of biological, psychological and social sciences are shown in Table 4.1 [110] .

Table 4.1: Blinding strategies [110]

Strategy	**Perturbation**	**Potential application**
Noising	Adding a random number from an appropriate statistical distribution the to data points or model parameters	Testing which of several injury prevention training methods is most effective in reducing injury to Arsenal players.
Biasing	Adding a hidden value that is biased in a particular direction to obscure differences in experimental conditions.	Estimating whether the costs of a controversial health and safety regulation exceed its benefits.
Cell scrambling	Labels for experimental conditions are shuffled, so that it is unclear which set of results belong which conditions.	Testing a prediction that audio-books are better to understand than hard-copy books.
Item scrambling	Relabel (randomly) each data point to de-identify experimental conditions.	Analysing group differences that may be easy to recognize even in the presence of noise and bias (for example, effects of reaction time and intoxication).
Various combinations	Scrambling rows but keep pairs of variables together to preserve correlation. Variable blinding —swapping labels of various variables	

Strategies in Table 4.1 may be adopted for software engineering experiments. One researcher —or, more —methodically alters data values, data labels or both. The second researcher then conducts as much analysis as possible 'in the dark' [110]. Figure 4.1 outlines this process described in more detail in subsequent sections adopted for a software engineering experiment using blind analysis. The blinding was achieved as follows. Researcher 2 selected a data set. The application of the different prediction systems to the data set was performed by Researcher 1 who then sanitised the treatment labels. Next the results files were passed to Researcher 2 who performed the statistical analysis. Once this was complete Researcher 1 revealed the actual treatments. Clearly for blind analysis to be effective it requires a minimum of two researchers. Before unblinding, researchers should agree that they are confident enough of their analysis to publish whatever the result turns out to be, without further rounds of rethinking. Researchers are not prohibited from conducting extra analysis once data are unblinded, but doing so risks bias, therefore researchers should label any further analysis as 'post-blind' [110].

4.3 Blind Analysis Design and Discussion

The focus of this thesis is analogy-based estimation for SEE. Therefore the experiment and strategies are ABE for SEE, but this is a matter of convenience not necessity. This section gives a description of an experimental approach and the decision making involved using blinding when experimentally evaluating a new algorithm for feature weighting when using case-based reasoning to predict software project effort. The experimental method will be fully described later on in chapter 6 when blind analysis is included in empirical evaluation of the proposed technique. The description of incorporating blind analysis in software engineering experiment follows the steps numbered within Figure 4.1.

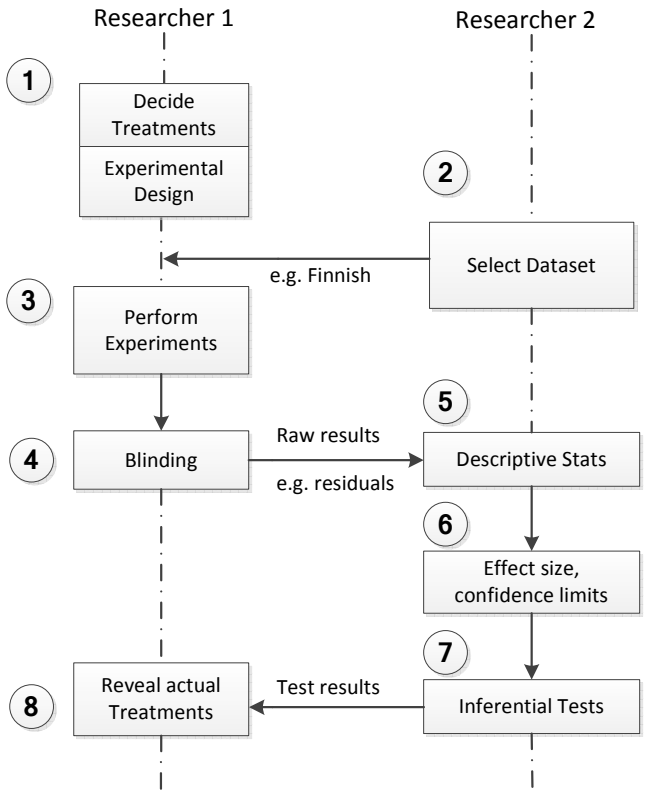


Figure 4.1: Experimental design and blind analysis process

Step 1: Researcher 1 determines different treatments or methods for software effort estimation using various analogy or case-based reasoning (CBR) methods. These methods could present a range of possible strategies for analogy based effort estimation. A trivial or naïve approach should be included (as a baseline) in order to determine the extent to which the more sophisticated techniques offer any value, in other words — we expect sophisticated techniques to be able to improve upon a baseline, and this is an important requirement for blind analysis as it provides a reference point for comparison.

Step 2: The choice of data set(s) is made independently by Researcher 2 without knowledge of the treatments. This is because it could be known that some data sets might particularly favour some SEE methods. The selection of data set was introduced since one may reasonably expect that some data set characteristics such as; distribution, size, presence of categorical features, noise, etc, could favour some techniques/treatments over others, and researcher 1

may be aware of this bias.

Step 3: The data set selected to be used in the study may require preprocessing before experiments can be performed on it. Preprocessing tasks such as; removing projects which have missing values so as to ensure none of the projects had missing values or removing any projects with values that are hard to interpret as meaningful, such as the number of people being negative or billions on a single project. The question on whether Researcher 1 should be permitted to remove any outliers is not easy to decide. Note a study [71] where 55% of researchers had modified outlier criteria after observing the impact of doing so on the results. This is one of the reasons why this blind analysis investigation adopted action research to enable researchers to improve or refine their actions. In this particular case no projects were removed — for one, software projects are commonly one offs [141], therefore removing projects as outliers does not help the investigation.

Step 4: All statistical analysis is based on results provided with anonymised treatment labels to Researcher 2. Note that if the experiments use a repeated measures design there may be no particular need to look at context variables or experimental moderators, but researcher 2 could still be interested in them therefore a discussion could take place between researchers 1 and 2. In other settings this might be relevant, however, in this particular it was only the treatments labels that need blinding consequently blind analysis did not inhibit richer or more sophisticated analysis when appropriate.

Step 5: There are a number of challenges relating to the statistical analysis of the experimental results. First, the distributions of the results must be established using descriptive statistics. The results maybe extremely skewed and not amenable to simple transformations. Second, there could be many ties. Third the data are dependent since one would compare the performance of different predictors on the *same* data. Finally alpha (α) may need correcting since multiple pairwise comparisons or tests could be needed.

Step 6: Next Researcher 2 may consider the questions of confidence limits for the descriptive statistics such as medians and then measures of effect size. Non-parametric methods could be required due to the non-normality of the distributions results. The Harrell-Davis percentile estimator [54] with bootstrap may be used as an efficient, robust technique to estimate the

95% confidence limits for the median (i.e., the 50th quantile) value of the results.

Step 7: The basic descriptive analysis from Steps 5 and 6 informs decisions on which inferential tests to perform. In practice there are a number of decisions to be made and no *a priori* reason to consider one superior to another in order to perform inferential tests.

- The level of *trimming* to apply since trimming provides a continuum of approaches from including all observations in estimating population characteristics to the other extreme of excluding all but the central point, i.e., the median.
- The choice between *Winsorized trimming* and trimming since Winsorizing involves the replacement of values with the trimmed minimum or maximum as opposed to discarding the values with trimming. The impact of such as choice is unclear.
- The type and direction of the *null hypothesis*, for example one could use one or two tailed tests.
- How to *correct alpha* since methods range from Bonferroni's correction which is a conservative method to methods such as Rom's method [171].
- The choice of *inferential test* to compare medians is again somewhat open even if we correctly restrict ourselves to robust methods since these include Cliff's, Brunner-Munzel and Wilcox's methods.
- Lastly, a small but subtle difference is median difference between treatments or comparison of the medians of the treatments

Finally researcher 2 reveals results to researcher 1 in the last step.

Step 8: Results in Tables 4.2 and 4.3 are used as an illustrative example obtained from my published blind analysis paper [154] comparing four treatments (FSW, FSS, CBR and Naïve) where the only relevant number is the highlighted cells used to illustrate the point of blind analysis. The predictors are compared pairwise (in this case six, since there are four treatments) starting with the greatest median difference. The probability of the median difference = 0 is given by p. The upper and lower bounds give the 95% confidence limits for the median difference therefore for a significant difference one would not expect the limits to straddle zero. The results are presented unblinded for the convenience of the reader.

The results of the analysis therefore show that whilst the technique FSW outperforms the

naïve and traditional CBR but there is no significant difference between FSW and FSS (highlighted cell). Thus one cannot argue that FSW is superior for this particular data set. Note that Researcher 2 might have easily and justifiably used trimming 10% or 20% (trimmed means) of each tail [172]. If a 20% trim is applied (see Table 4.3) then this yields a different set of results, that is: the same test that yielded $p = 0.954$ now yields $p = 0$; specifically, there is a significant difference between the absolute residuals from FSW and FSS such that FSW would be reported as being significantly superior.

Table 4.2: Pairwise comparison Wilcox's percentile bootstrap [154]

Test	**p**	**Lower bound**	**Upper bound**	**Median difference**
FSW v Naïve	~ 0	-2741.6	-2100.0	-2489.5
FSS v Naïve	~ 0	-2658.8	-1771.1	-2410.0
CBR v Naïve	~ 0	-2140.8	-1227.3	-1758.8
FSW v CBR	~ 0	-457.4	-146.0	-252.7
FSS v CBR	~ 0	-289.3	-58.9	-179.5
FSW v FSS	0.954	-0.5	0	0

Table 4.3: Pairwise comparison of mean absolute residual using Wilcox's percentile bootstrap (Trimmed means 0.2)

Test	**p**	**Lower bound**	**Upper bound**	**Median difference**
FSW v Naïve	~ 0	-2764.5	-2219.7	-2492.1
FSS v Naïve	~ 0	-2652.0	-2098.0	-2375.0
CBR v Naïve	~ 0	-2112.9	-1511.9	-1812.4
FSW v CBR	~ 0	-880.2	-479.3	-679.7
FSS v CBR	~ 0	-769.8	-355.5	-562.6
FSW v FSS	~ 0	-75.1	-59.0	-64.7

4.3.1 Blind Analysis Experiences

The decisions taken by Researcher 2, as previously mentioned, can lead to a different conclusion. For example, Table 4.3 shows that using an analysis based on 20% trimmed means results in $p \sim 0$ for the pairwise comparison of FSW v FSS (see the highlighted cell). This strongly contrasts with Table 4.2 where the same test yields $p = 0.954$. The consequence is that a 'result' may be transformed from insignificant to significant by changing the choice of inferential test. Thus in evaluating FSW v FSS Researcher2 could easily and *'correctly'* employ trimmed means to evaluate FSW v FSS. Trimmed mean looks to reduce the effects of outliers but in a less conservative fashion than analysis based on medians which in a sense is the most extreme form of trimming possible since only the central observation is retained [172]. The choice results in different conclusions for the evaluation of FSW v FSS.

Lessons from blind analysis experience

The basic principle of blind analysis was straightforward to implement. The analyst was only provided with residuals since actual predicted values could potentially jeopardise the blinding for techniques such as using a sample mean since all predicted values would be the same. One advantage of the relatively meaningless values was that the analyst (Researcher 2) could proceed in a somewhat detached fashion.

The point of blind analysis is not which is the most appropriate statistical approach to make comparisons between experiment treatments but that if the analyst has *a priori* expectations, and it's difficult not to, then these can influence the choice of technique and in a highly non-random fashion. Blind analysis does not prevent inappropriate analysis, it does, however, militate against systematic use of statistical methods in order to yield 'positive' results.

It is relatively easy to change the results of a statistical analysis without resorting to scientific misconduct. This is particularly the case for null hypothesis significance testing. For example moving to trimmed means (0.2) has the impact on the results transforming a not significant result (Table 4.2) in terms of evaluating a new algorithm into a significant one (Table 4.3). As a means of reducing systematic bias in terms of statistical and analysis decisions being made in order to achieve particular types of outcome I consider blind analysis has a great

deal to commend it.

It is not easy for researchers 1 and 2 to communicate without the risk of unblinding the data because the blinding was performed by researcher 1 and both were part of the experiment (action research). Decisions on statistical tests are taken without much guidance. I have just shown that you can get conflicting results without resorting to scientific misconduct, based simply on decision you made (such as whether to trim means or not). Based on these experiences and lessons learned I urge for a blind analysis protocol to help researchers perform blind analysis.

Threats to validity

This chapter described experiences for a single action research experiment. There is no control and $n = 1$.

All this demonstrates is that it is possible to manipulate results without recourse to poor practice or scientific misconduct and that it is straightforward to blind the analysis. Beyond this the argument rests upon advocacy. Nevertheless, I do argue that blind analysis should become normal practice within empirical software engineering when dealing with multiple treatments (and associated response variables) in some experimental or quasi-experimental setting.

I acknowledge that they are barriers to adopting any new concept or idea, therefore the next section looks at allay some of the fears or objections.

4.4 Overcoming barriers to blind analysis

Blind analysis is not a panacea, but it is very feasible. However, there are challenges and concerns about its adoption. This section looks at how some of the common objections could be addressed [110].

4.4.1 Concerns about blind analysis

How do we prevent people from peeking at the raw data? Blind analysis is not immune to fraudulent behaviour. But in ordinary research, teams of researchers can help to enforce compliance. Where blinding is part of the research culture, researchers often become its most effective custodians, for example, flagging the risk if a colleague asks for information (i.e., a plot) that might accidental unblind the result [110]. This was very evident when we performed blind analysis for [154] and subsequently for the work on this thesis. Everyone became conscious of information they gave out or asked for.

Can't we use other ways to avoid analysis bias? Preregistered analysis plans has been proposed as a potential solution. Preregistration requires that analysis plans are determined beforehand, and therefore offers some of the same benefits as blind analysis but limits the scope of analysis [110]. Because many analytical decisions cannot be anticipated, some of choices such as statistical tests are informed by the data (results) which you do not have at the start of experiments. This forces researchers to make some decisions knowing (consciously or unconsciously) how their choices affect the results. Blind analysis enables the analyst to engage in analysis and exploration without worrying about how his choices affect the results because they are informed by the data.

Isn't blind analysis too much hassle? Admittedly to do blind analysis there is extra effort involved. However, blind analysis could require as little effort as asking a colleague down the hall to alter labels for you.

The next section looks at challenges in adopting blind analysis.

4.4.2 Challenges for blind analysis

While the experimental experience shows that blinding can be a relatively straightforward procedure, there are challenges to its adoption:

Technical: learning what should be blinded while preserving features needed to permit appropriate analysis is not easy [110]. Blinding must also make sure it is not easy to identify

the benchmark treatment i.e., if the benchmark uses the sample mean for all prediction then just swapping the labels is not enough to blind the benchmark.

Communication during blind analysis process: Explaining to the analyst(s) the goal of analysis without unintentionally revealing the actual treatments or the benchmark is not easy. The analyst can pick up on non-verbal cues as to which treatment is which when discussing with the experimenter (researcher 1 in this case). Picking up non-verbal cues is not that difficult or unique to humans. Take for an example the story of Hans von Osten —often called "Clever Hans" —a horse that could *add* two single digit numbers [131]. Hans demonstrate his skill by pawing the ground with his hoof until the sum of the two numbers was reached, while those who had presented the numbers on black board watched. Remarkably Hans still managed to add the numbers correctly more often than not even with his trainer not in the room. The mystery of Hans's ability was solved in 1907 when the psychologist O. Pfungst proposed that the trial be conducted in which everyone in the room with Hans was blinded to at least one of the numbers presented [131]. Now with his observers blind to the answer, Hans was not able to produce a correct result. The conclusion was that Hans had been using subtle non-verbal cues from those observing him in the room —when deciding to stop pawing the ground —cues the observers were not even aware they were providing [90]. Therefore the lesson here is that the experimenter and the analyst must be blinded to the data. Previously I mentioned that for blind analysis to work at least two researchers are needed but now it looks like three would better. The third researcher would blind the raw results and pass them on to the analyst(s) leaving the experimenter and the analyst to communicate with little fear that some details may be un-blinded unintentionally. The details as to how the third researcher is included in the process described through Figure 4.1 is in chapter 6.

Motivation: The software engineering community must provide incentives for researchers to adopt a method that might make it harder for them to come up with desirable (although possibly false) results. This is important because before unblinding results, researchers must agree that they are confident enough of their analysis to publish whatever the result turns out to be.

4.5 Summary

Whilst there are minor challenges and some limits to the degree of blinding possible, blind analysis is a very practical and easy to implement method that supports more objective analysis of experimental results. Various statistical analysis decisions which ought *not* be guided by the hunt for statistical significance were highlighted and also showed that results can be inverted merely through a seemingly inconsequential statistical nicety (i.e., the degree of trimming). Apart from reducing analysis bias blind analysis has other benefits too — blind analysis can help researchers to consider the opposite of their expectations, fuelling both creativity and scrutiny of the theory and methodology [110] —considering the opposite is a corrective strategy for social judgement [109].

Blind analysis will bring software engineering in line with best practice from other branches of science. It is more a formalisation of good experimental practice than a new radical idea [110]. It is certainly no panacea, it needs to be stressed that blind analysis will not eliminate statistical errors and poor practice but what it does address is statistical procedures being systematically selected on the basis of them yielding desired results. If we can reduce the risk of analysis bias, why not do so? Therefore I argue that blind analysis should be the norm for analysing software engineering experiments.

The next chapter looks at feature weighting algorithm for ABE, therefore in order to improve the reliability of my results blind analysis will be used to analyse the results.

Chapter 5

Forward Sequential Weighting for Analogy Based effort Estimation

5.1 Introduction

The results of the systematic literature review presented in chapter 3 showed that feature weighting has been somewhat neglected in empirical software engineering. In particular, there is lack of studies on feature weighting algorithms distinguishing whether the weights apply should *globally* (i.e., over the entire instance space) or *locally* (i.e., vary in distinctive parts of the instance space ('local' regions)) [170] and empirical investigations into when to use a specific k (number of donors). The relationship between the quality of the prediction and the similarity distance d between the donor cases is not yet well understood. For example, it is not yet known to what extent a small d could be related to a smaller absolute residual. Depending on the nature of the relationship it may be possible to use it as a way to identify 'local' regions in a dataset, to improve effort estimation accuracy. Despite research findings of an old survey 1997 [10] and studies such as [4, 9, 10] and [56] reporting that assuming weights differ among 'local' regions of the instance space may improve results — software effort estimation studies continue to use a single weight set for entire data set — recall that from chapter 3 systematic literature review [153] that none of the selected studies employed differing weights sets a single data set, therefore did not assume weights differ among 'local' regions. Hence, based on these findings it is worth while to explore if Analogy-Based Estimation (ABE) would benefit from assuming weights differ among 'local' regions of the instance space. With this in mind, this chapter aims to answer the following questions:

- RQ1: What is the relationship between the absolute residuals and the similarity distance d between the donor cases?
  Rationale: The principle of ABE is that similar projects would have similar development effort, therefore we are expecting a negative relationship.

- RQ2: What is the relationship between the absolute residuals and the difference between the donor cases solutions (Y)?
  Rationale: The extent of how the variance between donors' solutions decreases prediction accuracy is unknown.

- RQ3: How can we exploit d between the donor cases to improve ABE?

Rationale: A weight set is obtained using all cases, despite the cases being independent, therefore it is possible that for some cases the weight set may not be the best option. Even though the cases are independent, the *'relationship'* obtained in RQ1 warrants d to be explored e.g., small d may require a different weight set from larger d, therefore d could be used to decide on which weight set to use.

Using answers obtained for the preceding research questions in this chapter a new technique to offer different weights set for different regions is described, and its predictive capabilities are evaluated in the following chapters. The technique extends ArchANGEL tool using a Feature Sequential Weighting (FSW) algorithm for analogy based effort estimation, representing one of the approaches for feature weight selection. This is effectively a supervised machine learning algorithm, and it is an uncomplicated greedy search based algorithm. It is more efficient to explore simpler algorithms first before investing in complex algorithm if a simpler algorithm proves to be just as effective. First, the formal problem description is presented, so that the mechanisms of FSW are clearly described.

5.2 Feature Weighting: Formal problem description

Let the software effort estimation data set be denoted as $\mathbf{D_{set}} = \langle \mathrm{X}, \mathrm{Y} \rangle$. X denotes a set of p case features $\langle \mathrm{X}_1, \ldots, \mathrm{X}_p \rangle$ and Y is the effort of the case. The i^{th} historical case is described as $\mathbf{c}_i = \langle \vec{x}_i, y_i \rangle$ where $\vec{x}_i = \langle \mathrm{x}_{i1}, \ldots, \mathrm{x}_{ip} \rangle$. Therefore x_{ij} represents the value of the j^{th} feature for the i^{th} case. The objective is to assign a weight to each feature value. Then the problem of effort estimation using feature weighting can be formally defined as:

Definition 5.1. *For a new case $\vec{c} = \langle x_{c,1}, \ldots, x_{c,p} \rangle$ with unknown effort y_c, determine $w_{c,j}$ (weight of each feature) so that the prediction accuracy, defined as $y_c - \hat{y}_c$, is maximized or its error is minimized.*

First, the rationale is introduced followed by an overview of forward sequential weighting, which provides the core operation of the approach. Subsequent sections discuss the application of, and limitations of FSW's method.

