



Run Run Shaw Library

香港城市大學
City University of Hong Kong

Copyright Warning

Use of this thesis/dissertation/project is for the purpose of private study or scholarly research only. ***Users must comply with the Copyright Ordinance.***

Anyone who consults this thesis/dissertation/project is understood to recognise that its copyright rests with its author and that no part of it may be reproduced without the author's prior written consent.

CITY UNIVERSITY OF HONG KONG
香港城市大學

A Framework to Sample Bellwether Moving
Windows for Improving Software Effort
Estimation

一種提高軟件工作量預測的移動窗口抽樣框架

Submitted to
Department of Computer Science
電腦科學系
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
哲學博士學位

by

Mensah Solomon

August 2018
二零一八年八月

Abstract

Context: Software effort estimation (SEE) is the process used in predicting the effort needed for software development, thus supporting scheduling, costing and the allocation of resources to meet project delivery deadlines. Accurately predicting the required effort of a *new* software project plays a key role in the strategic planning of project development and delivery since it minimises overestimation and underestimation. Previous studies have considered the use of recently completed projects (defined as *moving windows*) in developing SEE models which works by the assumption that *moving windows are more relevant for improving prediction accuracy of new projects*. However, the following constraints exist when considering moving windows for setting up SEE models for *new* projects: (1) the relevancy constraint of projects forming the moving window, (2) the sizing constraint of the moving window, (3) the aging constraint of the moving window, and (4) the weighting function constraint of the moving window. Thus the sampling of moving windows from chronological projects to be used as the training set for SEE models is known to have an impact on prediction accuracy based on the four aforementioned constraints. Furthermore, *conclusion instability* across SEE models (or learners) for a given dataset still threatens the reliability of prediction accuracy for *new* projects. Such instability may be attributed to the sampling problem based on the four aforementioned constraints and the lack of an *effort classification* benchmark in SEE. Thus given relevant sampling of moving windows from historical projects and trained with a set of SEE learners, then the learner(s) which yields accurate classifications of predicted effort of hold-out projects can be used for *new* project estimation. For example,

given a set of learners (say $l_1, l_2, \dots, l_n$ where $i \dots n \in N$) trained and validated on a given set of historical projects with their respective *effort values*. If l_i and l_j ($\forall i, j \in N$) yield accurate classifications of their respective estimated efforts, then their prediction accuracies can be relatively trusted when use for *new* project estimation.

Objectives: The study seeks to tackle the four aforementioned constraints by conducting theoretical and empirical investigations. Thus the study sets out to first investigate the existence of exemplary and recently completed projects with defined window size and age parameters, and whether their use in SEE modelling improves prediction accuracy. Such exemplary and recently completed projects are referred to as the *Bellwether moving window* (BMW). This study also seeks to ameliorate the *conclusion instability* issue across learners using the BMW and an *effort classification* scheme. The study contributes with an introduced guideline (based on the studied datasets) to assist researchers and practitioners to sample BMW to be used for training prediction models.

Method: Empirical investigation of the moving window assumption was done based on the theory that *the prediction outcome of a future event depends on the outcome of prior events*. This investigation study was initialised with six postulations which were theoretically and empirically proven to establish the existence of BMW in a given chronological dataset to be used for SEE modelling. Sampling of the exemplary or relevant projects (*Bellwethers*) from four chronological datasets (*i.e.*, the ISBSG dataset, Kitchenham dataset, Desharnais dataset and Maxwell dataset) was undertaken using three introduced *Bellwether methods* (*SSPM*, *SysSam* and *RandSam*). The sizing and aging constraints of the BMW were addressed based on the Markov chain Monte Carlo methodology. The ergodic Markov chain was used to determine the stationarity of the

BMW. Two weighting functions (namely Biweight and Triweight functions) were introduced in addition to the existing four weighting functions (*i.e.*, Triangular, Epanechnikov, Gaussian and Rectangular) to further investigate their effects on prediction accuracy. The sampled BMW was benchmarked against the *growing portfolio* (the entire collection of historical projects) and were used to predict the targets of hold-out projects. The Eubank's optimal spacing theory based on the density-quantile function was adopted to discretise the software effort values of the studied datasets into three classes (*high*, *moderate* and *low*). This classification scheme is to provide a benchmark to assist researchers and practitioners to assess prediction results from different learners and select the learner(s) that provides accurate classifications of the hold-out projects estimation. Robust test statistics comprising Brunner's ANOVA-like test, Yuen's test and Cliff's δ effect size as well as mean absolute error (MAE) were used to assess the prediction performance of three learners – the baseline Automatically Transformed Linear Model (ATLM), ElasticNet regression and Deep neural networks. Statistical inferences were made at the 5% asymptotic significance level.

Results: Empirical results show that (1) *Bellwethers* are evident in the studied chronological datasets for SEE modelling, (2) the BMW has an approximate size of 40 to 80 exemplary projects that should not be more than 3 years old relative to the hold-out projects being estimated, (3) weighting the BMW with the Triweight function was more advantageous with respect to relative prediction accuracy against the *growing portfolio* benchmark, (4) training the Deep neural networks with the BMW minimises *conclusion instability* in SEE modelling against the *growing portfolio* benchmark, (5) the characteristics of the exemplary projects that constituted the BMW were further

investigated and was found that *trimming + log transforming* the data yielded improved prediction accuracy as compared to using untransformed data.

Conclusion: Results from the study show the effectiveness and recommended use of BMW for improved prediction accuracy in SEE modelling. The introduced guideline for BMW will serve as a foundation to assist software engineers and researchers to sample relevant training and validation sets with defined window size and age parameters prior to the development of *new* software projects. The software effort classification scheme is to assist researchers and practitioners to interpret the likely level of effort to be expended in software project development.

Keywords: *Bellwether effect, Bellwether moving windows, Growing portfolio, Markov chains, Software effort estimation.*

CITY UNIVERSITY OF HONG KONG
Qualifying Panel and Examination Panel

Surname: MENSAH
First Name: Solomon
Degree: PhD
College/Department: Department of Computer Science

The Qualifying Panel of the above student is composed of:

Supervisor(s)

Dr. KEUNG Wai Jacky Department of Computer Science
City University of Hong Kong

Qualifying Panel Member(s)

Dr. CHAN Wing Kwong Department of Computer Science
City University of Hong Kong

Dr. LI Zhenjiang Department of Computer Science
City University of Hong Kong

This thesis has been examined and approved by the following examiners:

Dr. LI Minming Department of Computer Science
City University of Hong Kong

Prof. WANG Jianping Department of Computer Science
City University of Hong Kong

Dr. KEUNG Wai Jacky Department of Computer Science
City University of Hong Kong

Dr. LAI Richard Y Department of Computer Science and Information
Technology
La Trobe University

Acknowledgement

The *PhD voyage* is full of hurdles and a very frustrating one indeed that can only be accomplished successfully with the support of people with profound interest in your intellectual and professional development. Therefore, I will like to recognize and appreciate those that have contributed tremendously to my *PhD*. First, all thanks and heartfelt gratitude goes to the Almighty God for His strength, guidance and direction in my life. Second, I will like to acknowledge my main supervisor and also the chair of my dissertation committee, Assistant Prof. Jacky Wai Keung, for your invaluable support during each stage of the *PhD voyage*. Your influence on my academic pursuit has undoubtedly been phenomenal. I thank you sincerely for your constructive criticisms and encouragements when things really looked bad. More importantly, I am grateful you believed in my academic and research capabilities, and provided the needed environment and conditions that made my *PhD voyage* a memorable one. I would also like to express my profound gratitude to the members of my dissertation committee, Associate Prof. Kong Wing Chan and Assistant Prof. Zhenjiang Li, for their invaluable constructive comments and criticisms which have shaped and significantly improved the quality and originality of my dissertation. My appreciation also goes to Prof. Stephen MacDonell, Dr. Michael Bosu and Dr. Ferdinand Katsriku for your candid comments, advice, and importantly your involvement in my *PhD voyage*. I thank you for the encouragement and support given me over the years in my PhD pursuit. My profound heartfelt gratitude also goes to Patrick, Brown, Elvis and Etta for your support and encouragement. To my friends in City University of Hong Kong (Ebo, Michael, Edward, Sammy, Ernest, Sylvester,

Jamal, Albert, Emmanuel, Bright, Hannah, Mi, Tunde and Raheem), I thank you for being there for me and the encouragement given me. Lastly, to anyone who have contributed in one way or the way towards the completion of my *PhD programme*, I say thank you. God richly bless you all.