5.3 Rationale and the assumptions of ABE

This thesis explores the possibility that offering a different weight set to a target case depending on how similar its donors are to improve estimation accuracy. Hence, it is dependent on there being a relationship between donors' similarity and accuracy of estimate. Therefore before proceeding any further this relationship must be established. The visual representation of d between donor 1 and donor 2 is shown in Figure 5.1.

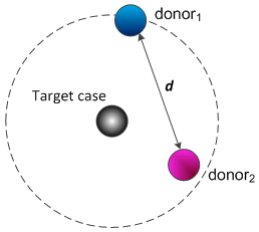


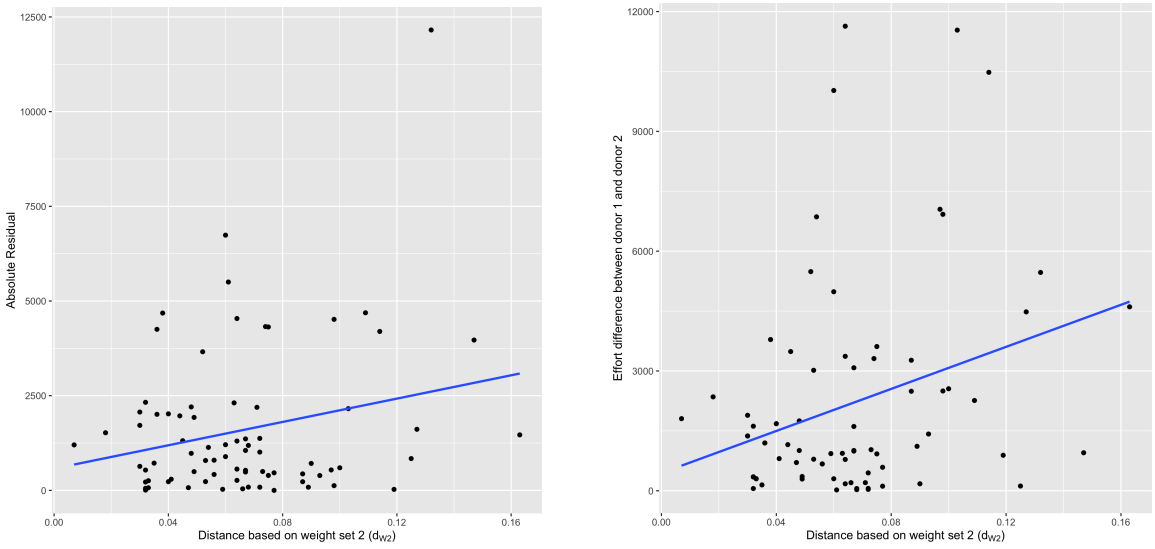
Figure 5.1: Donors distance (d)

The relationship between donors' similarity and accuracy of estimate must be based on information available or that could be determined when a target is presented for estimation. Information that is available to a target case includes the following items:

1. distance $d_{(pd)}$ between each case (potential donor) and the target case
2. distance d between each pair of cases (potential donors)
3. solution Y for each case (donor)
4. difference between each pair of cases' (potential donors) solutions $(Y_1 - Y_2)$
5. and any variation of the above points .i.e., weighted d

If any of the above information or combination is related to the absolute residual then it can be used to determine which weight set should be used for the target case in order to improve accuracy of the estimate. This in turn will provide answers to the first two research questions. The amount of information available to the target case is dependent on the number of donors used, for example distance $d_{(d3)}$ to the third donor is only available if $k = 3$ is used. Using a single donor ($k = 1$) severely limits the information available about the target case i.e., only distance $d_{(d1)}$ to the individual case is available. This does not provide any information about the structure/distribution or morphology of the case base in relation to the target case. Therefore for this reason this research will use at least 2 donors ($k = 2$) which provides more

information about the target case. Therefore ArchANGEL using weight set 2 (W_2) the effort of all cases in both datasets is estimated and recorded together with the distance d between the 2 donors for each case. This is used to calculate the absolute residual for each case. For each case (target case) in the data sets the similarity distance d between its donor cases was recorded as `donors_distance` and paired with its absolute residual ($|y_i - \hat{y_i}|$). This is used to answer RQ1, which is exploring the existence of a relationship between the absolute residuals and the similarity distance d between the donor cases. Recall ArchANGEL uses standardised Euclidean distance as a similarity measure. Figure 5.2 shows the scatter plot of d against the absolute residuals Figure 5.2 (a), and against difference between donor efforts 5.2 (b), with a linear trend line that suggests a positive correlation between the distance between donors (d) and both absolute residual and difference between donor efforts for Desharnais data set. This is an informal argument but the correlation shows some support for the idea that (d) may be related to residuals, while this shows the idea is worth pursuing, 5.2 also shows huge outliers / high influence points that negatively affect this relationship.



(a) Absolute residual vs d

(b) Effort difference vs d

Figure 5.2: Relationship between distance (d) and absolute residuals and efforts difference

Then the difference between the donor's solutions ($Y_1 - Y_2$) is determined and recorded as

`solution_distance`. This is used to answer RQ2, which is exploring the existence of a relationship between the absolute residuals and the difference between the donor's solutions. While Figure 5.2 shows positive correlation this is not enough to infer interaction effects. Next is to determine whether the main effects and interaction effect between d and prediction errors are statistically significant through analysis of variance (ANOVA).

A two-way analysis of variance which is an extension of the one-way ANOVA, is used to examine the influence of two different categorical independent variables on one continuous dependent variable, and also assessing if there is any interaction between the independent variables. I use a two-way ANOVA to understand whether there is an interaction between `solution_distance` ($d_{(d1,d2)}$) and `donors_distance` (d) on absolute residuals of effort estimates, where `solution_distance` and `donors_distance` are independent variables, and the absolute residuals are the dependent variable. Table 5.3 shows the results analysis of variance and eta squared for the Desharnais data set.

Table 5.3: Desharnais ANOVA results and Effect size eta squared

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	eta.sq
solution_distance	1	670685	670685	0.190	0.66389	0.002748968
donors_distance	1	36527835	36527835	10.369	0.00191	0.123830948
interaction	1	610673	610673	0.173	0.67838	0.002070207
Residuals	73	257172280	3522908			

Based on the p-values and a significance level of 0.05, for the results of Desharnais data set, based on Table 5.3 we can conclude the following:

- The p-value for `solution_distance` is 0.66389 and $R^2 = 0.23\%$. This evidence does not support a relationship/interaction between `solution_distance` and different absolute residuals, R^2 is also too low.
- The p-value for `donors_distance` (d) is 0.00191 and $R^2 = 12.3\%$, indicating that the levels of d are associated with different absolute residuals, although only 12.3% of variation in the response is explained by the model.
- The p-value for the interaction between `donors_distance` * `solution_distance` is 0.67838, indicating that the relationship between donors_distance and absolute residuals

does not depend on the value of `solution_distance`.

Because the interaction effect between `donors_distance` and `solution_distance` is not statistically significant, we can interpret the main effects without considering the interaction effect. Table 5.3 includes a model fitness for Desharnais dataset, showing about 12% of the total variance can accounted by `donors_distance`. So there are going to be limits as to what can be achieved however Cohen would consider an η^2 eta-square of 0.12 as tending to a large effect [32]. Results from one dataset may not be enough, especially as the data has an outlier therefore the same test was performed on the Finnish data set. Table 5.4 shows the results analysis of variance and eta squared for the Finnish data set.

Table 5.4: Finnish ANOVA results and Effect size eta squared

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	eta.sq
solution_distance	1	2.042e+08	204162078	17.604	3.35e-05	0.0306254034
donors_distance	1	1.281e+08	128052193	11.041	0.000973	0.0256931665
interaction	1	9.678e+05	967801	0.083	0.772829	0.0001941854
Residuals	401	4.651e+09	11597802			

Based on the p-values and a significance level of 0.05, for the results of Finnish data set based on Figure 5.4, we can conclude the following:

- The p-value for `solution_distance` is 3.35e-05 and $R^2 = 3.6\%$, this evidence does support a relationship/interaction between `solution_distance` and different absolute residuals.
- The p-value for `donors_distance` (d) is 0.000973 and $R^2 = 4.1\%$, indicating that the levels of d are associated with different absolute residuals.
- The p-value for the interaction between `donors_distance` * `solution_distance` is 0.772829, indicating that the relationship between donors_distance and absolute residuals does not depend on the value of `solution_distance`.

Again for the Finnish dataset the results show that `donors_distance` (d) can be associated with different absolute residuals, accounting for a total variance of about 3%. Also the interaction effect between `donors_distance` and `solution_distance` is not statistically significant, therefore for both datasets the main effects can be interpreted without consider-

ing the interaction effect. Now with the relationship between absolute residuals and donor distances established the next section describes feature-space partitioning approach.

5.4 A Feature-space Partitioning Approach

The question now is, how to identify these different 'local' regions when presented with a new case to predict? The concept of 'local' regions in this research refers to the region in which the distance between two donors would belong to, i.e. if region 1 is for distances between 0 and 0.5 then distance equalling or less than 0.5 would belong to region 1 — therefore use the weight set allocated to region 1. All these distances are with respect to the target case. The next section discuses a mechanism that can be used.

5.4.1 Feature Space Partitioning

Recall from previous section the relationship between d and absolute residuals was established. This is a negative correlation whereby the increase in d results in reduced estimation accuracy. The effect size on Desharnais is *large* (0.12) while it is *small* (0.025) on Finnish data set, therefore there are limits on what can be achieved based on this relationship. The question now is how to choose which weight set to use for each new case based on this relationship. The information available for each new case $\mathbf{c}$ is the distance d between its candidate donors and difference between their efforts. Recall also that from section 5.3 that for both data sets only d could be associated with the absolute residuals, note this association was not very strong. However d may be used in a meaningful way to identify different regions/partitions. This will be used as follows;

The weight set obtained using $k = 2$, is used as the base weight set to find d between all cases in the case base. The absolute residuals for each case obtained using different weight sets are paired with their corresponding d for the weight set generated with $k = 2$. — based on principle of Case-Based Reasoning (CBR) — one can assume that donors with a 'small' d will have similar solutions and thus they are providing enough information for the prediction, hence the target case will use the weight set computed with $k = 2$ that is W_k is W_2. On

the other hand, when the two donors are dissimilar (i.e., they have a 'large' d), they are unlikely to provide enough information to the target case's prediction and it will need to gain more information if we don't want to result in poor performance; therefore, the prediction of the target case will be done using the weight set computed with a higher number of donors ($k = 3$) say W_3. The use of W_3 changes the feature landscape, new donors are now available to the target case. Figure 5.5 shows the scatter plot of d for each case in the case base for the 2 different weight sets W_2 and W_3.

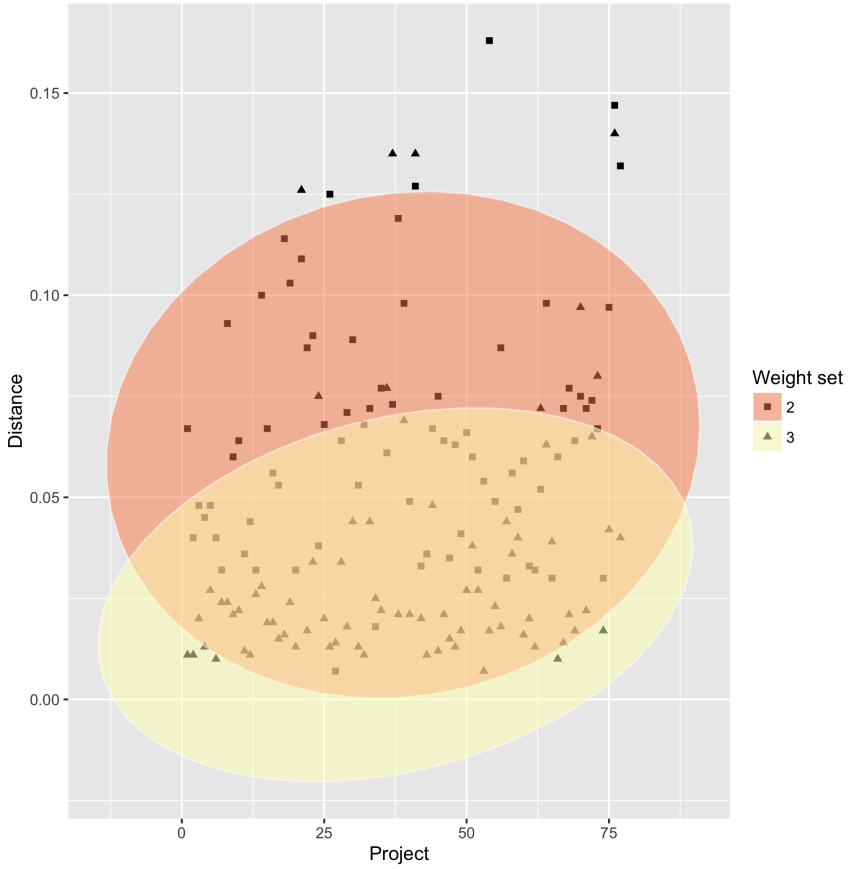


Figure 5.5: Scatter plot of d between first two donors for W_2 and W_3 for Desharnais data set

From Figure 5.5 one can clearly see that the distances between donors obtained using W_3 are smaller than the distances obtained using W_2; these have been clustered into two ellipses using the method `car::ellipse` detailed in [44] using the `R` package `stat_ellipse`. Looking at

the ellipses could lead one to be tempted to conclude that W_3 would provide better accuracy. Therefore, the absolute residuals for both weight sets W_2 and W_3 are plotted (see Figure 5.6) to see if and when W_3 performs better than W_2 in relation to the distance (d) between the donors.

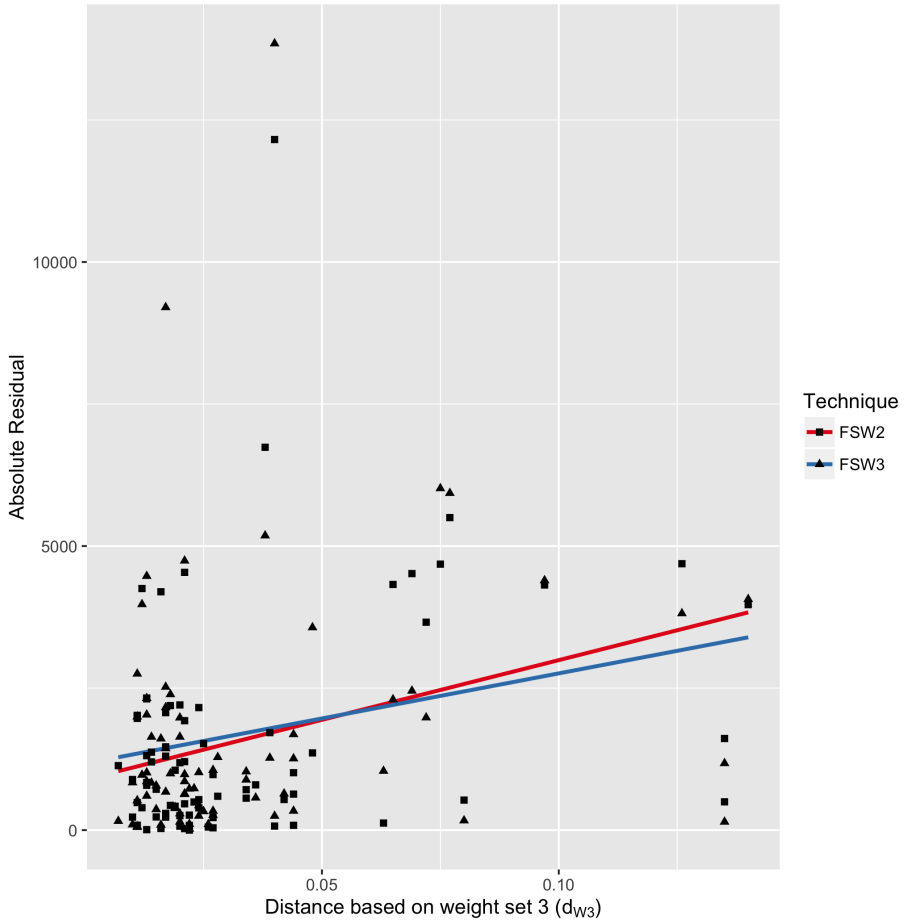


Figure 5.6: Comparing weight set 2 and 3 for Desharnais data set

Technique FSW2 means prediction using W_2 while FSW3 means using W_3, from Figure 5.6 one can see that FSW2 is 'better' than when the distance d between the donors is less than 0.05 there after it would be better to use FSW3. This supports the assumption that if the distance between the donors is *small* using two donors is enough to get good estimation accuracy and increasing the number of donors as the d increases.

One can easily generalize this strategy to whatever number of available weight sets by specifying the degree in which we will consider two donors to be dissimilar, using a more "information-dense" weight set (one generated using a further donor) when the distance increases. The question is given d of the new target case how do you decide between W_2 and W_3? That is, how to identify the 0.05 value for each dataset, since if the d for a target is greater than 0.05 we would use W_3 else we would use W_2.

A straightforward way to determine the ranges of distances d obtained for the target case for which to apply a "higher-order" weight set will be to use quantiles. An illustration for this is shown in Figure 5.7 for Desharnais data set for an example using only two weight sets. The question is when presented with a target case how do you choose the weight set 1 (W_1) or weight set 2 (W_2)?

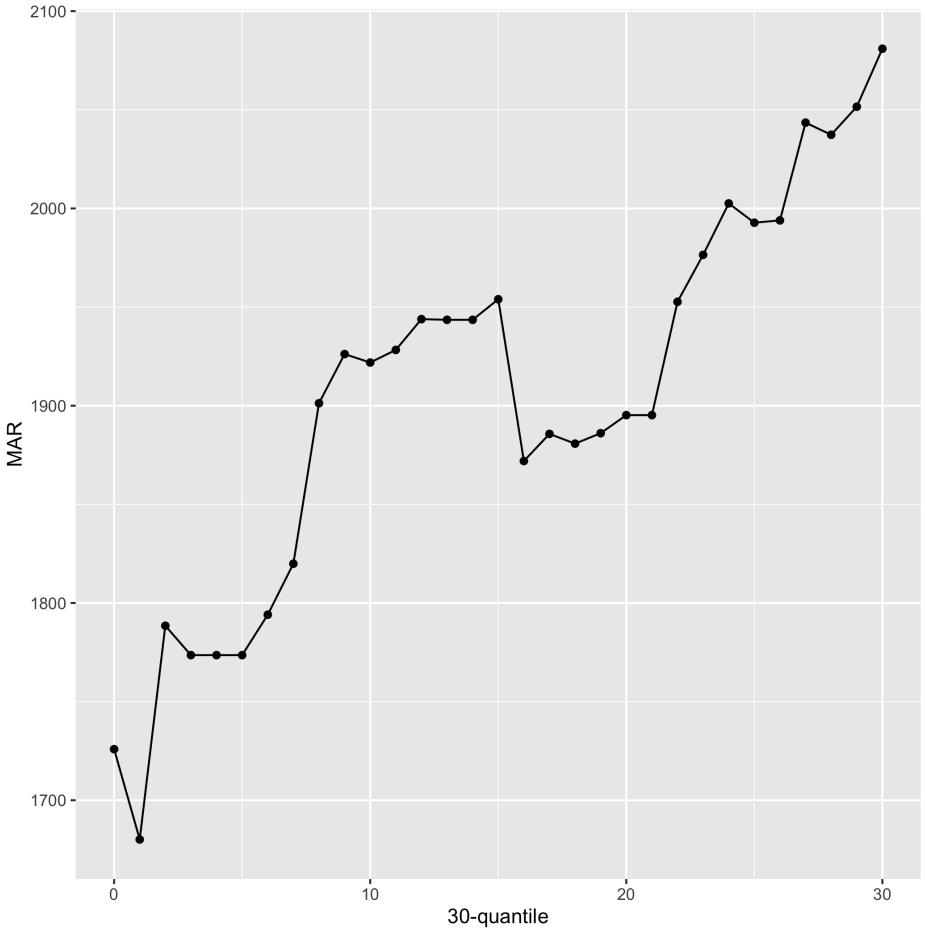


Figure 5.7: MAR's changes as a different distances' quantile of selection is chosen.

In Figure 5.7 the weight set selection strategy is shown: on the x axis I reported the distances' quantile — based on the weight set 1 — used to perform the selection within the dataset; I chose to employ quantiles rather then absolute distance values (as in figure 5.2) in order to consider the distribution of distances of each target case, which might be sensibly different if one employs a validation strategy based on moving windows (see chapter 6).

Given a sorted set $S = \{s_j : s_j \leqslant s_{j+1}\}$ (S is set of sorted quantiles), quantiles are cutpoints used to split the set in equal sized groups; one can split S in $q \in \mathbb{N}$ portions and compute

the values marking each range using the operator

$$\mathrm{P}(S, r, q) \coloneqq \mathrm{frac}\left(\frac{r(|S|+1)}{q}\right)\left(\mathrm{s}_{\lfloor \frac{r(|S|+1)}{q} \rfloor+1} - \mathrm{s}_{\lfloor \frac{r(|S|+1)}{q} \rfloor}\right) + \mathrm{s}_{\lfloor \frac{r(|S|+1)}{q} \rfloor},$$

which represents the value of the r-th q quantile of the set S, where frac$(\cdot)$ returns the decimal portion of its argument.

On the y axis I reported the Mean Absolute Residual for the whole predictions' set I obtained in relation to the chosen selection (on the x axis); this way we can easily observe how the choice of different selection strategies affect the performance of our algorithm. Selecting the first quantile (0) would result in always selecting the predictions obtained by the weight set 1 for all the cases in the dataset; moving on to the x axis by selecting a greater quantile e.g. 5 will result in using the predictions obtained by the weight set 2 only for those cases with a small distance between their first two donors which fall in quantile lower or equal than 5, and the ones by weight set 1 for the rest. Finally, selecting the maximum quantile (30 in this case) means that you are using the predictions obtained by weight set 2 for the whole dataset, since given any distance between first and second donor, this will always fall in a quantile lower or equal than 30 when $q = 30$.

The proposed selection strategy works by going through the case base and checking the distance between each available case's first two donors: the distances might belong (1) to the same dataset for all the cases (i.e. strategy a prediction model, e.g., ArchANGEL-X — see full description in section 5.6 — would use to do the prediction using the entire dataset) or (2) to different datasets (e.g., the validation strategy based on moving windows, see chapter 6); in the former case (1) one might choose a value (e.g., 0.005 in Figure 5.6) which will be used to select whether the predictions will be performed using weight set 2 — when the distance between the target case's first two donors is lower than that – or 3 — when it is greater than that. This is a sound strategy since the distances belong all to the same dataset and there is no need to normalize them; given a set of target cases whose distances between first and second donors belong to different sets (2) one needs to normalize the selection of the cutting point in order to be consistent across all the given cases and evaluate the performance of the proposed strategy for the whole available case base. This is done by using percentiles: by

selecting the weight set to be used based on the percentile the distance falls into in relation to the given distances' set provides a uniform strategy of selection which is independent from the chosen set.

In order to detail the proposed strategy, let $D = \{\mathbf{c}_i\}$ be our case base (donors) and $\text{donor}(\mathbf{c}, i) \coloneqq \mathbf{c}_i$ be the i-th donors of target case $\mathbf{c} \in D$, namely

$$\{\mathbf{c}_1, \ldots, \mathbf{c}_i, \ldots, \mathbf{c}_m \text{ where } \mathbf{c}_j \in D \setminus \{\mathbf{c}\} \wedge \mathrm{d}(\mathbf{c}_{j-1}, \mathbf{c}_j) \leqslant \mathrm{d}(\mathbf{c}_j, \mathbf{c}_{j+1}) \; \forall \; 1 < j < m\},$$

where $\mathrm{d}(\mathbf{c}_i, \mathbf{c}_j)$ measures the distance between case $\mathbf{c}_i$ and $\mathbf{c}_j$ using the weight set 1; we can define the set

$$\vec{\mathbf{d}}_2 = \{d_i : d_i = \mathrm{d}(\text{donor}(\mathbf{c}_i, 1), \text{donor}(\mathbf{c}_i, 2)) \wedge d_i \leqslant d_{i+1} \; \forall \; 0 \leqslant i < n\}$$

as the sorted set of the distances between their first and second donor using weight set 2.

Let $\bar{y}_c^1$ and $\bar{y}_c^2$ be the prediction for any target case $\mathbf{c}$ using respectively the weight set W_2 and W_3; our algorithm is outlined in the following procedure, that takes as arguments the case to predict c, the number of quantiles to use $q \in \mathbb{N}$, the case base D and the sorted set of the distances between each case's first and second donors, this is detailed in Algorithm 5.1:

Algorithm 5.1 Validation strategy

1: **function** FswVal(c, q, D, $\vec{\mathbf{d}}_2$)
2: **for** $0 \leqslant i \leqslant q$ **do**
3: $\bar{D}_i \leftarrow \left\{d \in D : \mathrm{d}(\text{donor}(d, 1), \text{donor}(d, 2)) \geqslant \mathrm{P}(\vec{\mathbf{d}}_2, i, q)\right\}$
4: **end for**
5: $r^* \leftarrow \underset{0 \leqslant i \leqslant q}{\text{argmin}} \left(\dfrac{\sum_{d \in \bar{D}_i} |y_d - \bar{y}_d^2| + \sum_{d \notin \bar{D}_i} |y_d - \bar{y}_d^1|}{|D|} \right)$
6: **if** $\mathrm{d}(\text{donor}(c, 1), \text{donor}(c, 2)) \leqslant \mathrm{P}(\vec{\mathbf{d}}_2, r^*, q)$ **then**
7: **return** $\bar{y}_c^2$
8: **else**
9: **return** $\bar{y}_c^1$
10: **end if**
11: **end function**

Function FswVal begins by computing the set $\bar{D}_i$ of cases whose distance between first and second donors falls into a quantile lower or equal than i, for each of the possible q quantiles ($0 \leqslant i \leqslant q$) for any given $q \in \mathbb{N}$; this means that $\bar{D}_0 \equiv \emptyset$ and $\bar{D}_q \equiv D$. Then it selects the best quantile r^* minimizing the Mean Absolute Residuals of all the cases, using the prediction obtained with the weight set 1 for those whose distance falls into a lower or equal quantile and 1 for the rest. Finally, it will predict case c using the weight set 2 if the distance between its first two donors falls into a quantile equal or lower than r^*, using the weight set 1 otherwise. Now that we have established how to select which weight set to use. The question now is how to create multiple weight sets (i.e., W_1 and W_2) in a meaningful manner i.e., the weights are not just an ensemble of sets from different algorithms used without any justification or related to the fundamental principle of CBR, which is the focus of this thesis. The next section describes ways of obtaining different weight sets from one weighting algorithm, exploiting the fundamental principles of ABE.

5.4.2 Weight Sets

Currently a single weight set is generated for the whole dataset as established in the literature review. This seems inadequate since the case base contains independent cases. Predicting individual efforts in a case base ideally would require a different weight set for each case, a weight set that fits each case's specific features. In other words, independent cases should demand a different weight set for their effort prediction. Since this is not feasible, one could split the case base into virtual partitions and apply a different weight set on each partition. How can one generate different weight sets for ABE that improve on the previous one? That is, non-random weight sets which exploit the fundamental principles of CBR. This weight sets should at least in theory result in improvement in terms of estimation accuracy. The next weight set must be offering some improvement over the previous. Using the principle of CBR "*similar cases have similar solutions*" hence if one donor (*k=1*) is sufficient to be used to produce a good prediction then it is used, and when *k=1* is not good enough to produce reasonably estimates two donors (*k=2*) are used, two donors provide more information that help improve estimation accuracy and so on. The other reason is that if the feature space

seems ill behaved / lots of variance then slowly moving to a more conservative strategy i.e., as k increases the prediction will tend towards using a sample mean (which could be improved by using one of the adaptation strategies discussed in chapter 2 such as weighted sample mean). This is also reflected in literature where researchers use different k values for different data sets [81].

Therefore, different weight sets can be generated by using a different number of donors progressively, thus resulting in multiple weight sets: for this study the first weight set W_2 is generated using two donors ($k = 2$) for each case, the second one W_3 is obtained using three donors ($k = 3$), and so on until W_5 ($k = 5$). Generally adding a new one will give no significant improvement, since on average it won't be similar enough to the target case to be of any benefit. The other reason is that, from the literature review reported in chapter 3 none of the selected studies used ($k > 5$)

The rationale for using multiple weight sets approach follows on the premises of ABE, where the use of a single specific weight set in a prediction will generate a poor estimate when the knowledge is limited. When the donors used to predict a target case are dissimilar (i.e. they are distant from each other, following definition of cases distance) limited useful knowledge. Therefore, a different weight set will be employed if it will provide more information where the current weight set underperforms: incrementing the number of donors will then provide more information in the prediction of a target case in the currently underperforming cases, since it had only a limited amount of information available (i.e., a smaller number of donors). Furthermore, many well-known datasets such as Desharnais heavily used in the literature show different accuracy performance using a different number of donors ($k = 1 \ldots 5$) [153], depending on their own morphology each data set is suited to a different k. Joining different datasets together to build a bigger case base (such as the International Software Benchmarking Standards Group (ISBSG)) would result in the need to use a different number of donors for each dataset's corresponding segment to obtain a good overall estimate. In the previous subsection the case of generating different weight sets for the whole dataset was made, each time using an increasing number of analogies. The question now is how to search for these weights to populate the weight set. The next section presents an algorithm used to generate

different weight sets varying the number of donors, implemented in ArchANGEL.

5.5 The feature weighting algorithm

The algorithm used in this study is essentially a greedy search algorithm with some modifications. This approach was chosen for three reasons. First, studies such as [85, 86] and [167] show that, it may not serve any purpose to use a complex search algorithm if the problem can be solved effectively using a simple algorithm (recall Ockham's razor) such as a greedy search algorithm suffice. Second, greedy search strategies seem to be computationally good and robust against overfitting [162]. Finally, a feature weighting algorithm presents the high end strategy for analogy based effort estimation strategies.

5.5.1 Supervised Feature weighting

The proposed algorithm uses the Feature Sequential Weighting (FSW) technique, employing supervised machine learning and the wrapper principles discussed in chapter 2. The weight search begins with an empty set of features weights and weights are progressively incorporated into a weight set. Every time a weight is assigned to a feature the accuracy of the predictions is evaluated. See Figure 5.8 which illustrates the work flow of a typical forward feature weighting algorithm used in ArchANGEL to evaluate possible combinations of input feature weights using jack-knifing, with Mean Absolute Residual (MAR) used as the evaluation criteria for its prediction accuracy, the system works in a greedy search fashion.
The Algorithm 5.2 works as follows: the objective is to find some vector such that each weight w_k satisfies $0 \leqslant w_k \leqslant 1$ and it yields a feature space that allows identification of better donors (minimizing Euclidean distance), given m cases and p features (see Figure 5.8 and Equation 5.1). This is an updated version of the scheme I proposed in [153].

$$\text{Similarity}(\mathbf{c}^t, \mathbf{c}^s) = \sum_{k=1}^{p} \sigma(\mathbf{c}_k^t, \mathbf{c}_k^s) \times w_k \tag{5.1}$$

where $\mathbf{c}^s$ is the source case, $\mathbf{c}^t$ is the target case, σ is the features similarity measure, p is the number of features and w_k is the k^{th} feature weight where $1 \leqslant k \leqslant m$. The feature weights

set: $w_k \in \{\frac{0}{i-1}, \frac{1}{i-1}, \frac{2}{i-1}, \ldots, \frac{i-1}{i-1}\}$, where i = number of values for w such that $i \geqslant 2$. This is to cater for the special case $i = 2$, is $\triangleq$ Feature Sequential Selection (FSS), whose feature's weights are: $w_k = 1 \vee 0$ i.e., a feature is either included or excluded.

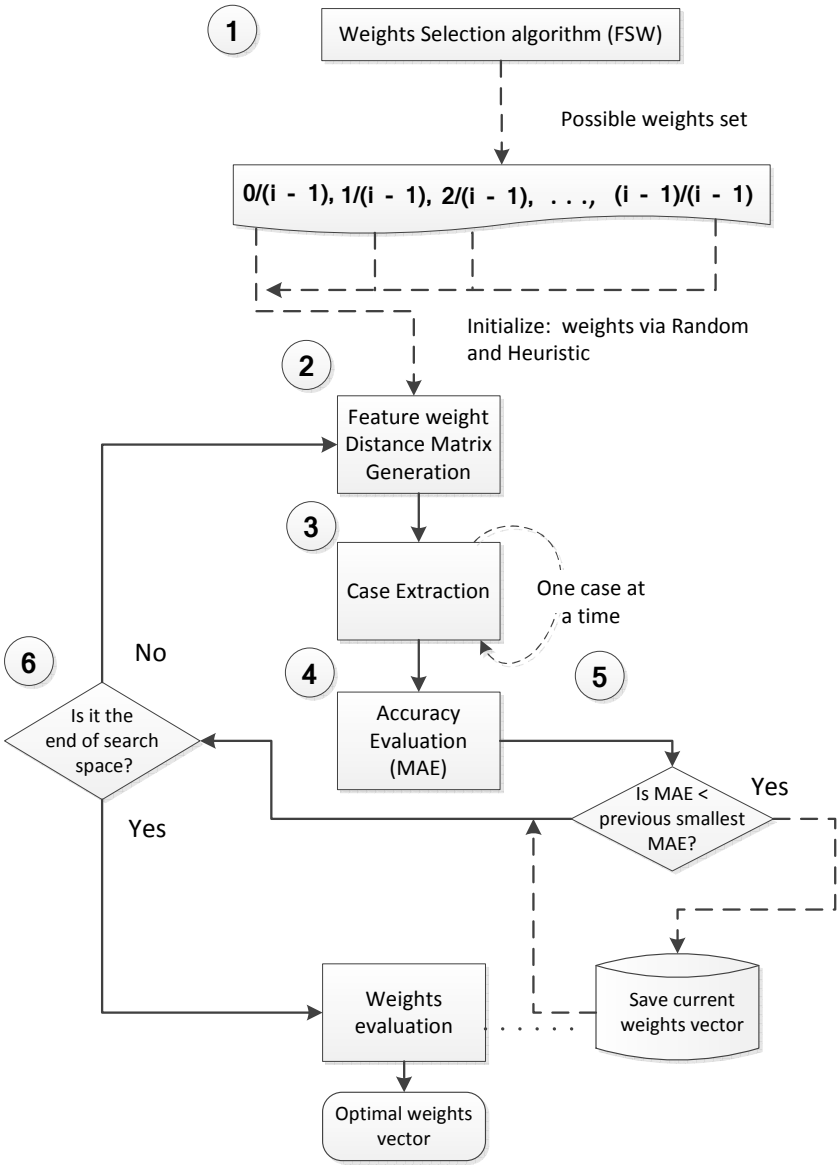


Figure 5.8: Feature weighting adapted from [152]

Algorithm 5.2 Feature Sequential Weighting (FSW)

Require: Historical effort estimation data sets

Ensure: feature weight vector

1: **Initialization :**
2: $W_s \leftarrow weightsSet$, $p \leftarrow \#features$, $n \leftarrow times$ ▷ All other feature's weights are set to 0
3: $w \leftarrow \{W_s[i], 0, ..., 0\}$ ▷ Starting with the first weight
4: **for** $i \in W_s$ **do**
5: **for** $j \leqslant p$ **do**
6: **if** $M(W_j[i]) < M(W)$ **then**
7: **if** $s \leqslant n$ **then**
8: $W \leftarrow \underset{w \in \{W_j[i], W_j[g(i)], W_j[g^{-1}(i)]\}}{\operatorname{argmin}} M(w)$
9: $s \leftarrow s + 1$
10: **else**
11: $W \leftarrow W_j[i]$
12: **end if**
13: **end if**
14: **end for**
15: **end for**
16: **return** W

Step 8 in the algorithm 5.2 is a modification introduced to reduce effects of local minima, this achieved in the form of feature re-weighting discussed in detail in the following subsection.

5.5.2 Feature re-weighting

Recall from chapter 2 that greedy search strategies are computationally good and robust against overfitting [162]. However, greedy search algorithms are also prone to converging to a local minima or maxima [50]. One approach to overcoming this is by re-weighting the features using function **g**. This function **g** could employ addition, subtraction, division and

multiplication to change the feature weight using predetermined constants or dynamic values.

$$W \leftarrow \underset{w \in \{W_j[i], W_j[g(i)], W_j[g^{-1}(i)]\}}{\operatorname{argmin}} M(w)$$

That is, instead of accepting the first weight that results in improvement one would check the accuracy when the weight is increased or decreased using function **g** i.e., the weight set W will be which ever is has the minimum error among $W_j[i]$, $W_j[g(i)]$ and $W_j[g^{-1}(i)]$. This not only increases the number of weights available (improving granularity) it may also reduce the epistasis effect. The next section looks at how this feature weighting strategy is implemented in ArchANGEL.

5.6 An eXtension for ArchANGEL

The proposed algorithm extends the basic ArchANGEL framework by including multiple weight sets and a mechanism that determines which weight set to use for each prediction. I call this method ArchANGEL-X, i.e., an eXtension for ArchANGEL. The ArchANGEL-X approach can be considered as a pre-process and functionality extension in ArchANGEL tool. The steps are as follows for any data set:

1. increment k by steps of 1 to generate the weight sets W_2, W_3, W_4 and W_5 (see previous section for details on why stop at $k = 5$) using the FSW algorithm 5.2.

2. Generate the sets of distances D_2, D_3 and D_4 using respectively W_2, W_3 and W_4 for each case in the case base along with its residuals with each weight set.

Algorithm 5.3 ArchANGEL-X prediction strategy using FSW

function ARCHANGEL-X(c, q, D, $\vec{\mathbf{d}}_2$)
  for $i \in \vec{\mathbf{d}}_2$ **do**
    $\bar{D}_i \leftarrow \{d \in D : \mathrm{d}(\mathrm{donor}(d,1), \mathrm{donor}(d,2)) \geqslant i\}$
  end for
  $d^* \leftarrow \underset{i \in \vec{\mathbf{d}}_2}{\operatorname{argmin}} \left(\frac{\sum_{d \in \bar{D}_i} |y_d - \bar{y}_d^2| + \sum_{d \notin \bar{D}_i} |y_d - \bar{y}_d^1|}{|D|} \right)$
  if $\mathrm{d}(\mathrm{donor}(c,1), \mathrm{donor}(c,2)) \leqslant d^*$ **then**
    return $\bar{y}_c^2$
  else
    return $\bar{y}_c^1$
  end if
end function

3. Generate the residuals of each case in the case base using W_5 (there is no need for D_5).

4. Use D_2 to determine where the weight set W_2 is most suitable by going through the case base D and checking the distance between target case's first two donors. If the distance between the target case's first two donors is lower than d^* — W_2 will be used to make the prediction. Otherwise use D_3 to determine where the weight set W_3 is most suitable for the remaining cases, and so on until W_5 will be the weight set used to make the predictions for the last set of remaining cases. The selection strategy works by going through the case base D and checking the distance between target case's first two donors. This is outlined in Algorithm 5.3:

Predicting effort for a target case is as follows. First determine its d using W_2, check which range this d belongs and use the appropriate weight set $W_?$ to predict the effort. In summary, ArchANGEL-X enables user to:

- Automate the entire feature weight assignment process
- Efficiently search and generate multiple weight sets

- employ multiple weight sets for single data set
- Identify the most appropriate weight set for the target case

ArchANGEL-X at the very least would provide the estimation equivalent to best accuracy of two weight sets when using a single weight set if using multiple weight sets does not improve estimation accuracy.

5.7 Summary

There is empirical evidence in the literature indicating that feature weighting improves estimation accuracy. This led to a number of software engineering researchers to consider the use of feature weighting in order to improve analogy-based estimation. This chapter described a novel approach to the estimation of software development effort by searching for multiple weight sets and identification of the most appropriate weight set for the current case. The next chapter describes how the ArchANGEL tool is used as part of an empirical analysis comparing the accuracy of FSW to FSS, CBR and Naïve approaches.

Chapter 6

Empirical Evaluation of FSW for Analogy Based effort Estimation

A variety of factors could be considered when assessing prediction systems, however arguably the most important and certainly visible factor is the relative accuracy of predictions obtained. Therefore any new estimation technique must justify itself first and foremost by the relative accuracy of its results.This chapter presents empirical results obtained using the ArchANGEL tool employing the Feature Sequential Weighting (FSW) strategy to predict project effort for 483 real life software development projects from 2 different industrial data sets.

6.1 Experimental Framework

Software effort estimation experiments involve three important points: (1) choice of feature weighting approach (estimation technique), (2) choice of performance metrics and (3) choice of data sets to be used. All these points are carefully considered based on the aims of the study. The framework presented in this chapter concentrates mainly on these three points for Software Effort Estimation (SEE). This is used at least partly to answer the research questions of this work. Throughout this study an Intel CORE™ i5 vPro™ PC was used to run R [134] and the ArchANGEL tool.

6.1.1 Choice of feature weighting approach

As explained in Chapter 5, one of the objectives is determining whether different weight sets for 'local' regions generally improve effort estimations given by a single weight set for the entire data set for analogy-based estimation. With that aim, three feature weighting available to Analogy-Based Estimation (ABE) were used. The three feature weighting approaches (as described in chapter 4) are:

1. *Case-Based Reasoning (CBR)* - it might be thought of as the baseline well-established technique and has been applied since the mid 1990s [150]. All features are equally weighted to predict the effort.