Table of Contents

Abstract	i
Acknowledgement	v
List of Tables	x
List of Figures	xi
Nomenclature	xii
Chapter 1	1
Introduction	1
1.1. Background and Motivation	1
1.2. Problem Statement	7
1.3. Scope of Study	9
1.4. Research Objectives	10
1.4.1. Global Objective	10
1.4.2. Specific Objectives	10
1.5. Research Contribution	10
1.6. Organization of Study	11
Chapter 2	14
Literature Review	14
2.1. Preliminaries	14
2.1.1. The Concept of Bellwether	14
2.1.2. Moving Window and Growing Portfolio	17
2.1.3. Markov chain Monte Carlo	19
2.1.4. Central Limit Theorem and Law of Large Numbers	21
2.1.5. Statistical Measures	22
2.2. Review on Bellwethers	25
2.3. Review on Moving Windows	36
2.4. Review on Growing Portfolio	41
2.5. Differences between Current study and Previous studies	48

Chapter 3	52
Methodology	52
3.1. Description of Datasets	52
3.2. Data Preprocessing.....	58
3.3. Theoretical Postulations	69
3.4. The Bellwether Methods.....	76
3.4.1. Stratification, Sampling and Prediction Method.....	76
3.4.2. Benchmark Sampling Methods	79
3.4.2.1. Systematic Sampling (<i>SysSam</i>).....	79
3.4.2.2. Random sampling (<i>RandSam</i>)	80
3.4.3. Assumptions.....	80
3.5. Experimental Setup	81
3.5.1. The Automatically Transformed Linear Model	82
3.5.2. The ElasticNet Regression.....	83
3.5.3. The Deep Neural Network (DNN).....	84
3.6. Performance Measures	87
Chapter 4.....	89
Results and Discussion	89
4.1. Bellwether effect in chronological datasets	89
4.2. Statistical significance of Bellwethers with respect to weighting/unweighting functions.....	97
4.3. Prediction of the p_i hold-out projects by the Bellwether and the growing portfolio benchmark.....	100
4.4. Guideline for Sampling Bellwethers.....	105
Chapter 5.....	109
Duplex Output Prediction Modelling.....	109
5.1. Software Effort Classification Scheme	109
5.2. Duplex Output Prediction Algorithm.....	113
5.3. Classification of Software Effort	116

Chapter 6.....	121
Final Remarks	121
6.1. Summary and Conclusion	121
6.2. Threats to Validity	125
6.2.1. External Validity	125
6.2.2. Internal Validity	125
6.2.3. Construct Validity	126
6.2.4. Conclusion Validity	127
6.3. Future Work	127
Bibliography	130
Appendix.....	142
Appendix 1: <i>Feature/Variable Definition in Kitchenham Dataset</i>	142
Appendix 2: <i>Feature/Variable Definition in Desharnais Dataset</i>	142
Appendix 3: <i>Feature/Variable Definition in ISBSG Dataset</i>	143
Appendix 4: <i>Feature/Variable Definition in Maxwell Dataset</i>	145

List of Tables

Table 1 Descriptive Statistics of Sampled Features in the Desharnais Dataset	53
Table 2 Descriptive Statistics of Sampled Features in the Kitchenham Dataset	55
Table 3 Descriptive Statistics of Sampled Features in the ISBSG Dataset	57
Table 4 Descriptive statistics of sampled features in the Maxwell Dataset.....	58
Table 5 Statistics of Outliers and Influential data points identified and trimmed off.....	66
Table 6 MAE evaluation of potential Bellwethers sampled by SSPM, SysSam and RandSam against the growing portfolio benchmark in the Desharnais dataset	91
Table 7 MAE evaluation of potential Bellwethers sampled by SSPM, SysSam and RandSam against the growing portfolio benchmark in the Kitchenham dataset	92
Table 8 MAE evaluation of potential Bellwethers sampled by SSPM, SysSam and RandSam against the growing portfolio benchmark in the ISBSG dataset.....	94
Table 9 MAE evaluation of potential Bellwethers sampled by SSPM, SysSam and RandSam against the growing portfolio benchmark in the Maxwell dataset.....	96
Table 10 Yuen’s test and Cliff’s δ effect size for pairwise differences between the weighted/ unweighted Bellwether (SSPM) and growing portfolio	99
Table 11 Statistical significant differences between prediction results of the Bellwether (BMW) and the growing portfolio (GP) relative to p_i	104
Table 12 Software effort classification based on density-quantile function.....	116
Table 13 Performance evaluation of learners using MAE and LSD	117
Table 14 Classification performance of predicted effort wrt to number of losses.....	118
Table 15 Actual vs. Predicted effort and their respective effort classes	120
Table 16 Statistical significant differences of predicted effort across learners	120

List of Figures

Figure 1 General Overview of the Bellwether Method for sampling Bellwether Moving Windows	18
Figure 2 Weighting functions	19
Figure 3 Kernel density plot showing the distribution of data before and after trimming + log transformation	67
Figure 4 Prediction of the pi hold-out projects by the Bellwether (BMW) and the growing portfolio (GP) in the Desharnais dataset.	102
Figure 5 Prediction of the pi hold-out projects by the Bellwether (BMW) and the growing portfolio (GP) in the Kitchenham dataset.	102
Figure 6 Prediction of the pi hold-out projects by the Bellwether (BMW) and the growing portfolio (GP) in the ISBSG dataset.	103
Figure 7 Prediction of the pi hold-out projects by the Bellwether (BMW) and the growing portfolio (GP) in the Maxwell dataset.....	103
Figure 8 Effort Estimation of projects without a classification scheme	110
Figure 9 Effort Estimation of projects with a classification scheme (H–high effort, M–moderate effort, L–low effort)	111

Nomenclature

A. Acronyms

ATLM	Automatically transformed linear model
DNN	Deep neural networks
BIC	Bayesian information criteria
BMW	Bellwether moving window
EMC	Ergodic Markov chain
GP	Growing portfolio
LOO	Leave-one-out validation
LASSO	Least absolute shrinkage and selection operator
MAE	Mean absolute error
MCMC	Markov chain Monte Carlo
OLS	Ordinary least squares
RandSam	Random sampling
SEE	Software effort estimation
SSPM	Stratification, sampling and prediction method
SysSam	Systematic sampling
TPM	Transition probability matrix

B. Notation

δ	Cliff's delta effect size
Ω	Sample space
f_{new}	Feature set of new projects
f_{bell}	Feature set of Bellwether projects
w_i	Projects within the i^{th} window

p_{ij}^n	Probability that state i will move to j after n s
t	Calendar years or months
α	Learning rate
γ	Momentum
D	Number of historical projects from a dataset
N	Number of projects sorted in ascending order
m	Subset of recently completed projects
q	Number of partitions
p_i	Hold-out project whose target is to be estimated

Chapter 1

Introduction

1.1. Background and Motivation

Software effort estimation (SEE) is the process of predicting the effort needed for scheduling, costing and allocating resources to meet project delivery deadlines [1][2][3][4]. SEE is a core activity in software development life cycle [5][6] specifically for scheduling and monitoring a *new* software project development. Poor SEE can jeopardise the budget allocation and negatively affect the software quality assurance of a *new* project to be developed. The key objective of SEE is to obtain a robust and accurate predictive model which has proven elusive over the past years. A successful and accurate SEE model will result in an effective scheduling, monitoring and management of the software development life cycle [7][8][6][9]. SEE plays a significant role in other software engineering domains for obtaining the most realistic predictive model for a target variable to be estimated. For example, in the defense domain, Lee *et al.* [10] proposed MND-SCEMP to improve on the prediction accuracy of resource requirements during development of weapon systems. In the context of technical debt analysis, Mensah *et al.* [11][12] introduced a novel effort prediction model for estimating the rework effort of prioritized self-admitted technical debt tasks [13][14]. Similarly, in bug localization, Zhao *et al.* [15] introduced an effort estimation metric for code inspection to assess the practical

usefulness of *learning-to-rank* [16]. SEE has also been considered by other researchers [17][18][19][20][21] in software defect prediction domain to improve on the quality of effort-aware defect prediction models.

According to Menzies *et al.* [22], predicting software effort is of key importance since it minimises underestimation and overestimation of software effort that can result in serious problems to a company [23]. Irrespective of how complex or simple an estimation model is, the prediction accuracy of a model depends on the sampled training and validation sets [24][25] as well as the relative consistency of the nature of dataset considered. Thus, there is the need for considering relevant selection of the training and validation sets for building a robust and relatively accurate prediction model. The existence and sampling of exemplary projects (considered as the training set and) referred to as *Bellwether* has successfully been used to improve prediction accuracy in some aspects of software engineering. Krishna *et al.* [26] proved the existence of *Bellwether* in the context of software defect prediction by making an implicit assumption that, *given N cases of defect prediction projects, there exists exemplary project data that can yield a relatively high prediction accuracy on the remaining project cases*. They considered their approach of finding the *Bellwether* as a simple transfer learning method [27] whereby using exemplary projects yielded improved prediction accuracy for new projects to be estimated. Similarly, in previous studies by Chen *et al.* [28][29], a subset of sales of an item was shown to provide a successful basis for the prediction of the item's annual sales. Chen *et al.* [28][29] obtained the *Bellwether* to be used for predicting annual sales of a new item by first sampling weekly and existing sales data based on a low data-collection

cost. The weekly sales data was considered as the *Bellwether* for the prediction of the annual sales of a new item. The existence of such exemplary projects from a given repository to be considered as the *Bellwether* for making predictions of new project cases is referred to as the *Bellwether effect* [26].

Based on Markov chain Monte Carlo (MCMC) methodology [30], the outcome of an event (new project) in an experiment depends on the previous experiment (recently completed projects with defined transition probabilities). In this study, it is argued with theoretical and empirical analysis that MCMC modelling based on a novel step-by-step method can be used to sample relevant and recently completed exemplary projects for a relatively higher prediction accuracy. The subset of exemplary and recently completed projects is referred to as the *Bellwether moving window* [31][32][33].