2. *Feature Sequential Selection (FSS)* - While CBR uses all features equally weighted, FSS' feature weights are either $0 \vee 1$. FSS excludes features that do not contribute (irrelevant

features) to the predicted value. It has generally been found to be an improvement over CBR [153].

3. *FSW* - it employs continuous, non-negative weights [153] to predict a new effort using a more efficient algorithm to search for individual feature weights.

These methods present a range of possible strategies for analogy based effort estimation. A trivial or naïve approach is included in order to determine the extent to which the more sophisticated techniques offer any value, in other words a baseline I expect to be able to improve upon.

In passing, note that in previous chapters the dangers of not using proper benchmarks were demonstrated and how researchers can be unaware that their methods in fact perform worse than guessing [149]. ABE using traditional CBR is the baseline technique and has been applied since the mid 1990s [150]. Subsequently FSS has generally been found to be an effective improvement over ABE; from my systematic review discussed in chapter 3 recall I have found that 16 out of 17 relevant primary studies reported positive results [153]. Finally FSW is a recent improvement to FSS [152] that uses a more efficient algorithm to search for individual feature weights as discussed in chapter 5.

For this experiment a leave-one-out cross-validation (LOOCV) procedure is employed [39]. Although computationally intensive for larger datasets when using a wrapper (because a new predictor has to be built for each case or project in the data set being held out) there is the advantage of the results being deterministic. By comparison, $m \times n$ fold cross validation will depend upon the random allocation of cases to the individual folds, thus there is often some variability in the results.

The next section looks at performance metrics used to assess the accuracy of FSW.

6.1.2 Choice of performance metrics

In order to assess the accuracy of cost estimation techniques, various performance metrics have been considered. Typically statistics such as Mean Magnitude of Relative Error (MMRE), Median of Magnitude Relative Error (MdMRE), PRED(p) and Standardised

Accuracy (SA) [149] have been used as the accuracy statistics for prediction systems. MMRE is one of the most widely used evaluation criteria for measuring the performance of competing software prediction models despite its flaws [153]. The basic metric in MMRE is Magnitude of Relative Error (MRE), defined as follows [34]:

$$\text{MRE} = \frac{|(y_i - \hat{y}_i)|}{y_i} \tag{6.1}$$

MMRE is given by:

$$\text{MMRE} = \frac{\sum_1^n \text{MRE}}{n} \tag{6.2}$$

where y_i is the i^{th} value being predicted, while $\hat{y}_i$ is its estimate, $y_i - \hat{y}_i$ represent the ith residual and finally the number of cases in data set D is represented by n. Small values for MMRE indicate low level of estimation error. Unfortunately MMRE has been shown to be a biased estimator of central tendency of the residuals of a prediction system because it is an asymmetric measure [43, 89, 126]. MMRE will be biased towards prediction systems that under-estimate, leading to over-optimism [75] since MMRE penalises overestimation more than underestimation. See Table 6.1 for an example where MRE leads to a conclusion that shows the prediction of project B being better than the prediction of project A, while in reality the absolute error of these two predictions is equal (80). The MRE values imply the prediction of project B is about 100 times better than the prediction of project A.

Table 6.1: Example to illustrate MMRE problem

	y_i	$\hat{y}_i$	Residual $(y_i - \hat{y}_i)$	Abs residual $\lvert y_i - \hat{y}_i \rvert$	MRE % $\frac{\lvert(y_i - \hat{y}_i)\rvert}{y_i} \times 10$
Project A	20	100	-80	80	800
Project B	100	20	80	80	80

The median of all the MREs are used to calculate MdMRE i.e. MdMRE = median(MRE). While MdMRE exhibits similar pattern to MMRE but it is more likely to select the true model especially in the underestimation cases since it is less sensitive to extreme outliers [43] The PRED(p) is the percentage of all predicted cost values that fall within p% of the actual

cost, and $p = 25\%$ is common used value. PRED(p) is defined as

$$\text{PRED}(p) = \frac{l}{n}, \tag{6.3}$$

where n represent the number of all projects and l denotes the number of projects whose MRE is less than or equal to p [104]. PRED(p) is not always suitable to compare competing software prediction models, because it does not make any distinction between the models that fall outside the $p\%$ range. For example if $p = 25\%$, a model whose majority of values lie at 26% of the actual cost would be ranked the same as a model whose values lie at 260% of the actual cost, since both of them are outside the set p.

Due to the aforementioned issues for MMRE, MdMRE and PRED(p), Shepperd and MacDonell proposed a new framework for evaluating prediction systems based upon an unbiased statistic i.e. SA, testing the result likelihood relative to the baseline technique of random 'predictions', i.e. guessing, and calculation of effect sizes [149]. Their evaluation is based on Mean Absolute Residual (MAR):

$$\frac{\sum_1^n |(y_i - \hat{y}_i)|}{n} \tag{6.4}$$

where y_i is the i^{th} value being predicted, while $\hat{y}_i$ is its estimate, $y_i - \hat{y}_i$ represent the ith residual and finally the number of cases in D represented by n. MAR unlike MMRE is not based on ratios therefore it is unbiased. The Standardised Accuracy measure proposed by Shepperd and McDonell [149], is given as:

$$\text{SA}_{P_i} = \left(1 - \frac{MAR_{P_i}}{\overline{MAR}_{P_0}}\right) \times 100 \tag{6.5}$$

The interpretation of SA is that the ratio represents how much better is the predicting system P_i than random guessing P_0, where $\overline{MAR}_{P_0}$ represent the mean of a large number of runs of random guessing. The proposed framework can be used in comparing competing predicting systems, providing a preferential order from binary preference relations such as $P_1 \prec P_2$ over the set P of candidate prediction systems. [149]. This preference relation may be interpreted as saying P_2 is preferred over P_1 or P_1 is less preferable than P_2.

In order to establish these preference relations, the following three fundamental questions must be addressed:

1. Is the performance of the proposed system better than random guessing?
2. Are the results statistically significant?
3. Is the effect size large enough to justify preferring one system over the other in practice?

After addressing these questions one is in a position to evaluate and validate any prediction method[152].

6.1.3 Choice of data sets

The analysis presented in this thesis is based on two software project effort data sets (see Table 6.2) namely Desharnais [37] and so-called 'Finnish dataset' used in [86]. These data sets were chosen because:

- *Accuracy metric used*: I preferred data sets used in studies that employed the Shepperd-MacDonell method [149] which uses Standardised Accuracy or at least studies that make the MAR available. This will enable comparison of results based on SA. Both these two datasets meet this condition. Recall that in the previous section I discussed issues related to the popular metrics such as MMRE, therefore not used in this thesis.

- *Widely used in feature subset selection for CBR*: Desharnais is one of the most widely used in order to estimate the software development effort. A review by Sigweni [153] on ABE models employing feature weighting, found that 15 out of the 19 selected studies used the Desharnais data set. The Finnish data set is also widely used in effort estimation studies e.g. [85, 86, 154].

- *Representative*: In order for the data sets to be representative it is helpful if they are of different sizes. Therefore one of the data set should be small — i.e. have a small number of cases and features — while the other should be large — therefore having a large number of cases and features. The Finnish data set would be representative of larger software engineering data sets (408 cases and 44 features), since software project effort data sets usually contain relatively few cases, typically less than 50 features and almost invariably under 500 cases [85]. Therefore the Finnish data set used in this

study is at the large end of this spectrum. Whilst the Desharnais data set with only 9 features and 77 cases would represent small to medium data sets.

Table 6.2: Selected data sets

Data set	No. of cases	No. of Features	Min effort	Max effort	Avg. effort	Std. dev. effort
Desharnais	81	9	546	23940	5046.31	4418.77
Finnish	408	44	0	63694	5006.3	7314.49

The next two subsections provide detailed descriptions and explanation on how these data sets were processed.

Desharnais data set

The original Desharnais data set consists of 81 software projects obtained from Canadian software houses over 3 different development environments, published by Desharnais [37]. Unfortunately, 4 projects contained missing values therefore excluded from the original 81 resulting in 77 projects used in this study. This data set is described by 9 attributes/features, one dependent attributes which is development effort measured in 'person-hours' (the feature to be predicted), and 8 independent attributes. ABE does not deal well with missing values therefore these 4 cases would lead to misleading estimation. The following section describes the structure of Desharnais data set.

- **Features:** Desharnais data set has nine features used to describe a project. Only one of the features (Language) is categorical while the rest are numerical. These features are fully described in Table 6.3. The actual effort in person-hours is the dependent variable.

Table 6.3: Desharnais data set features

Feature	Description
TeamExp	Team experience
ManagerExp	Manager's Experience
Length	Length of project
Transactions	Number of transactions
Entities	Number of entities
AdjustFctor	Sum of complexity factors
PointsAdjust	Number of adjusted function points
Language	Programming language
Effort	Development effort

- **Preprocessing:** Detection and removal of outliers is often necessary when building machine learning models but since my benchmarking studies did not remove them. I did not remove any outliers from my selected data set. The other reason is that it is not always a good idea to remove outliers as it will not always lead to improvement of performance for the model, or could artificially improve the performance through local optimisation but at the expense of generalisation.

- **Missing values:** Out of 81 projects in the Desharnais data set, four projects had missing values. Four projects represent is a small number (about 5% of the total number of projects). Since several previous studies (such as [120] and [86]) have removed these projects, I decided to remove them from the data set since it enables proper benchmarking.

Finnish data set

The Finnish data set is made up of project data collected by the software project management consultancy organisation STTF Ltd. The Finnish data set contains 407 cases described by

44 features.

- **Features:** The original Finnish data set had 90 features. The data set used in this study is the same data set used by [85], which removed some features due to missing values. The features are a mixture of categorical, continuous and discrete. These features are fully described in Table B1 in Appendix B. The actual effort (Worksup) is the dependent variable.

- **Preprocessing:** For the same reasons given for Desharnais data set I did not detect or remove any outliers on the Finnish data set.

- **Missing values:** Out of 407 projects in the Finnish data set, none had missing values, but two of them had the actual effort attribute being equal to zero, therefore I removed these two projects. Zero effort would affect my analysis, as I would not be able to calculate relative errors, besides two projects represent less than 1% of the total number of projects.

6.1.4 Method for Accuracy Estimation

A majority of software effort estimation studies use historical datasets to build and validate models for estimating software development effort. Almost all of them assign projects to training and testing sets without any consideration to the date in which the projects were completed [108]. Therefore it is likely that the training set used to build a model to estimate the effort for a given target case includes cases that have not even started at the time the prediction of the target case is commenced [108]. However, in a real life setting, only completed projects can be considered when coming up with an estimate, i.e., one cannot consider future projects. Thus there is an evident mismatch between normal industrial and research practice [108]. However this has not been completely ignored software engineering community: studies such as [6, 100, 107] considered chronological splitting their data into training and testing sets. The notion of a "moving window" is brought about as time passes, new projects enter the data set, and old projects are removed as the window moves past them (see Figure 6.1). In particular, if it is assumed that recent cases better reflect current

development tasks and practice, while past cases become less relevant over time, it might make sense to discard older cases [108].

Figure 6.1 shows the projects in the case base ordered using their completion date. Therefore C_1, denotes the earliest project (i.e., completed first). In terms of ABE, for example using window size 3, window 1 would be used to predict the effort for case C_4, and when C_4 is completed and its effort now established it is then added to the case base (window 2) and available to be used to predict the effort for case C_5, note C_1 is now considered less relevant therefore it is excluded as shown in window 2, this also maintains the window size of 3. The next section discusses how these windows can be defined.

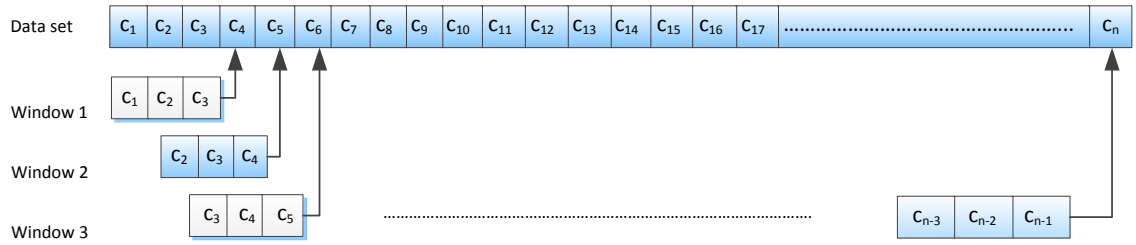


Figure 6.1: Evaluation data sets

6.1.5 Moving windows approach

A moving window might be defined in two ways, by the number of cases included per window, or by the period each window covers. With the former, only a fixed number (m) of the most recent cases are retained. With the latter, the size of the window is based on a fixed time span.

Each approach has potential benefits and limitations. A window based on a fixed number of cases could have a very small time span if several projects are completed within days of each other and conversely the time span could be very large if projects are completed very far apart say several years apart. If the time span is too large earlier cases that are perhaps irrelevant to current case would have to be included so that the window is the correct size i.e., the choosen m. On the other hand, a window based on a fixed number of projects can be scaled up or down to include enough cases needed for good analysis [107].

For a window based on time period, only projects within the defined period are included, therefore the number of cases could vary. One can only hope that there are enough projects to provide good estimate, or else there must be a trade-off between degree of recency against the number of cases included in the window. For a window based on time span can be scaled to include only those projects that reflect recent software development practices.
Ultimately though the choice of approach adopted may be informed by the data sets available: tables 6.5 and 6.4 show the distribution of projects completed per year for the Finnish and Deshairnais data sets used in this work.

Table 6.4: Number of projects completed per year for Finnish dataset

Dataset	**1978**	**-84**	**-85**	**-87**	**-88**	**-89**	**-90**	**-91**	**-92**	**-93**	**-94**	**-95**	**-96**	**-97**	**-98**	**-99**	**2000**	**-01**	**-02**
Finnish	1	1	1	3	13	15	31	41	48	33	22	6	4	23	58	67	34	4	3

The numbers of completed projects per year for Finnish data set range from 1 to 67, therefore if the windows are constructed based on this time period the windows will be very disproportionate. Although not to the same extreme, a similar effect can seen on the Desharnais (Table 6.5) data set.

Table 6.5: Number of projects completed each year for Desharnais dataset

Data set	**1983**	**1984**	**1985**	**1986**	**1987**	**1988**
Desharnais	3	5	22	28	14	5

It is for this reasons that a window based on time period would not be suitable for these data sets. Therefore the second approach is used in this study, where a window contains the N most recently completed projects. The next section discuses how N could be determined.

Window sizes

We need to establish the smallest amount of data that seems usable, for the smallest window size. Humphrey [62] suggested that windows of size 3 may be usable if they exhibit a reasonable correlation between size and effort i.e., ($R^2 \geqslant 0.5$). The other point to consider is the

number of training cases. Kitchenham et al. [88] suggested that at least 30 training cases must be used. The size of windows is also limited by the available data. Earlier on the set of completed projects is small, therefore one can only have smaller windows. However, as the data set becomes large one might consider windows of larger size. In order to see the effect of window size in addition to chronological arrangement of cases this study experimented with window sizes ranging from 3 cases to 45. This would cover the window size m recommended by [62] and the number of training cases suggested by [88].

Evaluation data sets

When evaluating the difference between multiple approaches (among windows of different sizes or between using windows against not using them), it makes no sense to include only cases available to all approaches. For example window sizes 3, 5, 10 can only be compared to window size 45 on the 46^{th} case onwards in the sorted case base. Therefore, as the window size increases, the available set of evaluation cases decreases. This places an upper bound on the window sizes that could be investigated. The absolute limit leaves only one case for evaluation. Table 6.6 shows the evaluation data sets. The cross window size comparison for Finnish data set can be done on 361 cases and for Desharnais on 32 cases, resulting in a loss of 45 cases from each data set.

Table 6.6: Number of sub data sets per window size for both data sets

data set	**size 3**	**size 5**	**size 10**	**size 15**	**size 30**	**size 45**
Finnish	403	401	396	391	376	361
Desharnais	74	72	67	62	47	32

Recall that the training set must consist of cases that were completed before the target case, in order to reflect the real life industrial practices. First, both the selected datasets were sorted chronologically by the projects' completion date. Whilst the Desharnais data set already contains this information (i.e. the feature `YearFin`), the Finnish data set required an additional step — namely adding the duration of the project in months (`duration`) to

the start date (`DATESTART`) of each project in the case base — in order to obtain a new feature named `EndDate` (used only for sorting, not as part of the feature space used for the prediction). Moreover, cases with identical end dates were again sorted using a random number assigned at the beginning in order to invalidate their original order.

I then built different sets of case bases using a moving window technique, see Figure 6.1 for an example. I extracted a continuous sequence of $m + 1$ cases (recall that m is the window size), starting from the oldest one in both datasets. Each sequence is exploited as one single case base with the oldest m cases as the knowledge base, and the additional one as the target case whose effort has to be predicted.

This way, $77 - m$ case bases were obtained for Desharnais and $406 - m$ case bases for Finnish each containing a single target case, thus requiring a total of $483 - 2m$ predictions for a single window size m.

For validation, $m \in \{3, 5, 10, 15, 30, 45\}$ were selected in order to compare the proposed technique with the main existing ones in relation to different sizes of the knowledge base: the lower bound 3 has been selected since proposed algorithm requires at least 3 donors to create two different weight sets in order to predict a target case (also suggested by Humphrey [62]), while 45 represents the upper bound since – as stated by Kitchenham [88] – at least 30 cases are required for a proper validation, thus Desharnais' size is a limitation.

This resulted in a total of 2328 sub-data sets for Finnish data set and 354 sub data sets for Desharnais data set, the breakdown of each window size can be seen in Table 6.6. The next section looks at how ArchANGEL was set-up to predict target cases for the evaluation.

6.1.6 Experimental process

This section first details the experimental process then outlines the blind analysis procedure adopted for this thesis. First ArchANGEL was setup and used converted the windows created in the previous section into the correct input format (XML). A total of 2682 of sub data sets were converted. ArchANGEL was setup for prediction using feature subset strategy as shown in Table 6.7:

Table 6.7: ArchANGEL setup

Field	**Option**
Target feature	*Actual Effort*
Adaptation strategy	*Inverse distance weighted*
Number of donors (k)	2
Holdout strategy	*Jack-Knifing*
Performance indicator	*Mean(|r|)*
Selection strategy	*CBR or FSS or FSW*

The effort of all target cases in all evaluation datasets were estimated using all the three different techniques (FSW, FSS and CBR) as the feature subset strategy and the mean of absolute residuals for these predictions was used as the accuracy indicator for each estimation technique. FSW was executed using different weight sets (W_k), specifically W_2 and W_3. The method to choose between W_2 or W_3 is based on percentiles as described on chapter 5. Since this study uses blind analysis the raw data (unextracted results) from ArchANGEL was passed onto a colleague to blind the results before being passed on to the analyst as per the blinding process described in the next section.

This work uses blind analysis in the experimental process as described in the blinding process from Chapter 4, with modifications to deal with issues encountered then. First I introduced a third researcher into the blinding process, see Figure 6.2 detailing the role of the third researcher. The introduction of the third researcher was brought on to deal with the communication issues encountered in the first experiment [154] between the experimenter and analyst. The third researcher (researcher 3) blinds and unblinds results after analysis of the data to reveal actual treatments, this enables researcher 1 and 2 to communicate freely without fear of accidentally unblinding the data. Also researcher 3 does not need to be closely involved in the work so it is not too hard to organise.

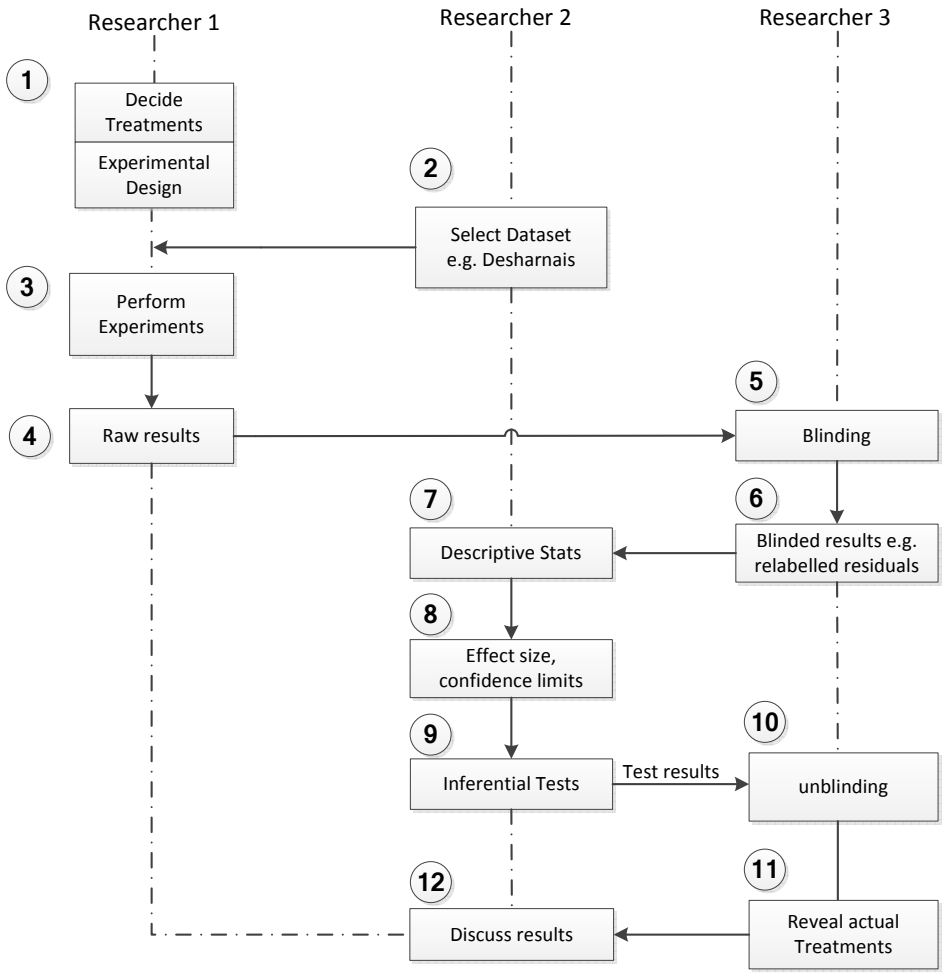


Figure 6.2: Blinding process

6.2 Empirical evaluation of Feature Sequential Weighting

The motivation for this work is to find ways to reduce analysis bias and at the same time investigate the use of multiple weight sets instead of one for ABE, as well as compare a traditional validation scheme against a scheme that reflects industry practice. Therefore the results are going to be reported in terms of traditional jackknifing validation and then pseudo time-series — with blind analysis wrapped around the whole process.

Recall from chapter 5 the proposed FSW should at least perform as the best of the two weight

sets (in case of using only two sets). This would provide an improvement over a single weight set, if the used weight set is not the best of the two. Therefore, the first point is to determine when to use a particular weight set. Recall from chapter 5 that I proposed an algorithm 5.1 that uses quantiles to select the weight set to be used for prediction based on the d of the target case's donors.

Figure 6.3 shows the plots for Desharnais data set. Based on these plots using quantiles the selection is as follows: for window sizes 3, 5, and 15 FSW uses weight set 3 only — for sizes 10 and 45 it uses weight set 2 only. For window size 30 FSW uses a combination of weight set 3 and weight set 2 split at percentile 5 i.e, any d that is falls in a percentile below 5 uses weight set 2 and the rest use weight set 3.

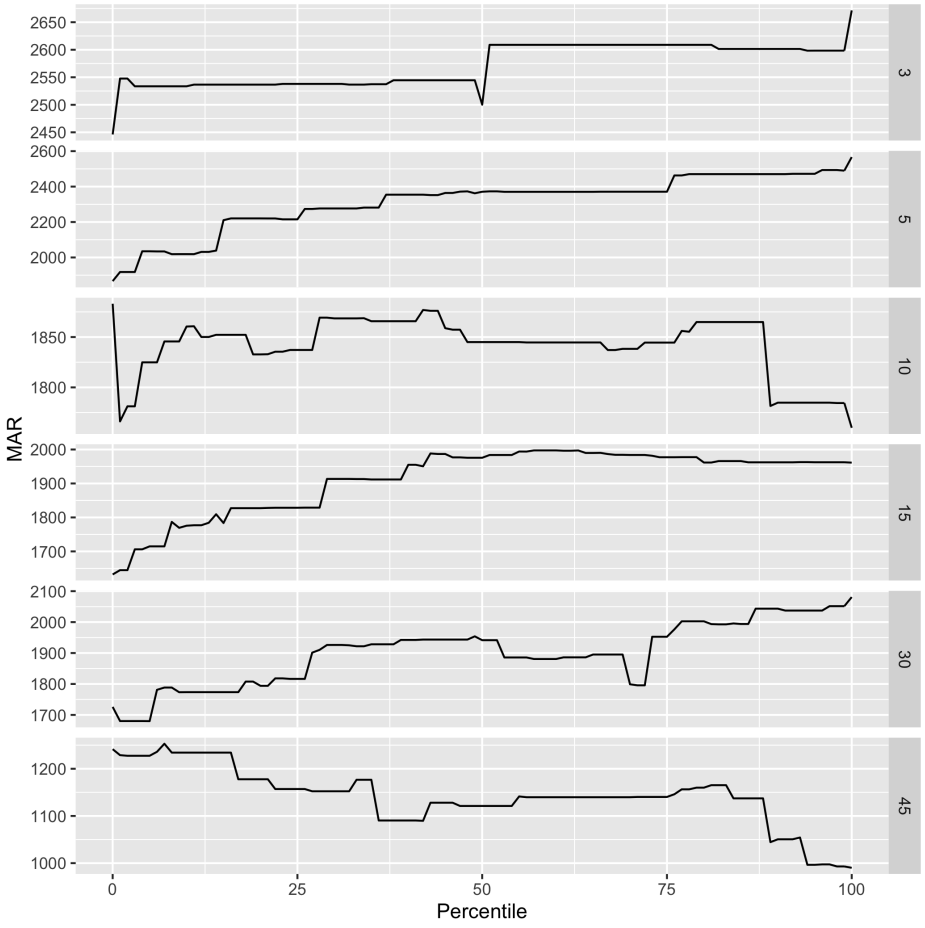


Figure 6.3: Desharnais percentiles against MAR

Quantiles were also used to select the appropriate weight set for Finnish data set. Figure 6.4 shows the percentile versus MAR plots for Finnish data set. Based on these plots the selection is as follows: for window sizes 3, 10, 15, and 30 FSW uses weight set 3 only — for window size 5 it uses a combination of weight set 3 and weight set 2 split at percentile 51 and for window size 45 uses a combination of weight set 3 and weight set 2 split at percentile 7.

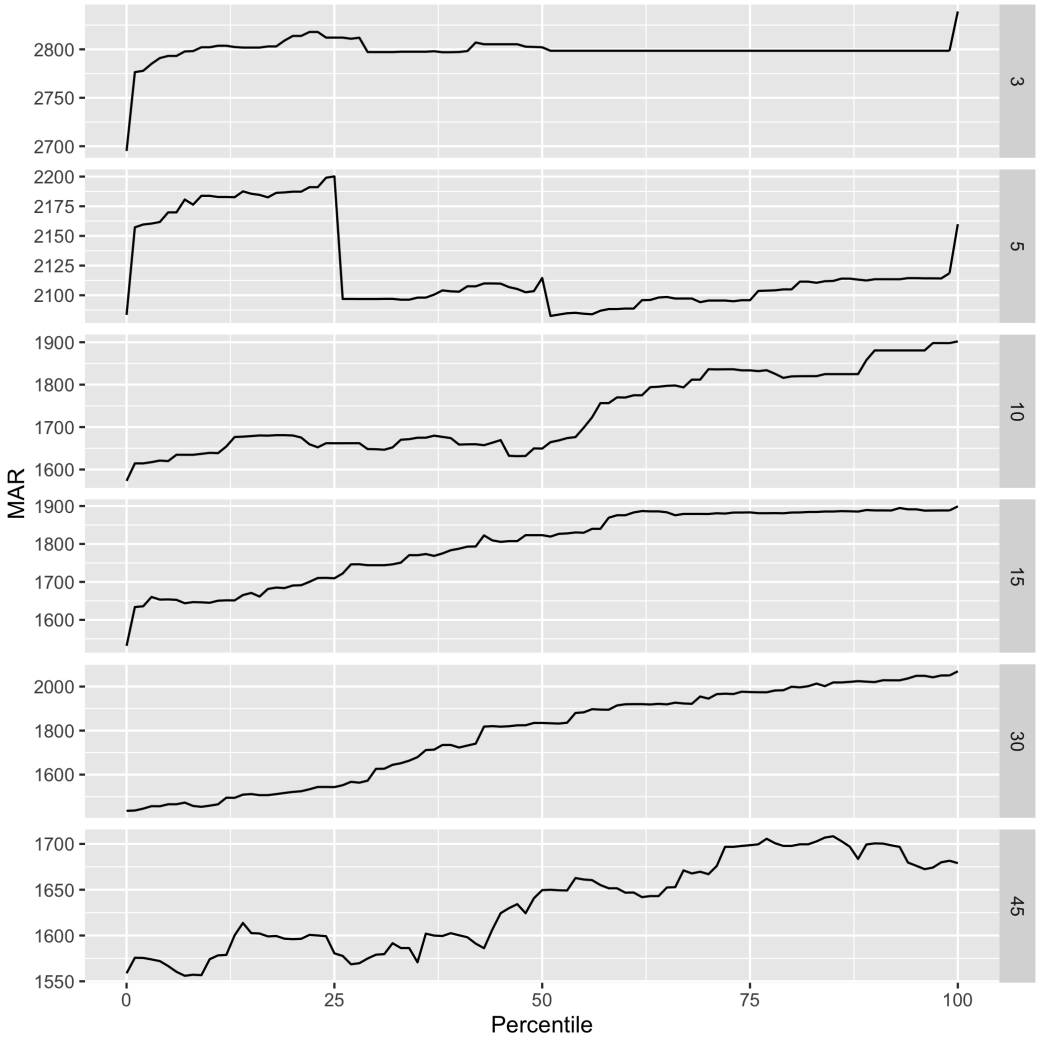


Figure 6.4: Finnish percentiles against MAR

Table 6.8 shows the performance of each weight set — e.g., FSW_2 is feature weighting using weight set 2 (generated using 2 analogies) and FSW is a combination of FSW_2 and FSW_3,

its composition is based on percentiles as described in the previous section.

Table 6.8: Performance of weight sets based on MAR for both data sets showing potential gain

window size	Desharnais (MAR)				Finnish (MAR)			
	FSW_2	FSW_3	FSW	$+\Delta$	FSW_2	FSW_3	FSW	$+\Delta$
0	1587.87	1698.76	1587.87	7%	1840.51	1953.29	1835.74	6%
3	2446.10	2671.48	2446.10	9%	2695.09	2839.08	2695.09	5%
5	1866.42	2566.64	1866.42	38%	2083.45	2159.93	2082.56	4%
10	1883.10	1759.86	1759.86	7%	1573.13	1902.01	1573.13	21%
15	1632.08	1961.34	1632.08	20%	1531.43	1899.14	1531.43	24%
30	1725.90	2080.95	1686.66	23%	1435.36	2069.39	1435.36	44%
45	1241.49	990.140	990.140	25%	1558.78	1679.00	1556.01	8%

The results in Table 6.8 show the use of multiple weight sets leads to some improvement even when only two weight sets are used. Overall the improvement ($+\Delta$) over either FSW_2 or FSW_3 ranges from as little as 4% to 44% with an average of 17% improvement over the different window sizes. FSW performs better than both FSW_2 and FSW_3 in four separate occasions as reported in the highlighted cells. The next section presents experimental results for comparison of FSW to FSS, CBR and Naïve techniques.

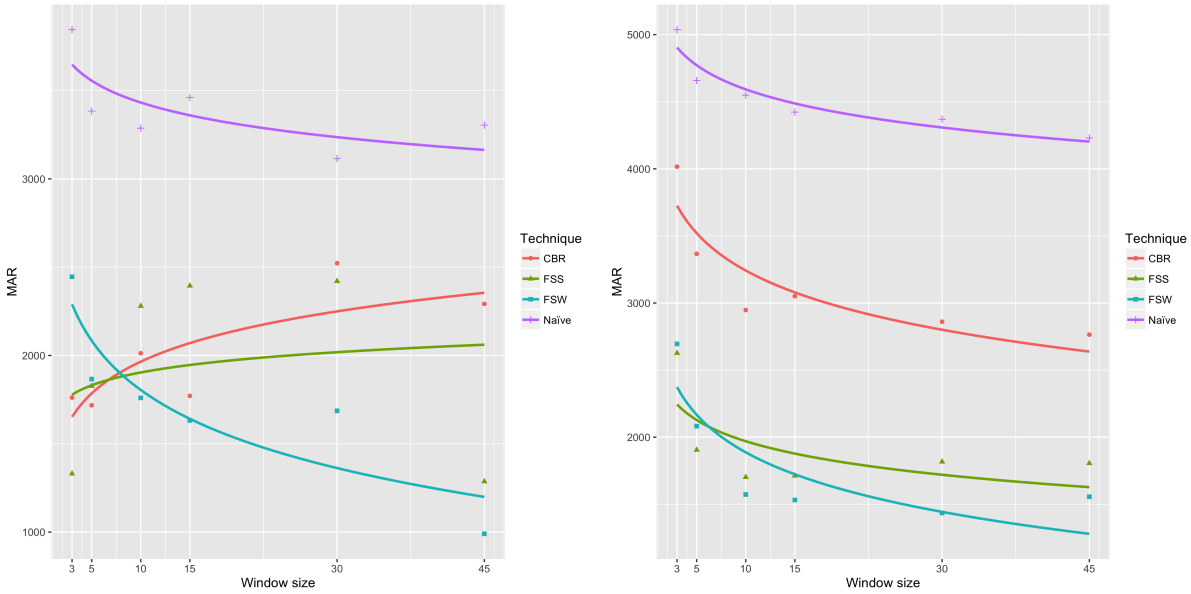
6.2.1 Experimental results for comparison of FSW to FSS, CBR and Naïve

FSW's performance is compared to FSS, CBR and Naïve. This enables one to make assessment to its predictive value. The results are reported in two parts: (1) Reporting the performance of each technique based on MAR observed from the tool, therefore obtained unblinded The only way to blind this output is to have a different person from the one who developed the technique run the experiments, this seems too much extra effort for no real benefit because at this stage one is only reporting observations not doing any analysis. (2) report inferential tests results obtained through blind analysis. Table 6.9 shows the performance based on MAR of FSW against FSS, CBR and Naïve techniques. The highlighted cells (bold numbers) indicate the best performing technique.

	window size						
Data set	$w = 3$	$w = 5$	$w = 10$	$w = 15$	$w = 30$	$w = 45$	*fullset*
Desharnais							
Naïve	3844.86	3383.93	3287.23	3460.88	3115.11	3304.49	3008.37
CBR	1761.53	**1717.94**	2012.70	1771.39	2522.49	2292.26	2241.03
FSS	**1330.79**	1826.82	2279.74	2394.62	2420.67	1286.45	1793.84
FSW	2446.10	1866.42	**1759.86**	**1632.08**	**1686.66**	**990.14**	**1587.87**
Finnish							
Naïve	5037.07	4658.79	4548.85	4422.99	4369.63	4229.82	4586.05
CBR	4016.22	3366.46	2947.61	3050.68	2860.41	2764.12	2548.05
FSS	**2626.49**	**1904.53**	1701.81	1712.11	1816.53	1805.14	1950.54
FSW	2695.09	2082.56	**1573.13**	**1531.43**	**1435.36**	**1556.01**	**1835.74**

Table 6.9: Performance of each technique for both data sets based on MAR

Figures 6.5 (a) and 6.5 (b) show the trends of the techniques using a logarithmic smoother. As the window sizes increase the performance of all techniques improves bar CBR on Desharnais data set. Note that as the window size increases the magnitude of increment reduces. This could be due to addition of outdated or irrelevant projects into the search space.



(a) Desharnais data set trends

(b) Finnish data set trends

Figure 6.5: Performance trends of techniques over window size per data set based on MAR

Overall results (see Table 6.9) based on MAR show that FSW performs better that FSS, CBR and Naïve for both data sets. However, interestingly for small window sizes (3 and 5) FSW does not perform better than FSS and CBR. For window size 3 FSS is the best for both data sets, while for window size 5 in Desharnais data set CBR performs better and this suggests that for small data sets FSS could be the better option. These are only preliminary observations. Inferential tests results will show if any of these 'superior' performances are statistical significant and I also have to asses effect sizes. These tests are performed using blind analysis as part of the motivation of this thesis — to produce credible/consistent results. In an ideal outcome as a developer of a new technique one would prefer the results that show that for all instances FSW is the best performer to have statistical significance, and for window sizes 3 and 5 to have no statistical significance. These results would therefore show that the new technique is better that existing ones. This a natural tendency, if one takes into account Maslow's hierarchy of needs for a human being with respect to a researcher — esteem and self-actualization [114].

One can achieve the desired outcome by choosing only statistical tests that reinforce superiority of FSW. This could be due to lack of guidance on statistical decisions, not necessarily deliberate scientific misconduct. These decisions include:

- The level of *trimming* to apply.
- The choice between *Winsorized trimming* and trimming.
- The type and direction of the *null hypothesis*.
- Method to *correct alpha*.
- The choice of *inferential test*.

One could also report only statistical significant results to support the advancement of FSW, withholding other results and this is known as selective reporting.

6.2.2 Blind Analysis Experimental results

The inferential tests results are reported following the steps in the blinding process shown in Figure 6.2, starting at step number 7. First I discussed the Finnish data set results then Desharnais.

Finnish results analysis

Descriptive Stats: All statistical analysis of the results is based on absolute residuals. These were provided with anonymised treatment labels to Researcher 2 by researcher 3. The four treatments are labelled PS1, . . . , PS4. Note that for this experiment uses a repeated measures design and there was no particular need to look at context variables or experimental moderators. It is only the treatments' labels that needed blinding consequently blind analysis did not inhibit richer or more sophisticated analysis when appropriate.

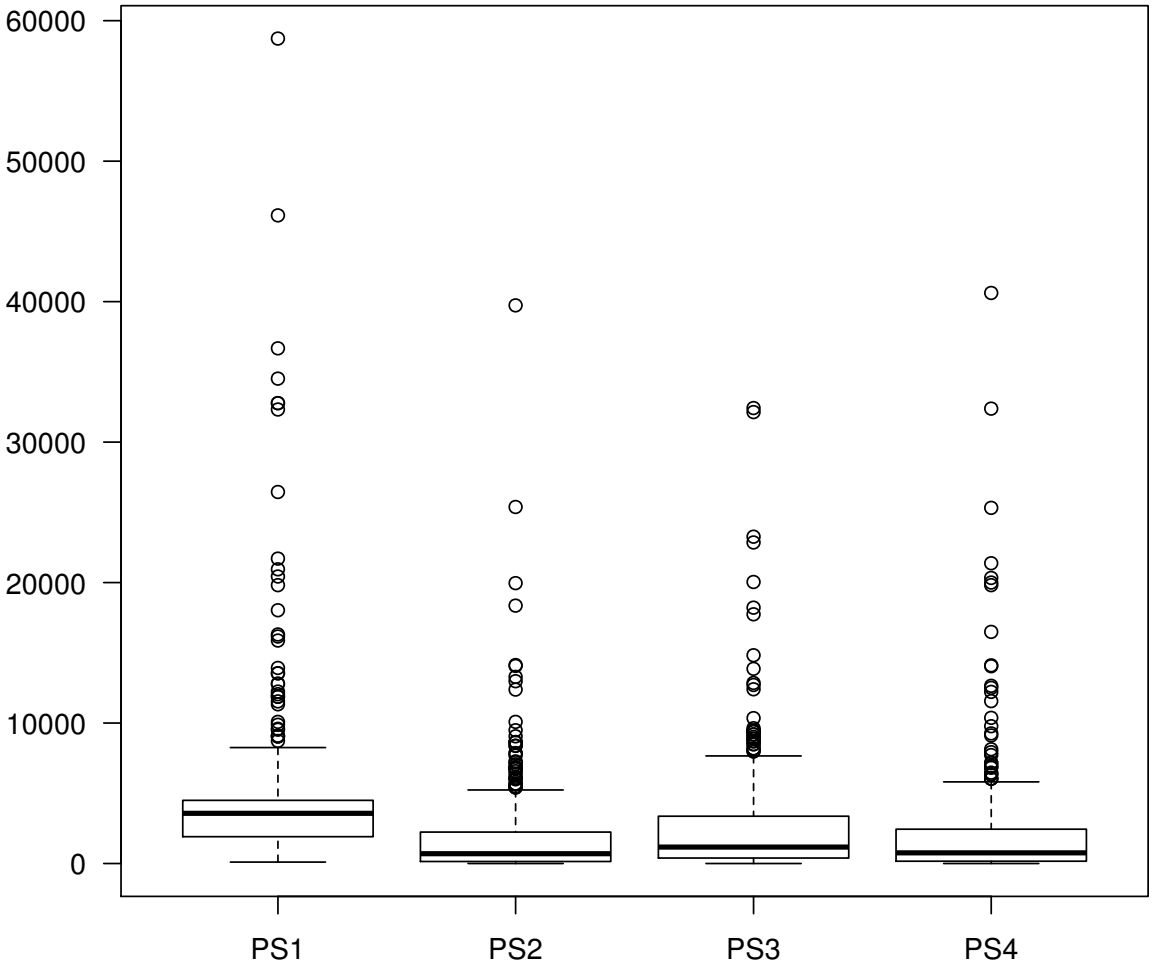


Figure 6.6: Boxplots showing residual distributions for Finnish Data Set

Descriptive stats, show that there are a number of challenges relating to the statistical analysis

of the experimental results. First, the distributions of the residuals are extremely skewed and not amenable to simple transformations (see the box plots of the absolute residuals for the prediction systems PS1, . . . , PS4. in Figure 6.6). Second, there are many ties (depending on the particular pairwise comparison, this ranges between 114 and 168 out of 406 cases). Third, the data are dependent since we are comparing the performance of four different predictors on the *same* data. Finally, alpha needs correcting since multiple pairwise comparisons or tests are needed (in our case six, since there are four treatments).