The new focus has been on selecting the most relevant, reliable and recent data samples from historical data repositories and their inclusion strategies for predictive modelling [34][35][36][37]. According to Lokan and Mendes [38][39], selection of recently completed training set (*moving window*) from a set of chronologically arranged projects for model building can further improve prediction accuracy in SEE. The *moving window* theory is based on the assumption that recent projects are likely to share similar characteristics with new projects. Recent studies [31][34][40] support the theory by Lokan and Mendes that the use of a *moving window* as the training set improves prediction accuracy of SEE models. The *moving window* approach has recently been supplemented with weighting functions which have been known to *further improve prediction accuracy of SEE models*. This theory was postulated by Amasaki and Lokan [34][41] and was

empirically confirmed to improve prediction accuracy when using large-sized moving windows. These previous studies found SEE models built using recently completed projects (*moving window*) superior to SEE models that use all available historical projects (commonly referred to as the *growing portfolio*).

Irrespective of the significant improvement of using weighted moving windows, challenges still exist in determining how large and how old the window should be prior to predicting new project cases. The *window size* simply refers to the number of most recently completed projects appropriate for building the SEE model. The *window age* refers to the elapsed time or duration span of projects (forming the window) that have existed for not more than t calendar years or months [35].

Previous studies [37][34][41] considered using window sizes such as 20, 30, 40 ... 120 projects to build the prediction model. After assessing the prediction accuracy, they selected the best window size to predict the software effort (target) of the new projects. Similarly, projects forming different window ages (such as 1 year, 1.5 years, 2 years, 2.5 years, ...) were considered by previous studies [7][9] for building the prediction model. These arbitrary sizing and aging parameters used in moving windows significantly affect prediction accuracy irrespective of the weighting functions used [37][34]. This is mainly due to the variations in the sample sizes, project ages and the different chronological datasets used in modelling. A chronological dataset is composed of projects from a given repository with their respective development start and/or completion dates. Over the years, most researchers used chronological datasets such as the Kitchenham dataset [42], the ISBSG dataset [36] and the Finnish dataset [36] for predicting the

software effort. Recent studies [40][37] that used the ISBSG dataset found moving windows to improve prediction accuracy. On the other hand, in the use of the Finnish dataset, the moving windows assumption proved reliable in a recent study by Amasaki and Lokan [34] but unreliable in a recent study by Lokan and Mendes [36]. Thus, negative result was recorded by Lokan and Mendes [36] in using moving windows for SEE as compared to the *growing portfolio* (wrt the Finnish dataset). Even though there is a challenge in the use of moving windows based on previous study's [36] findings, it is argued in this study that effective sampling of a subset of exemplary projects can result in improved prediction accuracy as proven in previous studies [31][33]. Thus, more studies with different datasets and different modelling techniques are needed to further investigate the utility of moving windows and its generalisation in SEE. That is, *effective sampling of exemplary projects (BMW) with defined window size and age can be modelled based on Markov chains to further improve prediction accuracy*. This paper adopts the concept of *Bellwether effect* to sample moving windows for predicting the targets of *new* project cases.

Aside of the aging and sizing constraints of the *moving window*, challenges exist in determining the most relevant project cases to be considered as the training and validation sets since irrelevant project cases can affect prediction accuracy in SEE modelling. According to Keung *et al.* [8], the elusiveness of accurate predictive models for software effort has been a challenge for researchers, software engineers, practitioners, project managers and developers, who need an appropriate model and reliable training and validation sets to make relatively effective predictions [43]. Despite the numerous

prediction models [5][7][44][45], the SEE research community still faces difficulties in making reliable and accurate predictions due to the following factors as highlighted by Mensah *et al.* [3]: (1) variations within SEE datasets, (2) the methodological procedures used for generating the training and validation sets, (3) the type of SEE models used and (4) the performance evaluation measures used. These factors can affect the relevancy constraint of the *moving window* modelling approach resulting in poor *conclusion stability* of prediction results [8][3]. Thus, there is the need to consider appropriate project data for training and validation of SEE models in order to improve on the *conclusion instability* issue as reported in a previous study by Mensah *et al.* [3]. *Conclusion instability* refers to the variations in estimated results for a given dataset from different prediction models [8][43]. Previous studies [8][43][5][46][36][34][47][48] have shown that the *conclusion instability* of prediction estimates is significantly affected by the heterogeneous nature of projects emerging from different organisations, variations within SEE datasets and the methodological approach used to sample training and validation sets.

The challenge of this study is to theoretically and empirically prove the utility of *Bellwethers* with defined size and age parameters as well as appropriate selection of relevant project cases for improved prediction accuracy in SEE. The study makes use of chronological datasets from recognised data repositories (*i.e.*, ISBSG and PROMISE [49]) for the empirical analysis. The study addresses the window sizing, aging, weighting and relevancy constraints by introducing three *Bellwether methods* – (1) stratification, sampling and prediction method (*SSPM*), (2) systematic sampling (*SysSam*) method and (3) random sampling (*RandSam*) method.

This study also adopts the *duplex output modelling* approach introduced by Mensah *et al.* [3] to classify the continuous dependent variable (*i.e.*, *software effort*) into discretised classes (*i.e.*, *effort class*). The classification of software effort can further provide a benchmark to help identify the level of effort (*high*, *low* and *moderate*) expended during a software project development. According to Bakir *et al.* [50], estimating the classes of effort instead of the actual effort values minimises overestimation and underestimation. Mensah *et al.* [3] define *software effort classification* as the process of discretising the software effort of the training set into classes that will allow self-guided interpretation of the estimated results. Thus, for all training sets, they [3] first discretise the continuous dependent variable (*software effort*) into three classes based on Eubank's density-quantile function [51]. The classification of software effort is an auxiliary discretised dependent variable (*software effort class*) in addition to the main continuous dependent variable (*software effort*). These two dependent variables, referred to as the *duplex variables*, are used in addition to the independent variables in each dataset to set up the predictive models. The prediction of the *software effort* and its respective *effort class* (regarded as the *duplex* output variables) is referred to as *duplex output software effort estimation* [3].

1.2. Problem Statement

Previous studies [34][36][35][37] have shown that the prediction accuracy of software effort estimation (SEE) models is affected by the sizing and aging constraints of moving windows. That is, for a given dataset, different sizing and aging parameters of the moving

window significantly affect prediction accuracy in SEE modelling [31][32][33]. The application of different weighting functions (namely Triangular, Epanechnikov, Gaussian and Rectangular) affect the prediction accuracy as discussed in previous studies [34][37][41]. Sampling of relevant and exemplary chronological project cases from a given historical dataset (*growing portfolio*) to be used for the training and validation sets can further improve prediction accuracy of SEE models [31]. Thus, given N chronological projects, the prediction accuracy is affected by the following challenges:

- relevancy constraint of project cases
- different window sizing constraint
- different window aging constraint
- different weighting functions applied

The challenge is how to sample from the search space (N) the best size and age of recently completed and exemplary projects (*Bellwether moving window*) that can improve the prediction accuracy of *new* project(s) to be estimated. Thus, there is no effective stratification and sampling framework to select relevant and recently completed chronological projects for relative effective prediction accuracy. This study seeks to address the four aforementioned challenges in the context of software effort estimation modelling based on the concept of *Bellwether* [26][28] and Markov chain Monte Carlo [30].

1.3. Scope of Study

This study is in the domain of software effort estimation (SEE). The scope of the study covers the sampling of recently completed and exemplary project cases described as the *Bellwether moving windows* for building SEE models. Specifically, the study addresses the window sizing, aging and relevancy constraints of using *Bellwether moving windows* for SEE modelling. The sampling procedure covers chronological projects (*i.e.*, projects with start and/or completion dates) sampled from single and cross organisations sourced from the ISBSG and the PROMISE repositories. Here, projects from organisations in the PROMISE repository [49] and the International Software Benchmarking Standards Group (ISBSG) repository¹ are utilized for the empirical analyses of this study. The study covers the application of deep learning (*i.e.*, deep neural networks [52]) and machine learning algorithms (*i.e.*, ATLM, ElasticNet, *etc*), robust statistical tests and probabilistic methods (*i.e.*, Yuen's test, Brunner's test, Cliff's δ effect size and Markov chains). It also makes use of robust error evaluation measures (*i.e.*, MAE and LSD) as well as effective preprocessing techniques (*i.e.*, *log-transform*, Kernel density plots, trimmed means, DIFFTS and Cook's distance) on the utilized datasets to obtain robust and easy to implement predictive models. This study incorporates two weighting functions, namely Biweight and Triweight in addition to the existing weighting functions [34] (*i.e.*, Triangular, Epanechnikov, Gaussian and Rectangular) to further examine their effect on using moving windows for SEE modelling.

¹ <http://www.isbsg.org>

1.4. Research Objectives

1.4.1. Global Objective

The main objective of this study is to investigate the existence of exemplary projects (*Bellwether effect*) in SEE focusing on chronological projects and provide a framework for sampling *Bellwether moving windows* for improved prediction accuracy.