Table 6.10: Comparing absolute residuals by prediction system

Technique	**Mean abs residual**	**Median abs residual**
PS2 (FSW)	1835.7	725.4
PS4 (FSS)	1950.5	788.8
PS3 (CBR)	2548.1	1132.1
PS1 (Naïve)	4586.1	3675.0

Effect size, confidence limits: Next, Researcher 2 considered the questions of confidence limits for the descriptive statistics such as medians and then measures of effect size. Non-parametric methods are required due to the non-normality of the distributions of the absolute residuals. The Harrell-Davis percentile estimator [54] with bootstrap was used as an efficient and robust technique to estimate the 95% confidence limits for the median (i.e., the 50th quantile) value of the absolute residuals (see Table 6.11). If the intervals are compared it would seem some overlap, for example, PS2 and PS4 and others do not, for example, PS3 and PS1. Note that the treatments are listed in decreasing order of performance so that smallest residuals — and therefore best predictive performance — occur first. The treatments are labelled for the reader's convenience only, as this information was not available to Researcher 2 at the time of the analysis.

Table 6.11: Harrell-Davis 50th percentile estimators for prediction system absolute residuals

Technique	**Lower bound**	**Upper bound**	**Estimated median**
PS2 (FSW)	582.86	906.49	723.19
PS4 (FSS)	670.42	1048.16	812.35
PS3 (CBR)	974.43	1402.89	1148.43
PS1 (Naïve)	3514.84	3859.27	3699.27

The next part of the analysis turns our attention to effect size [40], in this case measured as Δ, which is defined as the difference in the median absolute residuals for the two treatments being compared and normalised by the pooled standard deviation. This is reported in Tables 6.12 along with the Standardised Accuracy (SA) of each approach relative to guessing based on permutation. To help interpret the effect sizes relative to guessing, these could be characterised as 'small' (~ 0.2) or 'medium' (~ 0.5) and although not obtained, ~ 0.8 might be regarded as a 'large' effect size [32]. Analysed in this way none of the SEE techniques can be seen as particularly successful and this is a powerful reminder of how far we still have to go in pursuit of practical and effective SEE.

Tables 6.12: Results for Finnish dataset (SA and effect size Δ)

Table 6.13: Window size 3

Approach	**Criteria**	
	SA (%)	Δ
FSW	46.99	0.318
FSS	48.34	0.327
CBR	21.00	0.135
Naïve	0.92	-0.007

Table 6.14: Window size 5

Approach	**Criteria**	
	SA (%)	Δ
FSW	58.60	0.399
FSS	62.13	0.424
CBR	33.07	0.220
Naïve	7.38	0.041

Table 6.15: Window size 10

Approach	**Criteria**	
	SA (%)	Δ
FSW	68.56	0.468
FSS	65.98	0.451
CBR	41.08	0.278
Naïve	9.08	0.056

Table 6.16: Window size 15

Approach	**Criteria**	
	SA (%)	Δ
FSW	69.32	0.466
FSS	65.70	0.441
CBR	38.89	0.256
Naïve	11.40	0.066

Tables 6.17: Results for Finnish dataset (SA and effect size Δ)

Table 6.18: Window size 30

Approach	**Criteria**	
	SA (%)	Δ
FSW	71.30	0.467
FSS	63.68	0.415
CBR	42.81	0.271
Naïve	12.63	0.063

Table 6.19: Window size 45

Approach	**Criteria**	
	SA (%)	Δ
FSW	68.81	0.437
FSS	63.82	0.402
CBR	44.60	0.268
Naïve	15.23	0.063

Table 6.20:

No windows — entire Finnish data set

Approach	**Criteria**	
	SA (%)	Δ
FSW	63.51	0.436
FSS	61.23	0.421
CBR	49.35	0.339
Naïve	8.84	0.061

Inferential Tests: The basic descriptive analysis from descriptive stats and Effect size/confidence limits steps suggests that the medians of the absolute residuals appear to

differ by treatment but this difference needs to be tested using inferential statistics. Likewise effect size/confidence limit step suggests that the 95% confidence limits from the medians do not all overlap implying significant differences.

Unfortunately traditional non-parametric tests such as Wilcoxon-Mann-Whitney can lack power [41], do not handle ties well and are unlikely to be satisfactory [172]. For this reason a robust test was used to compare differences in marginal medians using Wilcox's percentile bootstrap using the R function `dmedpb` from the `WRS` library. Family-wise errors arising from multiple testing were controlled using Rom's method since $k < 10$. For details see Wilcox [171].

The predictors are compared pairwise starting with the greatest median difference. The probability of the median difference = 0 is given by p. The upper and lower bounds give the 95% confidence limits for the median difference therefore for a significant difference one would not expect the limits to straddle zero.

Table 6.21: Pairwise comparison of median absolute residual differences using Wilcox's percentile bootstrap

Test	p	**Lower bound**	**Upper bound**	**Median difference**
FSW v Naïve	~ 0	-2983.048	-2213.732	-2738.328
FSS v Naïve	~ 0	-2841.37	-2105.15	-2579.47
CBR v Naïve	~ 0	-2257.923	-1567.178	-1839.484
FSW v CBR	~ 0	-230.1578	-58.22942	-116.9264
FSS v CBR	~ 0	-212.9723	-51.74236	-115.7749
FSW v FSS	0.9	-14.12227	0	0

The analysis is shown in Table 6.21 where the pairwise comparisons between prediction systems are organised in decreasing order of difference this facilitates the application of Rom's method which is based on the idea of sequential rejection so that once a threshold has been exceeded there is no purpose in testing for smaller differences [172]. Again, the results are presented unblinded for the convenience of the reader.

	statistic			
Window size	p	*Lower bound*	*Upper bound*	*Median difference*
Size 3				
FSW v Naïve	~ 0	-1466.658	-1014.749	-1234.648
FSS v Naïve	~ 0	-1530.935	-1071.009	-1261.569
CBR v Naïve	~ 0	-406.3729	-171.2047	-304.7058
FSW v CBR	~ 0	-627.9851	-392.7379	-523.6321
FSS v CBR	~ 0	-685.8376	-411.2059	-551.3394
FSW v FSS	0.918	-0.18916	0	0
Size 5				
FSW v Naïve	~ 0	-2125.6	-1612.217	-1926.077
FSS v Naïve	~ 0	-2155.8	-1750.245	-1916.543
CBR v Naïve	~ 0	-991.5451	-467.4115	-738.9186
FSW v CBR	~ 0	-775.2357	-486.2929	-582.8682
FSS v CBR	~ 0	-890.9782	-601.9987	-754.4851
FSW v FSS	0.914	-0.2111147	0	0
Size 10				
FSW v Naïve	~ 0	-2582.604	-1931.864	-2129.129
FSS v Naïve	~ 0	-2384.833	-1844.053	-2035.337
CBR v Naïve	~ 0	-1470.86	-938.6779	-1102.149
FSW v CBR	~ 0	-764.4481	-442.9886	-582.3201
FSS v CBR	~ 0	-620.2204	-335.6739	-468.8317
FSW v FSS	$\sim$ **0**	-27.34796	-1.68059	-10.03917
Size 15				
FSW v Naïve	~ 0	-2506.792	-2023.476	-2320.333
FSS v Naïve	~ 0	-2455.978	-1938.133	-2155.905
CBR v Naïve	~ 0	-1469.365	-862.1027	-1169.304
FSW v CBR	~ 0	-626.1683	-329.9169	-483.6
FSS v CBR	~ 0	-506.7596	-278.8388	-399.2315
FSW v FSS	$\sim$ **0**	-21.34563	-1.836587	-8.921373
Size 30				
FSW v Naïve	~ 0	-2413.3	-1916.268	-2209.648
FSS v Naïve	~ 0	-2206.874	-1697.714	-1907.658
CBR v Naïve	~ 0	-1693.691	-1238.293	-1434.605
FSW v CBR	~ 0	-760.8053	-342.9459	-582.6768
FSS v CBR	~ 0	-628.0211	-151.9311	-280.9804
FSW v FSS	$\sim$ **0**	-43.16877	-4.980373	-15.00122
Size 45				
FSW v Naïve	~ 0	-2349.639	-1845.058	-2112.106
FSS v Naïve	~ 0	-2177.317	-1599.637	-1878.732
CBR v Naïve	~ 0	-1630.536	-1061.932	-1344.858
FSW v CBR	~ 0	-495.7877	-197.5857	-345.3598
FSS v CBR	~ 0	-416.1132	-124.861	-220.63
FSW v FSS	**0.002**	-78.94588	-2.702868	-25.51186

Table 6.22: Pairwise comparison of median absolute residual differences using Wilcox's percentile bootstrap for each window size for Finnish data set

Discussion of results: The results of the analysis therefore show that whilst the new technique FSW outperforms the naïve sample mean and traditional CBR there is no significant difference with FSS for this particular data set, despite a slightly superior effect size Δ and SA value (see Table 6.20). Thus based on this evidence I cannot argue the new feature weighting technique is superior for this particular data set when validation is performed without regard to completion date of projects.

Table 6.22 shows the results of pairwise comparison of median absolute residual differences using Wilcox's percentile bootstrap for each window size for Desharnais data set. This validation based on moving windows, where a project can only be predicted using past projects to reflect industry practice. The results of the analysis in Table 6.22 therefore show that the new technique FSW outperforms the naïve sample mean, traditional CBR, and FSS for window size 10, 15, 30 and 45 — with significant difference with FSS and a superior effect size Δ and SA value (see Tables 6.12). FSS performs better than FSW in window sizes 3 and 5, but there is no significant difference with FSW. Thus, based on these evidence I can argue that the new feature weighting technique is superior for this particular validation scheme on Finnish data set.

The next section looks at inferential tests for Desharnais data set.

Desharnais results analysis

The inferential tests results for Deharnais data set are reported following the steps in blind process shown in Figure 6.2, starting at step number 7.

Descriptive Stats: As in the analysis for Finnish data set all statistical analysis of the Desharnais results is based on absolute residuals. These were provided with anonymised treatment labels to Researcher 2 by Researcher 3. The four treatments are labelled A, ..., D. Note that for this experiment uses a repeated measures design and there was no particular need to look at context variables or experimental moderators. It is only the treatments labels that needed blinding consequently blind analysis did not inhibit richer or more sophisticated analysis when appropriate.

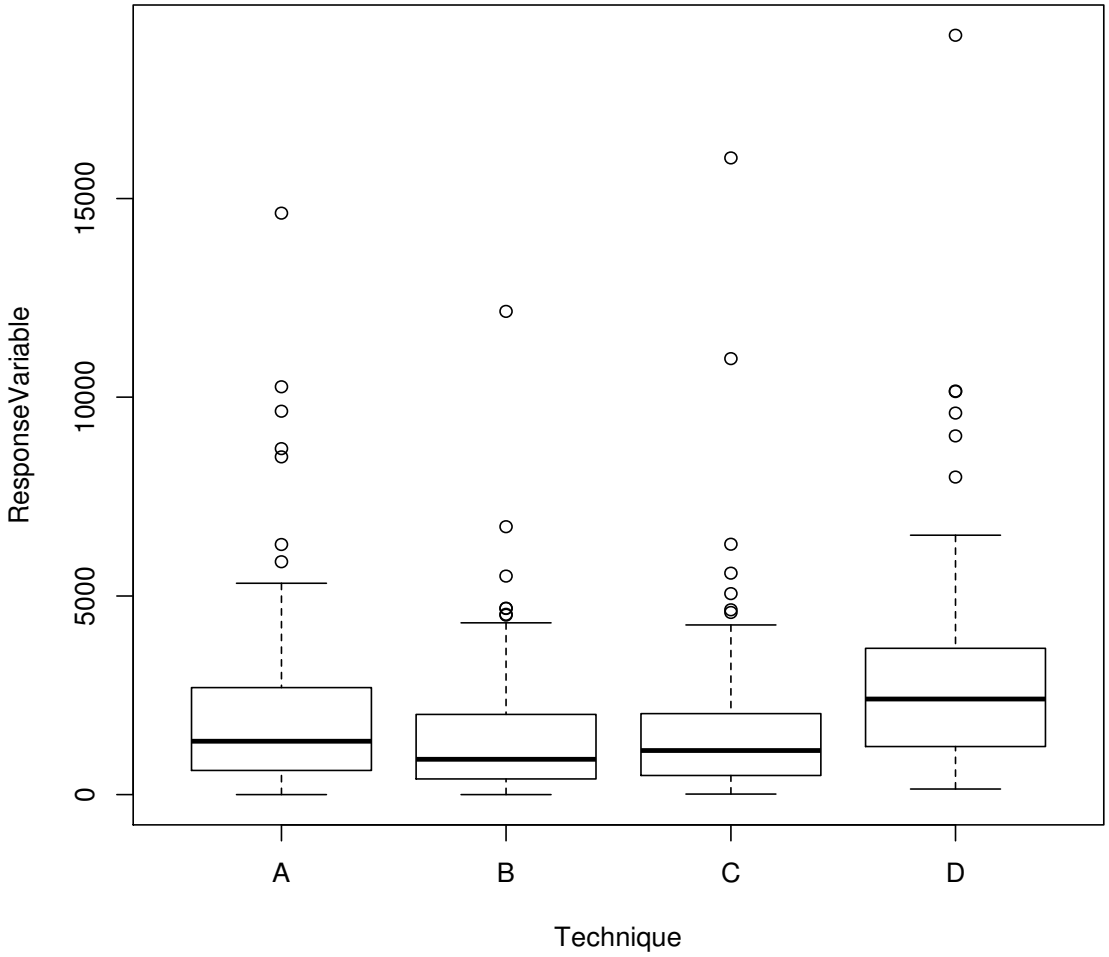


Figure 6.7: Boxplots showing residual distributions for Desharnais Data Set

Descriptive stats, show that there are a few challenges relating to the statistical analysis of the experimental results. First, the distributions of the residuals are extremely skewed and not amenable to simple transformations (see the box plots of the absolute residuals for the techniques A, ..., D. in Figure 6.7).

Effect size, confidence limits: Next Researcher 2 considered the questions of confidence limits for the descriptive statistics such as medians and then measures of effect size. Non-parametric methods are required due to the non-normality of the distributions of the absolute residuals. The Kruskal-Wallis test is a rank-based nonparametric test that is be used to determine if there are statistically significant differences between groups of an independent variable

on a continuous dependent variable. Table 6.23 shows the Kruskal-Wallis test results for Desharnais data set, unblinded for the benefit of the reader.

Table 6.23: Kruskal-Wallis Test

Technique	**N**	**Median**
A (CBR)	77	1340.500
B (FSW)	77	890.7080
C (FSS)	77	1110.680
D (Naïve)	77	2404.900
chi-squared = 28.07, df = 3, p-value = 3.512e-06		

Based on the results in Table 6.23, the estimates of the medians for the four techniques are 1340.5, 2404.9, 1010.754 and 1110.684. I reject the null hypothesis because the p-value of $3.512e^{-}06$ is less than the significance level of 0.05, and conclude that the medians are not all equal, therefore the techniques are different.

The next part of the analysis was to turn attention to effect size [40], in this case measured as Δ which is defined as the difference in the median absolute residuals for the two treatments being compared and normalised by the pooled standard deviation. This is reported in Tables 6.24 along with the Standardised Accuracy (SA) of each approach relative to guessing based on permutation. Note that even using the sample mean is 20.8% better than guessing in the worse case scenario (window size 3).

Tables 6.24: Results for Desharnais dataset (SA and effect size Δ)

Table 6.25: Window size 3

Approach	**Criteria**	
	SA (%)	Δ
FSW	49.61	0.575
FSS	72.59	0.527
CBR	63.71	0.422
Naïve	20.80	0.241

Table 6.26: Window size 5

Approach	**Criteria**	
	SA (%)	Δ
FSW	61.98	0.953
FSS	62.79	1.213
CBR	65.00	1.113
Naïve	31.07	0.626

Table 6.27: Window size 10

Approach	**Criteria**	
	SA (%)	Δ
FSW	64.84	0.396
FSS	54.46	0.406
CBR	59.79	0.430
Naïve	34.33	0.049

Table 6.28: Window size 15

Approach	**Criteria**	
	SA (%)	Δ
FSW	67.23	2.694
FSS	51.91	2.579
CBR	64.43	2.638
Naïve	30.50	2.357

Table 6.29: Window size 30

Approach	**Criteria**	
	SA (%)	Δ
FSW	66.12	2.062
FSS	51.38	1.861
CBR	49.33	2.025
Naïve	37.43	1.582

Table 6.30: Window size 45

Approach	**Criteria**	
	SA (%)	Δ
FSW	80.13	1.276
FSS	74.18	1.092
CBR	53.99	1.066
Naïve	33.67	0.917

Table 6.31: No windows — entire Desharnais data set

Approach	**Criteria**	
	SA (%)	Δ
FSW	67.15	0.775
FSS	62.89	0.726
CBR	53.64	0.619
Naïve	37.77	0.436

Inferential Tests: The basic descriptive analysis from previous two steps suggests that the medians of the absolute residuals appear to differ by treatment but this difference needs to be tested using inferential statistics. A robust test was used to compare differences in

marginal medians using Wilcox's percentile bootstrap using the R function `dmedpb` from the `WRS` library for reasons stated previously for Finnish data set. Again family-wise errors arising from multiple testing were controlled using Rom's method since $k < 10$. The predictors are compared pairwise starting with the greatest median difference. The probability of the median difference = 0 is given by p. The upper and lower bounds give the 95% confidence limits for the median difference therefore for a significant difference one would not expect the limits to straddle zero.

Table 6.32: Pairwise comparison of median absolute residual differences using Wilcox's percentile bootstrap for Desharnais data set

Test	p	**Lower bound**	**Upper bound**	**Median difference**
FSW v Naïve	~ 0	-1825.467	-830.3226	-1550.281
FSS v Naïve	~ 0	-1836.027	-532.2249	-1228.576
CBR v Naïve	0.004	-1393.409	-311.9091	-759.9091
FSW v CBR	0.004	-366.3493	-108.9135	-188.0408
FSS v CBR	0.152	-382.3571	41.58824	-223.6316
FSW v FSS	0.080	-145.1616	1.538595	-28.3533

The analysis is shown in Table 6.32 and 6.33 in where the pairwise comparisons between prediction systems are organised in decreasing order of difference, this facilitates the application of Rom's method which is based on the idea of sequential rejection so that once a threshold has been exceeded there is no purpose in testing for smaller differences [172]. Again the results are presented unblinded for the convenience of the reader.

	statistic			
Window size	p	*Lower bound*	*Upper bound*	*Median difference*
Size 3				
FSW v Naïve	~ 0	-1328.666	-597.0077	-942.6078
FSS v Naïve	~ 0	-2641.896	-575.2948	-1457.416
CBR v Naïve	0.012	-1842.75	-225.75	-835.8333
FSW v CBR	0.744	-713.3697	463.75	-193.5038
FSS v CBR	0.104	-160.621	1.660324	-63.63034
FSW v FSS	0.864	-330.0035	903.4862	72
Size 5				
FSW v Naïve	~ 0	-1876.094	-902.8825	-1359.542
FSS v Naïve	0.002	-1471.135	-112.2625	-562.9878
CBR v Naïve	0.032	-2016	-155.75	-656.6
FSW v CBR	0.02	-1058.4	-285.967	-707.5126
FSS v CBR	0.338	-23.8	244.7794	80.44451
FSW v FSS	**0.006**	-1130.524	-170.7936	-687.8406
Size 10				
FSW v Naïve	~ 0	-1943.872	-1084.669	-1611.311
FSS v Naïve	0.064	-1414.731	89.314548	-734.3471
CBR v Naïve	0.012	-1911.3	-436.45	-887.6
FSW v CBR	0.038	-703.5	-151.0133	-485.9274
FSS v CBR	0.27	-138	1014.185	262.3012
FSW v FSS	**0.014**	-1233.314	-368.7952	-571.7346
Size 15				
FSW v Naïve	~ 0	-2082.877	-1386.638	-1765.683
FSS v Naïve	0.012	-1366.077	-217.8302	-995.7841
CBR v Naïve	~ 0	-2010.383	-712.8167	-1099.833
FSW v CBR	~ 0	-783.8726	-196.8967	-685.2981
FSS v CBR	~ 0	226.4475	914.7807	559.0569
FSW v FSS	$\sim$ **0**	-1462.663	-438.5139	-754.9407
Size 30				
FSW v Naïve	~ 0	-2374.371	-991.6033	-1549.467
FSS v Naïve	0.068	-984.799	66.73333	-503.1151
CBR v Naïve	0.06	-1637.767	194.9667	-996.1667
FSW v CBR	0.016	-1411.743	-485.4379	-1033.07
FSS v CBR	0.684	-531.3	942.8024	55.68182
FSW v FSS	$\sim$ **0**	-1261.834	-382.3223	-922.5947
Size 45				
FSW v Naïve	~ 0	-2967.669	-1157.417	-2150.532
FSS v Naïve	~ 0	-1471.135	-112.2625	-562.9878
CBR v Naïve	~ 0	-2016	-155.75	-656.6
FSW v CBR	~ 0	-1058.4	-285.967	-707.5126
FSS v CBR	0.028	-1351.921	-68.74777	-540.0028
FSW v FSS	0.404	-300.4017	17.96799	-29.06363

Table 6.33: Pairwise comparison of median absolute residual differences using Wilcox's percentile bootstrap for each window size for Desharnais data set

Discussion of results: The results of the analysis in Table 6.32 therefore show that whilst the new technique FSW outperforms the naïve sample mean and traditional CBR there is no significant difference with FSS for this particular data set, despite a superior effect size Δ and SA value (see Table 6.31). Thus, I cannot argue the new feature weighting technique is superior for this particular validation scheme on Desharnais data set. Interestingly, even though FSS outperforms the CBR, there is no significant difference ($p = 0.152$) between them — FSS *should* be an improvement over CBR.