1.4.2. Specific Objectives

The study addresses the following three specific objectives:

- To investigate the existence of exemplary and recently completed projects (*Bellwether moving windows*) in the context of software effort estimation.
- To stratify N chronological projects and sample relevant project cases (*Bellwether moving windows*) for SEE modelling based on defined and proven theoretical postulations.
- To compute the window age and size of the *Bellwether moving window* based on Markov chain Monte Carlo methodology.
- To investigate whether the use of *Bellwether moving windows* together with an effort classification scheme can minimise the *conclusion instability* problem across learners.

1.5. Research Contribution

This study makes five contributions:

- theoretically and empirically prove the existence of *Bellwether moving windows* with defined window size and age for software effort estimation.
- investigate the significance of *Bellwether effect* on software effort estimation accuracy.
- introduce three *Bellwether methods* for sampling *Bellwethers* for improved prediction accuracy.
- provide a guideline for sampling *Bellwether moving windows* for its practical use in predictive modelling.
- introduce the concept of *duplex output predictive modelling* using *Bellwethers*.

1.6. Organization of Study

In chapter 2, a set of preliminaries with regard to understanding specific terminologies (such as the *Bellwether* concept, moving windows and *growing portfolio*) is discussed. Chapter 2 also presents a summary of related works pertaining to using *Bellwethers*, moving windows and *growing portfolio* for predictive modelling. This chapter concludes with clarity of differences between previous studies and the current study at hand.

Chapter 3 details the empirical framework and methodological procedure employed in the empirical analysis of this dissertation. Specifically, a detailed description of the chronological datasets as well as a set of preprocessing techniques are discussed. The chapter also introduces 6 postulations together with their theoretical proofs for investigating the use of *Bellwether moving windows* for software effort estimation. Thus,

in order to investigate if there exists such exemplary and recently completed project data (*i.e.*, *Bellwether moving window*), there is the need to first establish a theoretical foundation prior to empirical analysis of the benchmarked datasets. This dissertation has established a theoretical foundation that once there exists such *Bellwether moving window* in a given dataset, then it can yield a relatively improved prediction accuracy as compared to using the *growing portfolio*. Again, the theoretical foundation has established solid foundation to address the sizing and aging constraints of using moving windows for predictive modelling. The proposed *Bellwether methods*, namely *SSPM*, *SysSam* and *RandSam* are also discussed. It should be noted that these three *Bellwether methods* were tested on chronological datasets since such *Bellwether moving windows* can be obtained from projects known start dates and/or completion dates. Five assumptions are presented for researchers and software engineers to take into consideration when making use of the three proposed *Bellwether methods*. Finally, the experimental setup and performance evaluation measures are discussed.

Chapter 4 presents the experimental results and discussion from the empirical analysis of the existence of *Bellwethers* in the studied datasets. Evaluation measures such as the mean absolute error (MAE), Cliff's δ effect size, Yuen's test, Brunner's method, etc are used to assess the performance of the *Bellwether moving window* against the *growing portfolio* benchmark from each of the studied datasets. Finally, the chapter introduces a useful guideline for incorporating *Bellwether moving windows* for prediction purposes. This will assist researchers and software engineers to effectively sample *Bellwethers* for the training and validation needs of predictive modelling.

Chapter 5 of the dissertation is concerned with addressing the *conclusion instability* problem in software effort estimation modelling. Thus, this study hypothesizes that discretizing the software effort of a given dataset based on Eubank's optimal spacing theory can ameliorate the *conclusion instability* problem irrespective of the prediction model or learner used. Thus, the *duplex output model* is introduced to estimate the software effort as well as the respective class (*high*, *low* and *moderate*) in which it belongs. That is, the dissertation argues that estimating the level of effort (*i.e.*, *effort class*) in addition to the software effort (*i.e.*, *target*) during setting up prediction models will assist researchers and software engineers to know the effort expended in development. This was done by making use of multiple learners, multiple evaluation measures and variant of datasets.

Finally, chapter 6 provides a summary of this dissertation followed by a concluding remark of the results obtained from the empirical analyses. It also discusses the threats that can possibly affect validity of the empirical results. Specifically, external validity, internal validity, construct validity and conclusion validity are discussed. Finally, future directions arising from this dissertation are discussed.

Chapter 2

Literature Review

This chapter presents a set of preliminaries in order to understand this dissertation. The chapter also presents review of related works in relation to the use of *Bellwethers*, *moving windows* and *growing portfolio* for prediction modelling. The chapter concludes with a set of differences between existing studies and the current dissertation.

2.1. Preliminaries

2.1.1. The Concept of Bellwether

According to the Oxford dictionary, *Bellwether* refers to a sheep with a bell around the neck that leads a flock. Simply put, a *Bellwether* is one that takes the lead or initiates as defined by the Merriam-Webster dictionary. The *Bellwether* concept in the domain of software defect prediction is defined by Krishna *et al.* [26] as a simple transfer learning approach consisting mainly of the *Bellwether effect* and the *Bellwether method*. In the domain of software effort estimation (SEE), Mensah *et al.* [31][33] investigated the existence of *Bellwether* in chronological datasets and defined the notion of a *Bellwether moving window*.

- *The Bellwether effect states that, given a set of projects from a given repository, there exists a subset of exemplary projects that can form the Bellwether and such projects considered as the training set yield a relatively high prediction accuracy of the remaining projects. This Bellwether can be considered for predicting the target of a new project.*
- *The Bellwether method uses a step-by-step heuristic approach to search for that particular Bellwether from the given dataset of historical projects and applies it to a new project whose target is to be estimated. Chen et al. [28][29] define this approach as Bellwether analysis.*
- *The Bellwether moving window refers to exemplary and recently completed projects that yield improved prediction accuracy of new project(s) to be estimated [31]. It should be noted that projects forming the Bellwether moving window have their completion dates prior to the starting date of the new project(s).*

Within computer science the concept of *Bellwether* was first considered by Chen et al. [28][29] in the context of Online Analytical Processing (OLAP) queries. Their intent was to find a small subset of queries that could be used to make accurate prediction of the target of a new query. Krishna et al. [26] in the domain of defect prediction implemented the concept to find exemplary projects (referred to as *Bellwether*) from a set of non-chronological datasets to successfully make predictions on the remaining projects. Each project was considered as a potential *Bellwether* in each iteration of making successful prediction of the remaining projects until the project data with the best prediction accuracy was obtained. That particular project was considered as the *Bellwether* to make successful

prediction of defects in a *new* project (set as a hold-out). Results from these studies [26][28][29] show that *Bellwethers* are not rare and can be used for building prediction models for new project cases with relatively higher prediction accuracy than would be achieved otherwise.

This dissertation investigates the feasibility of using *Bellwethers* as moving windows for building effective SEE models. The dissertation presents a general overview of how the selection of exemplary projects to be used as the *Bellwether moving window* (BMW) for predicting the targets (software effort) of *new* project cases (set as hold-out) in Fig. 1. The study first sorts a set of N projects chronologically and apply a statistical stratification technique to obtain only the recently completed projects. The study then checks for the existence of *Bellwethers* based on defined *Bellwether methods* as elaborated in chapter 3. The moving window which yields a relatively higher prediction accuracy is selected and defined as the BMW. The potential BMW, w_i is used to predict the effort of projects in each of the remaining windows, w_j in $N \forall i \neq j$. If the prediction accuracy of projects within the remaining windows, w_j is relatively low, then the projects within the potential BMW, w_i is adjusted by reducing or adding more project(s) sequentially until the best relative prediction accuracy is obtained. In each iteration process of updating the potential window, a transition probability matrix is generated and its stationarity verified by computing the respective ergodic Markov chain. Further elaboration of the *Bellwether method* for sampling the BMW (depicted in Fig. 1) is detailed in chapter 3.

2.1.2. Moving Window and Growing Portfolio

The development of SEE models using recently completed projects with completion dates less than the start date of a *new* project (regarded as the *moving window*) has been the focus for researchers [37][34][35][40]. The performance accuracies of most of these models were superior to the traditional SEE models. This approach is different from the existing approach of using the entire collection of historical projects as the training set for setting up predictive models [7][5][54][23][43].

Growing portfolio is the term for using all historical projects (with completion dates less than the start date of a *new* or hold-out project) for building a predictive model. Weighting functions have been proven to further enhance prediction accuracy in a recent replication study by Amasaki and Lokan [34]. They applied four functions, namely Triangular, Epanechnikov, Gaussian and Rectangular (or uniform) as shown in Fig. 2 (noting that application of the Rectangular function is equivalent to using unweighted moving window [34]). This current study incorporates the weighting/unweighting functions depicted in Fig. 2 to both the moving window and the *growing portfolio*. We also explore the effect of two additional weighting functions, namely the Biweight and Triweight [55][56]. Here, projects are weighted in a non-decreasing order following a similar approach to that used by Amasaki and Lokan [34] whereby a more recently completed project has heavier weight. From Fig. 2, u is defined as follows: $u = x_i/n$, $i = 1, 2, \dots, n$. Here, x_i represents project i ranked in a non-decreasing order of completion date in a set of n projects. That is, u decreases as older projects are assigned with lower ranks.

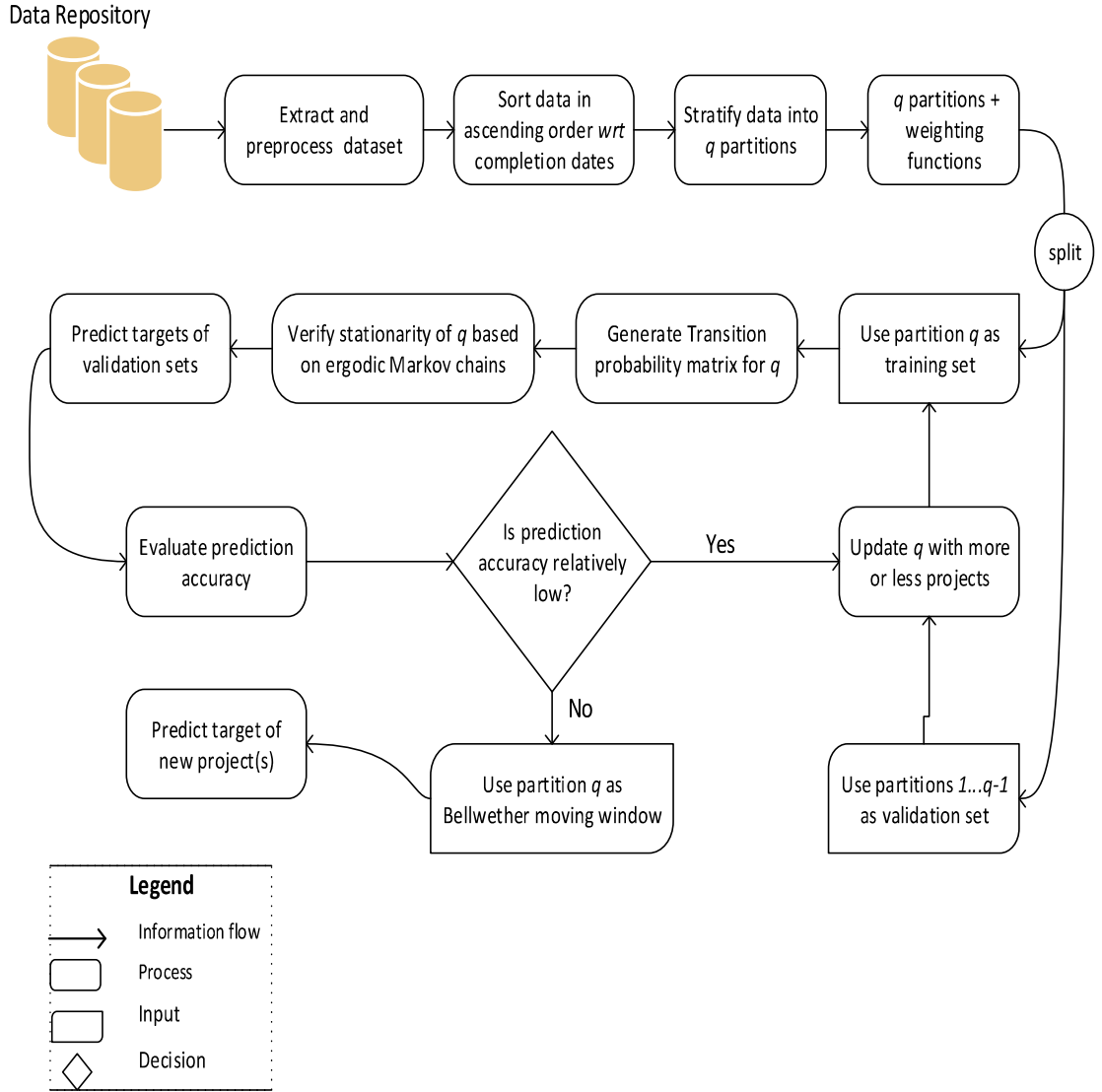


Figure 1 General Overview of the Bellwether Method for sampling Bellwether Moving Windows

The rectangular or uniform weighting function is the simplest of the weighting functions [34]. It takes the value of 1 if the absolute value of u is less than 1. A shortcoming of the rectangular function is that the estimated density function is not smooth and may have some jumps. Alternatively, the Epanechnikov weighting function provides smooth density functions. The Rectangular, Epanechnikov, Triangular, Biweight and Triweight

functions have finite support, and hence points beyond u units will not add any contribution to the density at such points. On the other way, the Gaussian function does not support restrictions and hence it takes into consideration all the data values [56].

Weighting function	Formula
<i>Rectangular (Uniform)</i>	$w(u) = 1, u < 1$
<i>Triangular</i>	$w(u) = 1 - u , u < 1$
<i>Epanechnikov</i>	$w(u) = 1 - u^2, u < 1$
<i>Gaussian</i>	$w(u) = \exp(-(2.5u)^2/2), u < \infty$
<i>Biweight</i>	$w(u) = 15/16(1 - u^2)^2, u \leq 1$
<i>Triweight</i>	$w(u) = 35/32(1 - u^2)^3, u \leq 1$

Figure 2 Weighting functions

To the best of our knowledge, this is the first study to consider these two additional weighting functions to investigate prediction accuracy of *moving windows* and *growing portfolio* in SEE. We evaluate the prediction accuracy performance using the BMW and the *growing portfolio* in constructing SEE models. This is done to investigate the feasibility of using a relevant and recently completed subset of chronological projects (*Bellwether moving window*) in building a robust and accurate predictive model.

2.1.3. Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a probabilistic technique that seeks to solve the problem of sampling by exploring a given sample space through the construction of an ergodic Markov chain (EMC) whose limiting distribution is the target distribution [30]. MCMC utilises Markov chains to effectively simulate a random variable X whose future state is independent of past states given the present state. At any time t , we can define a Markov process for a present state (in this case the effort of a new project) based on previous history (*i.e.*, historical projects) by finding the transition probabilities from one project state to the previous state with respect to their ages. Thus, in order to make a

prediction of a future state (in our case effort) in a Markov process, there is no need of knowing the entire past events (projects) but knowing the current event can be helpful [30][57]. According to Cook and Wolf [57], a Markov process determines what is the most likely event to occur from a given sequence of events. If we consider a variable P as a transition probability matrix of a Markov chain, then each p_{ij}^n element of P^n is the probability that the Markov chain beginning from state i will transition to state j after n steps. The transition probability matrix of a given sample with respective project ages can be generated as the probabilities of occurrences from one state to another. MCMC has been applied and proven successful in different software engineering domains such as software testing [58], image processing [59], human vision [60], robotics [61], election results [62], biomedical engineering [63], smart homes [64] and applied mathematics [65]. In the field of image processing, a study by Bandyopadhyay [59] proposed an approach by making use of MCMC to find the relevant segmentations of data by automatically clustering the image dataset under consideration. A study by Nortey *et al.* [62] developed Markov chain models to predict election results based on immediately past election results. In their study, confidence intervals for their predicted results were achieved using bootstrap percentiles.

This paper also provides a statistical investigation on the use of MCMC methodology to obtain a potential subset of projects to be considered as the *Bellwether* (chapter 3). Let D denotes a set of projects from a given repository (sample space) with d degree of dimensions (features). Then, $X = (x_1, \dots, x_d) \in D$ can represent a sample whose limiting distribution is known and can form the *Bellwether*. Each of the data points $x_1, \dots, x_d$

of the sample is allowed to take a discrete or continuous time value which denotes the project ages already known from the given repository (D). The Markov chain is then executed a number of times until the chain converges to its limiting distribution to obtain the target sample subset. The selected sample with its respective t ages and constructed limiting distribution forms the *Bellwether* which can be used as the moving window. We define this type of moving window as the BMW.

2.1.4. Central Limit Theorem and Law of Large Numbers

MCMC relies on the fact that the limiting properties of ergodic Markov chains possess some similarities of independent and identically distributed (*iid*) sequences [30]. Hence, in order to prove the existence of a *Bellwether* sample from a state space, the central limit theorem and the law of large numbers should hold. The central limit theorem states that as the sample size turns to infinity, the sample mean will follow a normal distribution with mean, μ and variance, σ^2 [66]. The law of large numbers states that as the sample size increases, its mean, $\bar{x}$ will be approximately the same as the population mean, μ [67]. The larger the sample size ($n > 30$), the smaller the sample variance, s^2 and turns to follow a normal distribution [68]. Although Markov chains are not independent sequences, the law of large numbers and central limit theorem are necessary for their independence. Thus, for ergodic Markov chains, successive transitions between visits to the same states are independent [30]. Thus, a moving window sample, X_i can be used as a *Bellwether* with defined window size and window age for the prediction of *new* project cases.