Table 6.33 shows the results of pairwise comparison of median absolute residual differences using Wilcox's percentile bootstrap for each window size for Desharnais data set. This validation based on moving windows (time-series), by which a project can only be predicted using past projects to reflect industry practice. The results of the analysis in Table 6.33 show that the new technique FSW outperforms the naïve sample mean, traditional CBR, and FSS for window size 10, 15 and 30 — with significant difference with FSS and a superior effect size Δ and SA value (see Tables 6.24). FSS performs better than FSW in window size 3 but there is not significant difference with FSW. Thus I can argue within limitation that the new feature weighting technique is superior for this particular validation scheme on Desharnais data set.

6.3 Summary

In summary, after analysing results from 483 software projects, I found out that:

- offering multiple weight sets (FSW) improves accuracy over the standard feature subset selection (FSS) and traditional CBR, when using pseudo time-series validation.
- for very small data sets (with 3 and 5 projects) traditional CBR, FSS and FSW perform almost the same (no statistically difference).
- traditional validation scheme (jackknifing) produces results different to pseudo time-series validation.
- as the window size increases the accuracy improves, but as the window size becomes large the magnitude of increment reduces for all techniques.

Validation based on time-series should be norm to reflect industry practice. This chapter also highlighted the easiness and importance of blind analysis for software engineering experiments. The variance is not very good in the Desharnais data set but always positive therefore the least I can say is that the prediction accuracy would remain the same but for most cases will improve. The degree of improvement will be dependent on the dataset used. The next chapter summarises thesis findings, work done, limitations and possible future research.

Observations

Feature weighting: The order in which weights are assigned features is important. Weights must be assigned in descending order, i.e. largest to smallest; this ensures that influential features are assigned highest weights first, else FSW performs poorly.

Data set: Finnish data set is very challenging for feature weighting. I think its because Finnish data has a lot of enhancement and maintenance projects. These projects have a lot of identical features, which I assume it is the same software development teams doing maintenance of different projects, and have the same duration period and worksup (effort), suggesting these are scheduled maintenance tasks. Therefore, depending on which features are weighted higher (selected for FSS) the distance to the target case can be as high as 1 but the residual being zero. Which goes against principle of CBR that dissimilar projects should have different solutions, on these instances they have identical efforts. The opposite also occurs where donors have small distance (i.e. in CBR theory they are similar) but their solutions are very different.

Blind analysis: Blind analysis also involves a social aspect, after all researchers are human beings, therefore also succumb to social pressures — the relationship between experimenter and analyst may play a role on the results reported — a junior researcher may be reluctant to report poor results for his supervisors 'complex algorithm'. Empirical studies such as [19, 23, 33] show that scientists often fail to adhere to norms normally proposed as part of good scientific practice such as objectivity and neutrality [142]. These studies have also shown how that which comes to be reported as scientific knowledge is partly dictated by social and psychological forces and partly by issues of economic and political power, both within science and in the larger society [142].

Chapter 7

Conclusions and Further Work

This chapter brings together all the research findings of the previous chapters and analyses both the contribution of the work and its limitations. First I summarise work carried out, then the next section focuses on the contributions of this thesis to empirical software engineering. Thereafter, limitations of the research and possible future work are discussed.

7.1 Summary of work done

The following is a summary of work done as part of this PhD:

1. I conducted a comprehensive systematic review of empirical studies of feature weighting ABE in the domain of software engineering.

The literature review entailed the following tasks:

1. A search of *all* peer reviewed studies on feature weighting Analogy-Based Estimation (ABE) from 1990 (first ABE paper was in 1992 [125]) to April 2014 located 183 potential papers, this number reduced to 19 after removing duplicates and reading the titles, abstracts, key words or full text of these papers to select relevant papers based on the inclusion criteria.
2. A meta-analysis on the selected studies.
3. Results were published [153].

From the study it was apparent that, there is widespread consensus that some form of feature weighting technique is beneficial to effort estimation but more research is required because estimates are still not perfect.

2. I introduced blind analysis to software engineering experiments as means to reduce researcher bias (specifically analysis bias).

Numerous empirical studies into the consistence of results of software experiments published in the literature (for example [72, 118, 145, 175]) report inconsistency among software engineering experimental results. To find ways to improve the consistency of results of software experiments I:

1. investigated sources of conflicting results including analysis bias.
2. considered ways of reducing analysis bias in experiments, and I discovered that blind analysis was used in other domains to reduce bias.
3. developed a procedure for blind analysis for software engineering experiments.
4. conducted an action research experiment to investigate the practicalities and viability as means to reduce researcher bias specifically analysis bias.
5. reported results and experiences (published [154]).

3. Proposed a new approach to feature weighting for effort estimation that offers multiple weight sets as opposed to the current situation where a single weight set is offered to a data set.

Having identified a need for research in the area of feature weighting techniques to improve analogy-based effort estimation. I developed a novel approach to feature weighting that involves the use of multiple weight sets. This involved the following:

1. an algorithm to efficiently search for appropriate feature weights.
2. a method to generate multiple weight sets; since this is not just an ensemble of different weights, a method was developed to generate the different weight sets based on a hypothesis rooted on Case-Based Reasoning (CBR) that supports the notion that the next weight set would be better when the previous one fails.
3. a mechanism to select the most appropriate weight set for the target case.

4. Time-series based validation

The analogy estimation technique was validated using a pseudo time-series (moving windows) approach that allowed the technique's performance to be observed in a setting resembling real life. This allowed the approach to be assessed in a more realistic environment akin to industrial practice where a project's effort is estimated based on past projects. For this I:

1. generated multiple data sets of different size based on the completion date of the projects from the main data set.

2. used quantiles to normalise different results from the multiple data sets.

Studies on Desharnais and Finnish data sets showed that proposed feature weighting technique can effectively choose the appropriate weight set and thereby improve estimation accuracy over a technique that uses a single weight set. Therefore offering multiple weights sets is beneficial to effort estimation. Blind analysis also proved to be a viable option for reducing researcher bias in terms of analysis.
I demonstrated that it is easy to transform a non-significant result into a significant one without resorting to scientific misconduct. Therefore, demonstrating the value of blind analysis in software engineering experiments as a vehicle to reduce analysis bias.

7.2 Contribution of this Thesis

Work described in this thesis makes the following contributions:

(i) Through meta-analysis of selected studies for systematic literature review, I established that Feature Weighting (FW) is beneficial to analogy-based estimation

(ii) Although meta-analysis of the studies suggests strong evidence that FW is helpful, it does not mean these techniques get perfect predictions, so there is still a need to improve.

(iii) Blind analysis is a very practical and easy-to-implement method that supports more objective analysis of experimental results, therefore reducing analysis bias.

(iv) Traditional validation methods such as jackknifing that ignore when the projects are completed produce different results from pseudo time-series validation.

(v) A new approach to feature weighting for effort estimation that offers multiple weight sets as opposed to the current situation where a single weight is offered.

- The efficiency of the new approach is comparable with existing techniques.
- After analysing results from 483 software projects from two separate industrial data sets, I conclude that offering multiple weight sets Feature Sequential Weighting (FSW) improves accuracy over the standard Feature Sequential Selection (FSS) and traditional CBR (ABE) when using pseudo time-series validation.
- In summary, the proposed technique enables the user to:
  - Automate the entire feature weight assignment process
  - Efficiently search and generate multiple weight sets
  - Employ multiple weight sets for single data set
  - Identify the most appropriate weight set for the target case

7.3 Limitations and Possible Future Work

Some of the techniques used in this thesis have not been previously applied to software engineering experiments or effort estimation. Therefore, it is important to recognise the limitations of the work presented.

7.3.1 Approach limitations

- The way ArchANGEL handles ties is primitive. If the Euclidean distance of two or more projects is equal, the first project encountered is always chosen as the closest analogy [141]. This becomes very pronounced on the Finnish data set where there are a multiple ties.
- ABE may be sensitive to the choice of the algorithm's similarity function therefore my results are limited to standardised Euclidean distance. Note that Euclidean distance is the normal approach in ABE so there is no reason to believe this is a major limitation.
- In order to validate the FSW using time-series the data sets needed to be sorted. This was achieved through the end-date (date the project was completed). Unfortunately

the data sets had duplicates end-dates, and duplicates were handled by randomly generating numbers to determine the ordering. The limitation for this sorting strategy was that Finnish data sets did not have end-dates, instead two of its features — duration and start-date were used to determine the end-date. Duration was based on months therefore one can only be precise up to 30 days this resulted in multiple duplicate end-dates (again handled by random ordering).

– The other limitation was on the use of the end-date its self — some projects with span a long time, therefore could end up validated using projects started after them because they were completed earlier (i.e., shorter duration).

– Window size could only be based on the number of projects, not time since the distribution of projects per year had a large variance. However, even for a small window size of such as 3, projects were validated using projects completed 2 or 3 three years earlier, this may not reflect well looking at technology advancement in that time period.

7.3.2 Limitations of Analysis

– The data set quality is unknown (recall that Finnish data set had two completed projects with effort recorded as 0, and Desharnais had projects with missing values — note that these projects were excluded from the study) and age of the data sets could also have a limiting factor on the results obtained (Desharnais data set was published in 1989). ABE may be intolerant to noise, therefore my results could be distorted by the presence of noise. Therefore my results are limited to the data sets used.

– Some datasets may not be representing industry standard but only the the elite organisations or only the best projects submitted. Single companies with unstable software development process would result in a dataset that is scattered and inconsistent resulting in the problem of divergences of variances. The same effect can also be observed in heterogeneous dataset for example, multi-organisational datasets such as International Software Benchmarking Standards Group (ISBSG), because organisations have different business models and development maturity. Older datasets may be representing

irrelevant software development approaches and technologies.

– Blind analysis described experiences for an action research experiment where I was also participant. There was no control and $n = 1$. But I demonstrated that it is possible to manipulate results without recourse to poor practice or scientific misconduct.

7.3.3 Possible Future work

· **Sensitivity analysis**

A possible future research work would be to investigate the sensitivity of feature weighting approach against window size (data set size) and number of features. In-depth investigating the efficacy of data cleaning techniques proposed in [106] on the results produced by feature sequentially weighting using different data sets.

· **Validation method**

Investigation of dynamic window sizes i.e., a window size based on both time period and number of projects e.g., the window size would based on the time period or number of projects — which ever produces the required window size first.

Research on the effects of using projects completed after the current project e.g., jackknifing (frequency of *future* projects being used)

· **Blind analysis protocol**

I argue for a protocol for blind analysis, this way, all parties involved know exactly the expectations of the study. One of the conditions could be as already stated in chapter 4 that, before unblinding, researchers should agree that they are confident enough of their analysis to publish whatever the result turns out to be, without further rounds of rethinking.

· **Improve ArchANGEL**

An important addition to improve ArchANGEL Tool would be a mechanism to deal with ties between source analogies. Incorporate benchmark algorithm therefore having Standardised Accuracy (SA) as one of the options for performance measure.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

[2] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. Mahrin. A systematic literature review of software requirements prioritization research. *Information and Software Technology*, 56(6):568–585, 2014.

[3] D. W. Aha. Case-based learning algorithms. In *Proceedings of the 1991 DARPA Case-Based Reasoning Workshop*, volume 1, pages 147–158, 1991.

[4] D. W. Aha and R. L. Goldstone. Concept learning and flexible weighting. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 534–539. Citeseer, 1992.

[5] A. J. Albrecht and J. E. Gaffney. Software function, source lines of code, and development effort prediction: a software science validation. *Software Engineering, IEEE Transactions on*, SE-9(6):639–648, 1983.

[6] S. Amasaki and C. Lokan. The evaluation of weighted moving windows for software effort estimation. In J. Heidrich, M. Oivo, A. Jedlitschka, and M. Baldassarre, editors, *Product-Focused Software Process Improvement*, volume 7983 of *Lecture Notes in Computer Science*, pages 214–228. Springer Berlin Heidelberg, 2013.

[7] L. Angelis and I. Stamelos. A simulation tool for efficient analogy based cost estimation. *Empirical Software Engineering*, 5(1):35–68, 2000.

[8] E. Aprile and et al. Dark matter results from 225 live days of xenon100 data. *Phys. Rev. Lett.*, 109:181301, Nov 2012.

[9] K. D. Ashley and E. L. Rissland. Waiting on weighting: A symbolic least commitment approach. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 239–244, 1988.

[10] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning for control. In *Artificial Intelligence Review*, pages 75–113. Springer, 1997.

[11] M. Auer and S. Biffl. Increasing the accuracy and reliability of analogy-based cost estimation with extensive project feature dimension weighting. In *Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on*, pages 147–155. IEEE, 2004.

[12] M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, and S. Biffl. Optimal project feature weights in analogy-based cost estimation: Improvement and limitations. *Software Engineering, IEEE Transactions on*, 32(2):83–92, 2006.

[13] T. O. Ayodele. *Types of Machine Learning Algorithms, New Advances in Machine Learning*. INTECH Open Access Publisher, 2010.

[14] M. Azzeh, D. Neagu, and P. Cowling. Improving analogy software effort estimation using fuzzy feature subset selection algorithm. In *Proceedings of the 4th international workshop on Predictor models in software engineering*, pages 71–78. ACM, 2008.

[15] M. Azzeh, D. Neagu, and P. Cowling. Software effort estimation based on weighted fuzzy grey relational analysis. In *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, page 8. ACM, 2009.

[16] V. K. Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi. A pso-based model to increase the accuracy of software development effort estimation. *Software Quality Journal*, 21(3):501–526, 2013.

[17] V. K. Bardsiri, A. Khatibi, and E. Khatibi. An optimization-based method to increase the accuracy of software development effort estimation. *Journal of Basic and Applied Scientific Research*, 2013.

[18] R. Bareiss. The experimental evaluation of a case-based learning apprentice. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 162–167, 1989.

[19] B. Barnes. *Scientific knowledge and sociological theory*, volume 2. London and Boston: Roudedge & Kegan Paul, 2013.

[20] R. Bellman. A Markovian decision process. Technical report, DTIC Document, 1957.

[21] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

[22] R. Bisio and F. Malabocchia. Cost estimation of software projects through case base reasoning. In M. Veloso and A. Aamodt, editors, *Case-Based Reasoning Research and Development*, volume 1010 of *Lecture Notes in Computer Science*, pages 11–22. Springer Berlin Heidelberg, 1995.

[23] D. Bloor. *Knowledge and social imagery*. University of Chicago Press, 1991.

[24] B. Boehm, C. Abts, and S. Chulani. Software development cost estimation approaches–a survey. *Annals of Software Engineering*, 10(1-4):177–205, 2000.

[25] B. W. Boehm. *Software Engineering Economics*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1981.

[26] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[27] L. Briand, K. El Emam, D. Surmann, I. Wieczorek, and K. Maxwell. An assessment and comparison of common software cost estimation modeling techniques. In *Software Engineering, 1999. Proceedings of the 1999 International Conference on*, pages 313–323, May 1999.

[28] J. Brownlee. A tour of machine learning algorithms. http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/, 2013. Accessed November 25, 2015.

[29] C. J. Burgess and M. Lefley. Can genetic programming improve software effort estimation? a comparative evaluation. *Information and Software Technology*, 43(14):863–873, 2001.

[30] T. Cain, M. J. Pazzani, and G. Silverstein. Using domain knowledge to influence similarity judgements. In *Proceedings of the Case-Based Reasoning Workshop*, pages 191–198, 1991.

[31] N.-H. Chiu and S.-J. Huang. The adjusted analogy-based software effort estimation based on similarity distances. *Journal of Systems and Software*, 80(4):628–640, 2007.

[32] J. Cohen. A power primer. *Psychological Bulletin*, 112(1):155–159, 1992.

[33] H. M. Collins. Introduction: Stages in the empirical programme of relativism. *Social Studies of Science*, 11(1):3–10, 1981.

[34] S. D. Conte, H. E. Dunsmore, and V. Y. Shen. *Software engineering metrics and models*. Benjamin-Cummings Publishing Co., Inc., 1986.

[35] M. Delgado-Rodríguez and J. Llorca. Bias. *J. of Epidemiolgy and Community Health*, 58:635–641, 2004.

[36] H. Demuth, M. Beale, and M. Hagan. *Neural Network Toolbox™ 6 User's Guide*. The MathWorks, Inc., March, 2010.

[37] J. Desharnais. Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction. Master's thesis, University of Montreal, 1989.

[38] K. Dickersin. The existence of publication bias and risk factors for its occurrence. *J. Am. Med. Assoc.*, 263:1385–1389, 1990.

[39] B. Efron and G. Gong. A leisurely look at the bootstrap, the jackknife and cross-validation. *The American Statistician*, 37(1):36–48, 1983.

[40] P. Ellis. *The Essential Guide to Effect Sizes: Statistical Power, Meta-Analysis, and the Interpretation of Research Results*. Cambridge University Press, 2010.

[41] M. W. Fagerland and L. Sandvik. The Wilcoxon–Mann–Whitney test under scrutiny. *Statistics in Medicine*, 28(10):1487–1497, 2009.

[42] G. Forman. Bns feature scaling: an improved representation over tf-idf for svm text classification. In *Proceedings of the 17th ACM conference on Information and Knowledge Management*, pages 263–270. ACM, 2008.

[43] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *Software Engineering, IEEE Transactions on*, 29(11):985–995, 2003.

[44] J. Fox and S. Weisberg. *An R companion to applied regression*. Sage, 2010.

[45] J. H. Friedman. Flexible metric nearest neighbor classification. *Unpublished manuscript available by anonymous FTP from playfair. stanford. edu (see pub/friedman/README)*, 1994.

[46] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[47] Gartner_Inc. Gartner says worldwide it spending to decline 5.5 percent in 2015. `http://www.gartner.com/newsroom/id/3084817`, 2015.

[48] J. C. Gower and P. Legendre. Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3(1):5–48, 1986.

[49] M. Greenacre and J. Blasius. *Multiple correspondence analysis and related methods*. CRC Press, 2006.

[50] L. K. Grover. Local search and the local structure of NP-complete problems. *Operations Research Letters*, 12(4):235 – 243, 1992.

[51] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[52] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[53] M. T. Hagan, H. B. Demuth, M. H. Beale, et al. *Neural network design*. Pub. Boston, 1996.

[54] F. Harrell and C. Davis. A new distribution-free quantile estimator. *Biometrika*, 69(3):635–640, 1982.

[55] P. F. Harrison. Blind analysis. *Journal of Physics. G. Nuclear and Particle Physics*, 28(10):2679–2691, 2002.

[56] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(6):607–616, 1996.

[57] L. Hedges and I. Olkin. *Statistical methods for meta-analysis*. Academic Press, London, 1985.

[58] J. Heinrich. Benefits of blind analysis techniques. Report CDF/MEMO/STATISTICS/PUBLIC/6576 Version 1, University of Pennsylvania, 2003.

[59] J. Higgins and S. Green. *Cochrane Handbook for Systematic Reviews of Interventions: Version 5.1.0 [updated March 2011]*. The Cochrane Collaboration, 2011.

[60] J. Hihn and G. Tregre. Assuring software cost estimates: Is it an oxymoron? In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 4921–4929, Jan 2013.

[61] S.-J. Huang and N.-H. Chiu. Optimization of analogy weights by genetic algorithm for software effort estimation. *Information and Software Technology*, 48(11):1034–1045, 2006.

[62] W. S. Humphrey. *A discipline for software engineering.* Addison-Wesley Longman Publishing Co., Inc., 1995.

[63] J. Hutton and P. Williamson. Bias in meta-analysis with variable selection within studies. *Applied Statistics*, 49(3):359–70, 2000.

[64] A. Idri, A. Abran, and T. M. Khoshgoftaar. Estimating software project effort by analogy based on linguistic values. In *Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on*, pages 21–30. IEEE, 2002.

[65] A. Idri, F. Azzahra Amazal, and A. Abran. Analogy-based software development effort estimation: A systematic mapping and review. *Information and Software Technology*, 58:206–230, 2015.

[66] A. Idri, A. Zakrani, M. Elkoutbi, and A. Abran. Fuzzy radial basis function neural networks for web applications cost estimation. In *Innovations in Information Technology, 2007. IIT'07. 4th International Conference on*, pages 576–580. IEEE, 2007.