2.1.5. Statistical Measures

In order to investigate the statistical significance of *Bellwethers* within the studied datasets, we employ the following robust and non-parametric statistical measures recommended by Kitchenham *et al.* [53]:

1) Brunner's test

Brunner *et al.* [69] developed a rank version of the Analysis of Variance (ANOVA) method for testing the statistical significance of treatment effects. This method makes use of midranks for duplicate observations within the same treatment group. Thus, when there are two or more duplicates within a group, the midranks play a key role by allocating the mean of the duplicates to each observation. The Brunner's method is relatively reliable and robust to complex factorial designs with unequal variances across treatment groups [53]. Tian and Wilcox [70] found in a recent study that Brunner's ANOVA-type method is more advantageous in terms of Type I error and power as compared to Agresti-Pendergast method [70]. Brunner's test statistic is recently used in a study by Phannachitta *et al.* [44] for testing the null hypothesis that there is no statistical significance of the difference between predicted effort from two Analogy Based Estimation (ABE) variants (i.e., $p > 0.05$). We consider this test statistic because of its ability to perform relatively better when considering small sample sizes for assessing the statistical significance of treatment effects [69][70][53]. In this study, we implement the Brunner's test statistic to test the null hypothesis that *there is no statistical significant differences across the six weighted/unweighted BMW and growing portfolio*. Statistical test is performed at 5% asymptotic significance level.

2) *Yuen's test*

Given two vectors of data samples with unequal sizes and unequal variances, the Yuen's test statistic provides an avenue for a robust pairwise testing based on their trimmed means [53][71]. This robust test statistic was proposed by Yuen [72]. If no trimming is applied on the data, then the test reduces to the Welch's t -test statistic [73]. Yuen's test uses the trimmed mean as a measure of central location to test for the significant differences between two samples. The test statistic can also be extended to cater for multiple group comparison (i.e., more than two treatment groups). Yuen's test works best when addressing the statistical significant differences between groups of different sizes. Hence, we use the Yuen's test to assess the statistical significant difference between the *growing portfolio* and the BMW. Note that the BMW obtained by the *Bellwether method* [32] is a subset of the *growing portfolio* (all historical data considered as training set). Thus, the sample sizes of the two treatment groups are different. Here, we perform statistical test at 5% asymptotic significance level.

3) *Cliff's δ effect size*

A robust effect size method as recommended by Kitchenham *et al.* [53], namely the Cliff's delta (δ) effect size [74] is used to find the practical significance across the BMW and the *growing portfolio*. The Cliff's δ effect size is chosen since it yields an effective computation measure irrespective of both the experimental and control groups having different sample sizes. Again, it is not affected by outliers and does not assume the sampling data to follow any distribution [75]. The rationale behind the Cliff's δ effect size is that, given two groups of observations without necessarily following the same

distribution [53], the Cliff's δ effect size is able to determine the amount of overlap that exists between these two groups. This non-parametric effect size method was also recommended by Arcuri and Briand [76] for empirical software engineering data analysis. Effect size computation is very important since it provides an appropriate and reliable measure for the practical usefulness of an experimental effect regardless of the *p-value* showing the statistical significance for a test statistic [53][76][77][78]. The Cliff's δ effect size is defined in (1) whereby each observation y_i from the BMW is compared to each observation y_j in the *growing portfolio*. As defined in (1), we count the total number of ($y_i > y_j$) and ($y_i < y_j$), compute the difference and divide by the product of the sample sizes for the prediction observations from the BMW (m) and the *growing portfolio* (n).

$$\delta = \frac{\text{sum}(y_i > y_j) - \text{sum}(y_i < y_j)}{nm} \quad (1)$$

The effect size or practical significance is interpreted based on the magnitude thresholds by Kampenes *et al.* [79] as follows: negligible effect size ($\delta < 0.112$), small effect size ($0.112 \leq \delta < 0.276$), medium effect size ($0.276 \leq \delta < 0.427$) and large effect size ($\delta \geq 0.427$). An effect size threshold of medium to large is chosen since results are misleading when the effect size is relatively small or negligible [22].

4) Kernel density plot

One of the most reliable and effective means of visualising the distribution of data is the use of Kernel density plot [53]. Even though histogram plays an important role in visualising the distribution of a data, variation can occur in the graph depending on the number of bins used [80]. According to Tukey [80], the aforementioned challenge can be

addressed using the kernel density estimation. This form of estimation addresses the probability density function of a feature or variable by smoothing the histogram. Kernel density estimation employs the kernel weighting function [81] to ensure that the total area within the bins or total area under the curve equals to one. Thus, the estimator creates a *bump* on each data point and computes the total *bumps* using the function defined in (2).

$$p(x) = \frac{1}{nh} \sum K\left(\frac{x - X_i}{h}\right) \quad (2)$$

where x is the data point for the density to be estimated; X_i is the midpoint of the interval; K is the *bump* or kernel density function; n is the sample size and h is the bandwidth [80]. It should be noted that we examine the effect of data transformation to further investigate if it improves the distribution of the data [53].

2.2. Review on Bellwethers

Although the *Bellwether* concept has been discussed in other domains of software engineering, it should be noted that this dissertation is first to investigate the use of *Bellwethers* for software effort estimation as discussed in recent studies by Mensah *et al.* [31][32][33]. Thus, *Bellwethers* have been discussed in previous studies by Chen *et al.* [29][28] in the context of Online Analytical Processing (OLAP) queries and in software defect prediction by Krishna *et al.* [26].

In 2006, a study by Chen *et al.* [29] was the first to introduce the concept of *Bellwethers* in terms of cube space regions in OLAP as well as machine learning and data mining. The rationale behind using *Bellwethers* was that OLAP queries can be used to generate class labels for prediction tasks, which creates the possibility of developing a

prediction model without involving human labor for data labeling. In their study, they showed the feasibility of the aforementioned rationale of using *Bellwethers* and demonstrated its value as well as developed efficient algorithms for *Bellwether analysis* in OLAP. In formulating their research problem, Chen *et al.* [29] first introduced a motivating example prior to defining the problem of *Bellwether analysis*. Thus, given that a company wants to make prediction of the annual profit of a new item after selling the item worldwide for one year. However, if this company can make accurate prediction this target value using prior features such as regional profit obtained in shorter duration such as the first week of sales in a smaller geographical location such as Wisconsin, then it can quickly make adjustment to either maximize profit or minimize. Thus, the first week sales in a smaller location (*i.e.*, Wisconsin) is an example of a *Bellwether* region defined by Chen *et al.* [29]. Their ultimate goal was to find such *Bellwether* regions as denoted in the motivational example as by a pair of specific time interval and location.

To find such *Bellwether* regions, the company can exploit its historical sales database containing transactions of items at specific times and locations containing quantities and profits earned from each transaction. By making use a simple data mining approach, the target value (*i.e.*, the first year worldwide profit) for each historical item in the database can be obtained. Thus, training and validation sets can be created by associating the features (*i.e.*, independent variables) of each historical item with its target value (*i.e.*, dependent variable). A predictive model such as a simple linear regression model can then be setup using the training and validation sets. This can then be used to predict the target value of a new item based on its predictive or independent features.

Chen *et al.* [29] considered *Bellwether analysis* necessary if such predictive model did not yield accurate predictions on the validation sets. However, since the features associated with the historical items were usually not sufficiently predictive, the prediction model yielded less significant improvements and hence, the need for *Bellwether analysis* in their investigation.

In order to improve the prediction accuracy, it is very necessary to add more informative features to the predictive modelling approach. However, obtaining such features for a new item incurs additional cost. If the company is able to sell the new item worldwide for a year, then they can be able to determine the exact worldwide profit and there will be no need for prediction even though that will incur a high cost. According Chen *et al.* [29], the objective of *Bellwether analysis* is to determine a cost-effective “region”, such that using new features collected from that region can best improve the accuracy of the model. Thus, a candidate region is defined as combination of a duration in the time dimension and a geographical place in the location dimension.

Chen *et al.* [29] define their *Bellwether* problem as follows: Given query-generated features over different regions, they seek to find a region with a low cost such that a highly predictive model can be built using features generated from that particular region. Data in the historical database is then used to find such a region and its corresponding predictive model. Using queries, they created a training set for each region, in which each training example represented an item containing the following: (1) the item-table features, (2) the query generated regional features (*e.g.*, Regional Profit) from the given region for that item and (3) the target value of that item. They then built a predictive

model for each region using the training set from that region, and evaluated the prediction accuracy. The region that had the best accuracy with cost under a specified budget was regarded as the *Bellwether* region. Thus, for a new item, they collected data from the *Bellwether* region at a cost within the given budget, and assumed the predictive model that used features generated from that region to have the best prediction accuracy over all other regions for which they could collect data at a cost under the budget.

Chen *et al.* [29] first used the iceberg cube techniques [82][83], Naïve Bellwether tree, Rain forest Bellwether tree and the Bellwether cube algorithms to search and obtain all candidate regions to be considered as the training set. They used the ordinary least squares (OLS) linear regression as the predictive model based on the 10-fold cross validation approach. The root mean squared error (RMSE) was used as the error evaluation measure. Two real datasets, namely the mail order dataset and the books store dataset were used for their empirical analysis. The mail order dataset contains a real catalog of transaction records of a US company in 1996. It contains 4,591,581 transactions and 1,515 items. The book store dataset contains 900,000 transactions of about 43,000 books sold by a large book store US company in 2004.

Results from their analysis of the mail order dataset showed that there was improved prediction performance for a budget of 10 to 30 in order to identify a feasible *Bellwether* region. With regard to the book store dataset, they were not able to identify a feasible *Bellwether* region since the RMSE of the bellwether model converged at budget 200. They reported the following plausible reasons for not being able to identify a clear *Bellwether* region: (1) the dataset was fairly small since it did not contain all transaction

details of all stores, and (2) there was actually no unique bellwether region at each budget point. Thus, Chen *et al.* [29] were able to show that *Bellwether* region exists in one of the studied datasets namely, the mail order dataset.

A follow-up study was done by Chen *et al.* [28] in 2009 whereby they systematically studied the concept of *Bellwether analysis* introduced in their previous study [29]. Thus, they intensively demonstrated the key idea that for a number of practically motivated problems, OLAP queries can be used to define the features and target values (or class labels) for prediction tasks, which opens the possibility of developing a prediction system without expensive human labor of data preparation. This was done by making use of scalable algorithms on large datasets (that cannot fit in main memory) to empirically perform the *Bellwether analysis*. Through a series of extensive empirical experiments, they showed that *Bellwethers* do exist in real-world datasets, and that their computational techniques achieve good efficiency on large datasets. Here, three *Bellwether* problems were defined by Chen *et al.* [28], namely *Basic Bellwether problem*, *Subset Bellwether problem* and *Combinatorial Bellwether problem*.

(1) *Basic Bellwether Problem*: Given a set of items (products or candidates in the preceding examples), a database and database queries that define: the set of candidate regions; the features of each item in each region; the target value of each item; and the cost of collecting data from each region, find the *Bellwether region*. This is the most predictive region that has a cost within a given budget among all the candidate regions, for the set of items. The predictive power of a region is measured by building a predictive

model that uses features of each item in that region to predict the target value of the item, and evaluating the accuracy of the model.

(2) *Subset Bellwether Problem*: Different kinds (or subsets) of items may need different *Bellwether regions*. For example, the *Bellwether region* for high-tech products may be quite different from that of food products. Thus, it is usually useful to find different *Bellwether regions* for different subsets of items. Now, the problem is essentially how to create subsets of items and find the *Bellwether region* for each created subset, so that one can make the best predictions for new items by using the *Bellwethers* for these subsets.

(3) *Combinatorial Bellwether Problem*: In the aforementioned previous problems, a single *Bellwether region* for a set (or each created subset) of items was found. Now, instead of one region, the aim is to find a set of regions for a set (or each created subset) of items, such that the combined cost of the set of regions is within a given budget and, by using the data combined from the set of regions, one can make the best predictions for new items. For example, instead of using sales data from a single region, say, California (which includes 58 counties) to predict the worldwide sales of a new item, one may be able to use sales data from a set of smaller regions, say, only four counties, possibly in different states, to achieve better accuracy at a lower cost (assuming the cost of a region is proportional to the number of counties in it).

The main differences of the *Bellwether analysis* introduced in their previous work [29] to their follow-up study [28] are as follows: (1) the integration of database query processing into predictive modelling and (2) scalable mining techniques for large datasets that cannot fit in main memory. In addition to their original results, their further studied

the *combinatorial Bellwether problem* (which had not been studied in their previous paper), and empirically showed that a greedy method for the *combinatorial Bellwether problem* can significantly increase prediction accuracy. Furthermore, they substantially enhanced their previous experimental results by: (1) a more rigorous study of a previously analyzed mail-order dataset; (2) a new analysis of a presidential election dataset; and (3) an extensive study of the newly introduced parallel version of the *Bellwether* tree algorithm, running on a cluster of 100 machines that implement the MapReduce [84] parallel computational framework.

With regard to the mail-order dataset, Chen *et al.* [28] made use of the *combinatorial Bellwether problem* to find a *Bellwether* set of regions instead of finding a single *Bellwether* region as done in their previous study [29]. They then used the regional features from this set of regions to predict the total profit of a new item. By making use of the *Bellwether* set of regions to make predictions resulted in minimum prediction error. The *Bellwether* sets of regions usually consist of three to six relatively small states. In comparing the *combinatorial Bellwether tree* (*i.e.*, finding different *Bellwether* sets for different subsets of items) to the *basic Bellwether search* (*i.e.*, finding a single *bellwether* set for all the items), they found that there was no significant improvement on prediction accuracy.

The presidential election dataset used by Chen *et al.* [28] is sourced from the Inter-university Consortium for Political and Social Research Archive². The dataset contains the numbers of votes for 58 US presidential candidates (Democrat or Republican) in each

² <http://www.icpsr.umich.edu>

county in the US from 1876 to 1988. The goal was to predict the percentage of votes for a candidate in the entire US based on the percentage of votes for that candidate in a small region. The presidential candidates were the items (in the terminology of *Bellwether analysis*), and counties and states were the regions. Each county had a unit cost. The cost of a set of counties (or a state) was the number of counties in the set (or the state). They first looked at the *Bellwether county* and the *Bellwether state* (obtained by the *basic Bellwether search*) for these presidential candidates, and then considered finding the *Bellwether set* of counties (*i.e.*, conducted *combinatorial Bellwether analysis*) and applied the *Bellwether tree* to the presidential election dataset. They used the OLS linear regression model, in which the target attribute was the percentage of votes for a presidential candidate in the entire US and the regional feature was the percentage of votes for that candidate in a region (county or state). The *Bellwether county* found from this dataset was Peoria, IL, and the *Bellwether state* was CA. They found that the *Bellwether state* (CA) had a smaller variance than the *Bellwether county* (Peoria, IL), indicating a better predictive power. This was shown using the 10-fold cross-validation with an RMSE of $4.42 \pm 1.36\%$ for the *Bellwether county*, and $3.06 \pm 0.79\%$ for the *Bellwether state*, where the number following $\pm$ is the standard deviation. Note that in each fold of the 10-fold cross-validation, they used 90% of the candidates to find the *Bellwether region* and evaluated the RMSE on the remaining 10% of the candidates.

By using the presidential election dataset, Chen *et al.* [28] concluded on the following: (1) using *Bellwether sets* of regions (no matter the method used) to make predictions was much better than using random sets; (2) when the number of counties

(i.e., budget) increased, the RMSE decreased; (3) the *Bellwether tree* had a better accuracy than the *basic bellwether search*; however, when the number of counties was large, the differences became insignificant; (4) the *Bellwether tree* using a set of four counties had RMSE of $1.28 \pm 0.47\%$, which was much lower than the single *Bellwether county* (with RMSE $4.42 \pm 1.36\%$) and also even lower than the *Bellwether state, CA* (with RMSE $3.06 \pm 0.79\%$), which included 58 counties.

In software defect prediction, Krishna *et al.* [26] were the first to explore the *Bellwether* concept in 2016. They defined two main terms, namely the *Bellwether effect* (i.e., existence of exemplary projects) and the *Bellwether method* (i.e., the use of certain heuristics to obtain the exemplary projects). Their ultimate goal was to support software project managers who seek to achieve conclusion stability in their results with a possible approach of declaring a prior project as the *Bellwether* for improved prediction accuracy and enhanced generalization. As a result of conclusion instability, software project managers have difficulties in finding general policies of effective management. According to Krishna *et al.* [26], this conclusion instability issue hinders project managers from providing clear guidelines on many aspects including the following: (1) when modules should be refactored; (2) when some module be inspected; (3) where to focus expensive testing procedures; (4) what return-on-investment might be expected due to decreased defects after purchasing some expensive tool; etc.

Krishna *et al.* [26] noted in their study that the original motivation for transfer learning for prediction modelling in software engineering might have been misguided. Previous experimental analysis using transfer learning methodologies built defect

predictors from the data sampled from cross projects. This approach of building predictive modelling in software engineering leads to poor prediction results. This make most researchers use relevancy filters to obtain small subsets of data which are most relevant to the current prediction problem [85]. According to Krishna *et al.* [26], these relevancy filters introduced the conclusion instability problems even though they generated relatively adequate predictions. This aforementioned phenomenon formed the basis of the motivation behind Krishna *et al.*'s study. Krishna *et al.* [26] suggested based on empirical evidence that researchers should have hunted for *Bellwethers* instead of using relevancy filters [85] to obtain relevant data for predictive modelling. The *Bellwether method* introduced by Krishna *et al.* [26] does not require sophisticated data mining methods to obtain the exemplary projects. Rather constructing simple for-loops around the studied datasets can yield a simple transfer learning method to obtain a *Bellwether* data to be used to build software defect prediction models with improved prediction accuracy. Krishna *et al.* [26] noted that irrespective of the granularity of historical datasets, there exists a subset of exemplary data that can be considered as the *Bellwether* for software defect prediction.

Krishna *et al.* [26] used 120 historical datasets categorized into four communities obtained from previous transfer learning studies. The datasets measure the software defects at defect levels of granularity ranging from function-level to file-level. The four communities in which the defect prediction datasets were sourced from the following: AEEEM, Relink, Apache and NASA. AEEEM gathered by D'Ambros *et al.* [86] contains 61 metrics: 17 object- oriented metrics, 5 previous-defect metrics, 5 entropy metrics measuring code change, and 17 churn-of-source code metrics. The RELINK dataset

gathered by Wu *et al.* [87] contains 26 metrics that can help to calculate the code complexity in order to improve the quality of defect prediction. The RELINK dataset has been manually verified and corrected [26]. The Apache and NASA datasets were extracted by Krishna *et al.* [26] from the PROMISE repository. The Apache dataset contains records of known defects for a number of classes and was gathered by Jureczko *et al.* [88] using a post-release bug tracking system. The classes within the dataset are described in terms of 20 metrics. Examples of the metrics include the CK and McCabes complexity metrics. Lastly, they used five proprietary datasets from the NASA community containing similar metrics [89]. Krishna *et al.* [26] preprocessed all datasets in order for them to share the same metrics prior to sampling the *Bellwethers*.