[67] J. Ioannidis. Why most published research findings are false. *PLoS Medicine*, 2(8):e124, 2005.

[68] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing surveys (CSUR)*, 31(3):264–323, 1999.

[69] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings 11th International Conference on Machine Learning*, volume 94, pages 121–129, 1994.

[70] M. Jørgensen, B. Boehm, and S. Rifkin. Software development effort estimation: Formal models or expert judgment? *IEEE software*, 26(2):14–19, 2009.

[71] M. Jørgensen, T. Dybå, K. Liestøl, and D. I. Sjøberg. Incorrect results in software engineering experiments: How to improve research practices. *Journal of Systems and Software*, 2015.

[72] M. Jørgensen, B. Faugli, and T. Gruschke. Characteristics of software engineers with optimistic predictions. *Journal of Systems and Software*, 80(9):1472–1482, 2007.

[73] M. Jorgensen and S. Grimstad. Avoiding irrelevant and misleading information when estimating development effort. *IEEE software*, 25(3):78–83, 2008.

[74] M. Jørgensen, U. Indahl, and D. Sjøberg. Software effort estimation by analogy and "regression toward the mean". *Journal of Systems and Software*, 68(3):253–262, 2003.

[75] M. Jørgensen and K. Moløkken-Østvold. How large are software cost overruns? a review of the 1994 chaos report. *Information and Software Technology*, 48(4):297–301, 2006.

[76] M. Jørgensen and M. Shepperd. A systematic review of software development cost estimation studies. *Software Engineering, IEEE Transactions on*, 33(1):33–53, 2007.

[77] M. Jørgensen and D. I. Sjøberg. The impact of customer expectation on software development effort estimates. *International Journal of Project Management*, 22(4):317–325, 2004.

[78] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, pages 237–285, 1996.

[79] C. F. Kemerer. An empirical validation of software cost estimation models. *Communications of the ACM*, 30(5):416–429, 1987.

[80] J. Keung. *Providing Statistical Inference to Case-Based Software Effort Estimation.* PhD thesis, University of New South Wales, Australia, 2007.

[81] J. Keung. Software development cost estimation using analogy: a review. In *Software Engineering Conference, 2009. ASWEC'09. Australian*, pages 327–336. IEEE, 2009.

[82] J. W. Keung. Theoretical maximum prediction accuracy for analogy-based software cost estimation. In *Software Engineering Conference, 2008. APSEC'08. 15th Asia-Pacific*, pages 495–502. IEEE, 2008.

[83] J. W. Keung and B. Kitchenham. Optimising project feature weights for analogy-based software cost estimation using the mantel correlation. In *Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific*, pages 222–229. IEEE, 2007.

[84] J. W. Keung, B. A. Kitchenham, and D. R. Jeffery. Analogy-x: providing statistical inference to analogy-based software cost estimation. *Software Engineering, IEEE Transactions on*, 34(4):471–484, 2008.

[85] C. Kirsopp and M. Shepperd. Case-based software project effort prediction. In *Bournemouth University, Technical Report*, 2002.

[86] C. Kirsopp and M. Shepperd. Case and feature subset selection in case-based software project effort prediction. In *the Twenty-second SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence*, pages 61–74, 2003.

[87] C. Kirsopp, M. J. Shepperd, and J. Hart. Search heuristics, case-based reasoning and software project effort prediction. 2002.

[88] B. Kitchenham, S. L. Pfleeger, B. McColl, and S. Eagan. An empirical study of maintenance and development estimation accuracy. *Journal of Systems and Software*, 64(1):57 – 77, 2002.

[89] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd. What accuracy statistics really measure [software estimation]. In *Software, IEE Proceedings-*, volume 148, pages 81–85. IET, 2001.

[90] J. R. Klein and A. Roodman. Blind analysis in nuclear and particle physics. *Annual Review of Nuclear and Particle Science*, 55:141–163, 2005.

[91] E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung. Exploiting the essential assumptions of analogy-based effort estimation. *Software Engineering, IEEE Transactions on*, 38(2):425–438, 2012.

[92] E. Kocaguneli, T. Menzies, and J. W. Keung. Kernel methods for software effort estimation. *Empirical Software Engineering*, 18(1):1–24, 2013.

[93] R. Kohavi and G. John. Wrappers for feature selection for machine learning. *Artificial Intelligence*, 97:273–324, 1997.

[94] R. Kohavi and F. Provost. Glossary of terms. *Machine Learning*, 30:271–274, 1998.

[95] A. Kołcz and C. H. Teo. Feature weighting for improved classifier robustness. In *CEAS'09: Sixth Conference on Email and Anti-spam*, 2009.

[96] A. Kołcz and W.-T. Yih. Raising the baseline for high-precision text classifiers. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 400–409. ACM, 2007.

[97] D. Koller and M. Sahami. Toward optimal feature selection. Technical Report 1996-77, Stanford InfoLab, February 1996. Previous number = SIDL-WP-1996-0032.

[98] J. Kolodner. *Case-Based Reasoning*. Morgan-Kaufmann, 1993.

[99] L. M. Laird. The limitations of estimation. *IT professional*, 8(6):40–45, 2006.

[100] M. Lefley and M. Shepperd. Using genetic programming to improve software effort estimation based on general data sets. In E. Cantú-Paz, J. Foster, and K. Deb, editors, *Genetic and Evolutionary Computation — GECCO 2003*, volume 2724 of *Lecture Notes in Computer Science*, pages 2477–2487. Springer Berlin Heidelberg, 2003.

[101] K. M. Leung. Naive bayesian classifier. `https://tom.host.cs.st-andrews.ac.uk/ID5059/L15-LeungSlides.pdf`, 2007.

[102] J. Li and G. Ruhe. Analysis of attribute weighting heuristics for analogy-based software effort estimation method aqua+. *Empirical Software Engineering*, 13(1):63–96, 2008.

[103] J. Li and G. Ruhe. Analysis of attribute weighting heuristics for analogy-based software effort estimation method aqua+. *Empirical Software Engineering*, 13(1):63–96, 2008.

[104] Y. F. Li, M. Xie, and T. N. Goh. A study of mutual information based feature selection for case based reasoning in software cost estimation. *Expert Systems with Applications*, 36:5921–5931, 2009.

[105] Y.-F. Li, M. Xie, and T. N. Goh. A study of project selection and feature weighting for analogy based software cost estimation. *Journal of Systems and Software*, 82(2):241–252, 2009.

[106] G. A. Liebchen. *Data cleaning techniques for software engineering data sets*. PhD thesis, Brunel University, School of Information Systems, Computing and Mathematics, 2010.

[107] C. Lokan and E. Mendes. Applying moving windows to software effort estimation. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 111–122. IEEE Computer Society, 2009.

[108] C. Lokan and E. Mendes. Using chronological splitting to compare cross- and single-company effort models: Further investigation. In B. Mans, editor, *Thirty-Second Australasian Computer Science Conference (ACSC 2009)*, volume 91 of *CRPIT*, pages 35–42, Wellington, New Zealand, 2009. ACS.

[109] C. G. Lord, M. R. Lepper, and E. Preston. Considering the opposite: a corrective strategy for social judgment. *Journal of Personality and Social Psychology*, 47(6):1231, 1984.

[110] R. MacCoun and S. Perlmutter. Blind analysis: Hide results to seek the truth. *Nature*, 526(7572):187–189, 2015.

[111] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, and S. Webster. An investigation of machine learning based prediction systems. *Journal of Systems and Software*, 53(1):23–29, 2000.

[112] C. Mair, M. Martincova, and M. Shepperd. An empirical study of software project managers using a case-based reasoner. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 1030–1039. IEEE, 2012.

[113] C. Mair and M. Shepperd. The consistency of empirical comparisons of regression and analogy-based software project cost prediction. In *Empirical Software Engineering, 2005. 2005 International Symposium on*, pages 10 pp.–, Nov 2005.

[114] A. H. Maslow and K. J. Lewis. *Maslow's hierarchy of needs*. Salenger Incorporated, 1987.

[115] E. Mendes, N. Mosley, and S. Counsell. A replicated assessment of the use of adaptation rules to improve web cost estimation. In *Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on*, pages 100–109. IEEE, 2003.

[116] E. Mendes, I. Watson, C. Triggs, N. Mosley, and S. Counsell. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2):163–196, 2003.

[117] T. Menzies, R. Krishna, and D. Pryor. The promise repository of empirical software engineering data. `http://openscience.us/repo.NorthCarolinaStateUniversity, DepartmentofComputerScience`, 2015. Accessed November 25, 2015.

[118] T. Menzies and M. Shepperd. Editorial: Special issue on repeatable results in software engineering prediction. *Empirical Software Engineering*, 17(1-2):1–17, 2012.

[119] A. J. Miller. Selection of subsets of regression variables. *Journal of the Royal Statistical Society. Series A (General)*, 147(3):389–425, 1984.

[120] L. L. Minku and X. Yao. Ensembles and locality: Insight on improving software effort estimation. *Information and Software Technology*, 55(8):1512–1528, 2013.

[121] N. Mittas, M. Athanasiades, and L. Angelis. Improving analogy-based software cost estimation by a resampling method. *Information & Software Technology*, 50(3):221–230, 2008.

[122] T. Mohri and H. Tanaka. An optimal weighting criterion of case indexing for both numeric and symbolic attributes. In *AAAI-94 Workshop Program: Case-Based Reasoning, Working Notes*, pages 123–127, 1994.

[123] K. Molokken and M. Jorgensen. A review of software surveys on software effort estimation. In *Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on*, pages 223–230. IEEE, 2003.

[124] K. Molokken-Ostvold and N. C. Haugen. Combining estimates with planning poker–an empirical study. In *Software Engineering Conference, 2007. ASWEC 2007. 18th Australian*, pages 349–358. IEEE, 2007.

[125] T. Mukhopadhyay, S. S. Vicinanza, and M. J. Prietula. Examining the feasibility of a case-based reasoning model for software effort estimation. *MIS Quarterly*, pages 155–171, 1992.

[126] I. Myrtveit, E. Stensrud, and M. Shepperd. Reliability and validity in comparative studies of software prediction models. *Software Engineering, IEEE Transactions on*, 31(5):380–391, May 2005.

[127] M. Obitko. Prediction using neural networks. `http://www.obitko.com/tutorials/neural-network-prediction/prediction-using-neural-networks.html`, 1999.

[128] A. L. Oliveira. Estimation of software project effort with support vector regression. *Neurocomputing*, 69(13):1749–1753, 2006.

[129] S. K. Pal and S. C. Shiu. *Foundations of soft case-based reasoning*, volume 8. John Wiley & Sons, 2004.

[130] W. Pedrycz. Computational intelligence as an emerging paradigm of software engineering. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 7–14. ACM, 2002.

[131] O. Pfungst. *Clever Hans:(the horse of Mr. Von Osten.) a contribution to experimental animal and human psychology.* New York: Henry Holt and Company, 1911.

[132] B. W. Porter, R. Bareiss, and R. C. Holte. Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45(1):229–263, 1990.

[133] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[134] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[135] R. Rosenthal. The "file drawer problem" and tolerance for null results. *Psychological Bulletin*, 86(3):638–641, 1979.

[136] S. Russell and P. Norvig. *Artificial Intelligence: A modern approach*. Pearson New International Edition, 2013.

[137] Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[138] R. D. Sagor. *The action research guidebook: A four-stage process for educators and school teams*. Corwin Press, 2010.

[139] Machine learning: What it is and why it matters. `http://www.sas.com/en_us/insights/analytics/machine-learning.html/`. Accessed November 25, 2015.

[140] R. Schapire. Machine learning algorithms for classification. In *Princeton University, slides*, 2006.

[141] C. Schofield. *An Empirical Investigation into Software Effort Estimation by Analogy*. PhD thesis, Bournemouth University, United Kingdom, 1998.

[142] W. R. Shadish, T. D. Cook, and D. T. Campbell. *Experimental and quasi-experimental designs for generalized causal inference*. Wadsworth Cengage learning, 2002.

[143] M. Shepperd. Case-based reasoning and software engineering. Technical Report TR02-08, Bournemouth University, UK, Bournemouth University, UK, 2002.

[144] M. Shepperd. Case-based reasoning and software engineering. In A. Aurum, R. Jeffery, C. Wohlin, and M. Handzic, editors, *Managing Software Engineering Knowledge*, pages 181–198. Springer Berlin Heidelberg, 2003.

[145] M. Shepperd. How do I know whether to trust a research result? *Software, IEEE*, 32(1):106–109, 2015.

[146] M. Shepperd, D. Bowes, and T. Hall. Researcher bias: The use of machine learning in software defect prediction. *on Software Engineering, IEEE Transactions*, 40(6):603–616, 2014.

[147] M. Shepperd and G. Kadoda. Comparing Software Prediction Techniques Using Simulation. *Software Engineering, IEEE Transactions on*, 27(11):1014–1022, Nov. 2001.

[148] M. Shepperd and G. Kadoda. Using simulation to evaluate prediction techniques [for software]. In *Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International*, pages 349–359. IEEE, 2001.

[149] M. Shepperd and S. MacDonell. Evaluating prediction systems in software project estimation. *Information and Software Technology*, 54(8):820–827, Jan. 2012.

[150] M. Shepperd and C. Schofield. Estimating software project effort using analogies. *Software Engineering, IEEE Transactions on*, 23(11):736–743, 1997.

[151] M. Shepperd, C. Schofield, and B. Kitchenham. Effort estimation using analogy. In *Proceedings of the 18th International Conference on Software Engineering*, pages 170–178. IEEE Computer Society, 1996.

[152] B. Sigweni. Feature weighting for case-based reasoning software project effort estimation. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 54:1–54:4, New York, NY, USA, 2014. ACM.

[153] B. Sigweni and M. Shepperd. Feature weighting techniques for CBR in software effort estimation studies: a review and empirical evaluation. In *The 10th International Conference on Predictive Models in Software Engineering*, pages 32–41. ACM, 2014.

[154] B. Sigweni and M. Shepperd. Using blind analysis for software engineering experiments. In *The 19th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2015.

[155] R. Silberzahn and E. L. Uhlmann. Crowdsourced research: Many hands make tight work. *Nature*, 526(7572):189–191, 2015.

[156] D. Skalak. Representing cases as knowledge sources that apply local similarity metrics. In *Proc. of the 14th Annual Conference of the Cognitive Science Society*, pages 325–330, 1992.

[157] D. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *11th Intl. Machine Learning Conf. (ICML-94)*, pages 293–301. Morgan Kauffmann, 1994.

[158] Q. Song and M. Shepperd. Predicting software project effort: A grey relational analysis based method. *Expert Systems with Applications*, 38(6):7302–7316, 2011.

[159] K. Srinivasan and D. Fisher. Machine learning approaches to estimating software development effort. *Software Engineering, IEEE Transactions on*, 21(2):126–137, 1995.

[160] I. Stamelos, L. Angelis, and E. Sakellaris. Brace: Bootstrap based analogy cost estimation: Automated support for an enhanced effort prediction method. In *Proceedings 12th European Software Control and Metrics Conference (ESCOM'2001)*, pages 17–23. Citeseer, 2001.

[161] P.-N. Tan, M. Steinbach, V. Kumar, et al. *Introduction to data mining*, volume 1. Pearson Addison Wesley Boston, 2006.

[162] J. Tang, S. Alelyani, and H. Liu. Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, page 37, 2014.

[163] A. Tosun, B. Turhan, and A. B. Bener. Feature weighting heuristics for analogy-based effort estimation models. *Expert Systems with Applications*, 36(7):10325–10333, 2009.

[164] F. Walkerden and R. Jeffery. An empirical study of analogy-based software effort estimation. *Empirical Software Engineering*, 4(2):135–158, 1999.

[165] C. J. Watkins and P. Dayan. Technical note: Q-learning. In *Reinforcement Learning*, pages 55–68. Springer, 1992.

[166] C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, University of Cambridge, 1989.

[167] M. Wattenberg and A. Juels. Stochastic hillclimbing as a baseline method for evaluating genetic algorithms. In *Proceedings of the 1995 Conference*, volume 8, page 430, 1996.

[168] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang. Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, 54:41–59, 2012.

[169] J. Wen, S. Li, and L. Tang. Improve analogy-based software effort estimation using principal components analysis and correlation weighting. In *Software Engineering Conference, 2009. APSEC'09. Asia-Pacific*, pages 179–186. IEEE, 2009.

[170] D. Wettschereck, D. W. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11(1-5):273–314, 1997.

[171] R. Wilcox. Pairwise comparisons of dependent groups based on medians. *Computational Statistics and Data Analysis*, 50(10):2933–2941, 2006.

[172] R. Wilcox. *Introduction to robust estimation and hypothesis testing (3rd Edn)*. Academic Press, 3rd edition, 2012.

[173] D. Wu, J. Li, and Y. Liang. Linear combination of multiple case-based reasoning with optimized weight for software effort estimation. *The Journal of Supercomputing*, 64(3):898–918, 2013.

[174] Z. Xu and T. M. Khoshgoftaar. Identification of fuzzy models of software cost estimation. *Fuzzy Sets and Systems*, 145(1):141–163, 2004.

[175] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 91–100. ACM, 2009.

Appendix A

Table A1: Selected studies

ID	Year	Study Author(s)	Source	Weighting Technique	Weight space	Ref
[S01]	2000	Mair et al.	Journal	Exhaustive Search	Binary	[111]
[S02]	2002	Kirsopp and Shepperd	Conference	Search Heuristics	Binary	[85]
[S03]	2003	Mendes et al.	Journal	Inverse Rank Weighted Mean	Continuous	[116]
[S04]	2004	Auer and Biffl	Conference	Exhaustive Dimension Weighting	Continuous	[11]
[S05]	2006	Auer et al.	Journal	Exhaustive Search	Continuous	[12]
[S06]	2006	Huang and Chiu	Journal	Genetic Algorithm	Continuous	[61]
[S07]	2007	Keung and Kitchenham	Conference	Mantel's Correlation	Continuous	[83]
[S08]	2008	Li and Ruhe	Journal	Rough Set Analysis	Continuous	[103]
[S09]	2008	Azzeh et al.	Conference	Fuzzy Logic	Binary	[14]
[S10]	2008	Keung et al.	Journal	Mantel's Correlation	Binary	[84]
[S11]	2009	Wen et al.	Conference	Principal Components Analysis (PCA)	Continuous	[169]
[S12]	2009	Li et al.	Journal	Genetic Algorithm	Continuous	[105]
[S13]	2009	Tosun et al.	Journal	PCA with Correlation Weighting	Continuous	[163]
[S14]	2009	Li et al.	Journal	Mutual Information	Binary	[104]
[S15]	2011	Kocaguneli et al.	Journal	Kernel Methods	Continuous	[92]
[S16]	2011	Song and Shepperd	Journal	Grey Relational Analysis	Binary	[158]
[S17]	2013	Bardsiri et al.	Journal	Genetic Algorithm	Continuous	[17]
[S18]	2013	Bardsiri et al.	Journal	Particle Swarm Optimisation	Continuous	[16]
[S19]	2013	Wu et al.	Journal	Combinations	Continuous	[173]

Appendix B

Table B1: Finnish data set features

No.	Feature	Description
1	Project_tech_ID	Project code
2	Project_name	Project identifier
3	Business_names	Business sector of the customer
4	Protype_names	Development type
5	Hardware_names	Hardware/platform type
6	Duration	Duration of the project in months
7	Size_ep99_proj	Total size of the project software in Experience 2.0 FP's
8	Worksup	Total effort of the supplier
9	SituCoeff	Situation coefficient multiplier
10	T01	Involvement of the customer representatives
11	T02	Performance and availability of the development environment
12	T03	Availability of IT staff
13	T04	Number of stakeholders
14	T05	Pressure on schedule
15	T06	Impact of standards
16	T07	Impact of methods
17	T08	Impact of tools
18	T09	Level of change management
19	T10	Maturity of software development process
20	T11	Logical complexity of software
21	T12	Size of database based on number of entities
22	T13	Number of interfaces to other software
23	T14	Quality requirements of software
24	T15	Efficiency requirements of software
25	T16	Training and installation/platform requirements
26	T17	Analysis skills of staff
27	T18	Application knowledge of staff
28	T19	Tool skills of staff
29	T20	Experience of project management
30	T21	Team skills of the project team
31	InpTot	Total number of input functions
32	InpFP	Size of inputs in Experience 2.0 fp's
33	InqTot	Total number of inquiry functions
34	InqFP	Size of inquiries in Experience 2.0 fp's
35	OutTot	Total number of output functions
36	OutFP	Size of outputs in Experience 2.0 fp's
37	IntTot	Total number of interface functions
38	IntFP	Size of interfaces in Experience 2.0 fp's
39	EntTot	Total number of entities
40	EntFP	Size of entities in Experience 2.0 fp's
41	AlgTot	Total number of algorithmic functions
42	AlgFP	Size of algorithms in Experience 2.0 fp's
43	AllTot	Total number of all types of functions
44	AllFP_ep20	Application size in Experience 2.0 fp's