Krishna *et al.* [26] proposed a *Bellwether method* to assist researchers first check for the existence of *Bellwethers* in the studied datasets and consider the searched *Bellwethers* as the training set for building a predictive model. They reported a *Bellwether* if that exemplary project data yielded the most accurate prediction accuracy in majority of the projects from a given community. They then used that particular *Bellwether* to make prediction of the target (*i.e.*, software defect) in the new project data. The learner or prediction model adopted by Krishna *et al.* [26] was the Random Forest [90] which is an ensemble method (endorsed by Lessmann *et al.* [91]) that builds multiple decision trees on random subsets of data. They constructed a confusion matrix and evaluated the prediction model using four performances measures obtained from the confusion matrix, namely *Accuracy*, *Recall or pd*, *Precision* and *False alarm or pf*. They also applied the

Kruskal-Wallis H -test, Scott-Knott, bootstrapping and the A12 effect size (≥ 0.6) as measures for statistical significant tests.

Results from their study show that *Bellwethers* exist in the studied defect prediction datasets. In comparing the *Bellwethers* with local data based on the same quality metrics, they found that training a software defect prediction model with a local data yielded similar prediction results as compared to training with a *Bellwether*. They benchmarked their *Bellwether method* with other transfer learners (*i.e.*, the two Burak filters [85][92]) and found that their *Bellwether method* out-performs those standard homogeneous transfer learners. Lastly, they seek to investigate the number of examples (or data) required to find *Bellwethers*. Karishna *et al.* [26] noted that they do not have a theoretical evidence to find the lower bound for such number of examples. After using 50 randomly selected defective modules and 100 non-defective modules, they concluded that *Bellwethers* can be found when the studied projects have discovered a “few dozen examples of defective modules”. Thus, no theoretical evidence was provided for the exact number of defective modules the studied projects should possess before they can be considered as a search space for *Bellwethers*. This sizing constraint is one of the issues that has been addressed by this dissertation in the domain of software effort estimation.

2.3. Review on Moving Windows

To the best of our knowledge, Kitchenham *et al.* [42] were the first to consider the selection of recently completed projects (*moving windows*) from chronologically completed projects for SEE. They made use of the Kitchenham dataset (sourced at the PROMISE repository [49]) comprising of 145 development and maintenance projects

(i.e., the *growing portfolio*) obtained from a single organisation. The Kitchenham dataset is described as a chronological dataset since the start and completion dates are recorded for all projects except for three projects with unknown completion dates. They divided the *growing portfolio* into four partitions and each partition was used as the training set for setting up the predictive modelling. Results from their regression analysis show that the size to effort estimation changed across the four partitions. They concluded that older projects should be removed from the dataset and prediction done with the remaining subset together with addition of recently completed projects. Lastly, they recommended the prediction of *new* projects with respect to the use of about 30 recently completed projects.

The application of moving windows for software effort estimation was further investigated by Lokan and Mendes [38] based on the following important question: *which of the projects can be considered as the training set to build the prediction model: should the focus be using all historical projects (i.e., growing portfolio), or a subset that is much relevant?* A key factor they considered was the project age, that is, whether it is more convenient to use a window of recent projects to setup the prediction model as compared to using an entire collection of historical projects. Thus, they considered the window of recent projects as a research problem to investigate due to the rapidly changing world we are leaving in, and as a result very old projects might not be relevant to new projects to be developed. Their study's goal was to investigate the effect of prediction accuracy in SEE using a moving window. They analyzed the ISBSG dataset and constructed an SEE model based on the linear regression. The MMRE, MAE, PRED(25) and Adjusted R^2 were used

as the performance evaluation measures of the prediction results from the SEE model. Results from their study showed that using moving windows yielded statistically significant effect on prediction accuracy as compared to using the *growing portfolio*. They found that at least 20 projects can be considered as a minimum possible threshold size when using moving windows for SEE based on the studied dataset. They considered a window size of about 75 projects representing about 1 to 2 years as the best moving window for improved accuracy for project cases sampled from the studied single organization selected from the ISBSG dataset.

A recent study by Amasaki and Lokan [34] investigated the effect of weighted moving windows on SEE accuracy using the Finnish dataset. They found that the application of different weighting functions has different effects on the SEE accuracy. They concluded that weighted moving windows yielded significant effect with comparably larger window size as compared to unweighted moving windows. Their findings confirmed results from their previous study [37] where similar observation was found using the ISBSG dataset. Amasaki and Lokan [34] considered window sizes ranging between 20 and 120 projects (*i.e.* 20, 30, 40, ..., 120) in building the prediction model based on weighted linear regression. Results show that models built with windows of sizes ranging from 20 to 30 projects yielded minimum MAE as compared to the use of *growing portfolio* in SEE modelling. The prediction accuracy diminished when using Triangular, Epanechnikov and Gaussian weighting functions for window sizes below 80 projects whilst the unweighted moving window (Rectangular) resulted in better prediction accuracy. Lastly, with window sizes more than 80 projects, prediction accuracy was

highly improved when using the three weighted moving windows (Triangular, Epanechnikov and Gaussian) as compared to using unweighted moving windows. Results from their study [34] showed that the use of moving windows sampled from the Finnish dataset can improve prediction accuracy, which was a contradiction to previous studies by Lokan and Mendes [36], and Amasaki and Lokan [93]. They [36][93] reported a negative result that companies should not discard older projects (*i.e., growing portfolio*) but rather continue to use them for the training needs of a prediction model instead of using moving windows. The plausible reason given to the contradictory result from the three-single company subsets of the Finnish dataset was due to the different characteristics of the datasets. Here, the linear regression was used for setting up the prediction model and evaluation assessment was done using the MAE (lowest value preferred) and the Adjusted R^2 (since it considers the amount of variation in effort, highest value preferred). The Holm-Bonferroni correction was applied to test for the statistical significance of differences in accuracy at 5% significance level. After the application of the Holm-Bonferroni correction, they found that it caused no significant change that in the utilized datasets, moving windows were not helpful as compared to using the *growing portfolio*. Even though there was a contradictory result that moving windows did not yield improved prediction accuracy against the *growing portfolio* benchmark in the Finnish dataset, Amasaki and Lokan [93] suggested that the heterogeneity of datasets from different industries do not necessarily account for the different results obtained.

A similar study by Amasaki and Lokan [37] examined the effects of weighted moving windows and unweighted moving windows on prediction accuracy. They also

considered projects from the ISBSG dataset (release 10) and used weighted linear regression in building the SEE model. Results from their study show that: 1) unweighted moving windows are more effective in SEE with larger window size (at least 40 projects) as compared to unweighted *growing portfolio*; 2) weighted *growing portfolio* of large size yielded better prediction accuracy as compared to unweighted *growing portfolio* with respect to MAE; 3) unweighted moving window is more effective in prediction accuracy than weighted *growing portfolio*; 4) for large window size of at least 100 projects, weighted moving window yielded better prediction accuracy than unweighted *growing portfolio* for both MAE and mean magnitude of relative error (MMRE). Here, the Gaussian weighting function was considered the best; 5) weighted moving windows of large sizes are better than weighted *growing portfolio* with respect to MAE and MMRE. The Gaussian function was considered the best weighting function in this case; and 6) weighted moving windows with large sizes (at least 80 projects) are more effective in accuracy than unweighted moving windows for MAE and MMRE. Here, no preference to the best weighting function was made in their study. Amasaki and Lokan [37] concluded that using windows of large size (more than 75 projects) improve prediction accuracy.

Lokan and Mendes [35] investigated the effect of using moving windows based on different durations and how it affects prediction accuracy. The ISBSG dataset (release 10) and the Finnish dataset were considered in their study. They found that prediction accuracy can be improved when using moving windows based on duration. Here, stepwise multivariate regression was used in building the SEE model and windows of different

durations ranging from 1 to 7 years were considered in prediction modelling. Results show that SEE models built with moving windows of about 3 years with at least 75 projects (training set) yielded better significance in prediction accuracy. Similar result was found in their previous study [39] whereby moving windows of about 3 to 4 years yielded improved prediction accuracy with a relative effective size of 81 to 89 projects using the ISBSG dataset.

2.4. Review on Growing Portfolio

The use of all available historical projects as the training set from a given dataset (referred to as the *growing portfolio*) has been the state-of-the-art for many researchers and software engineers when developing SEE models. Different techniques especially in the fields of machine learning, data mining and currently deep learning have been explored to aid in the construction of SEE models. Till date, none of these techniques can be referred to as the standard for SEE modelling. A plausible reason attributed to not having a standard SEE model might be due in part to the training dataset considered by researchers and practitioners. As a result, our review section considers a number of studies that have developed SEE models using the *growing portfolio* for improved prediction accuracy.

A recent study by Menzies *et al.* [22] investigated if modern SEE models resulted in better prediction accuracy as compared to the traditional SEE models focusing on using the *growing portfolio* as the training set. Here, they compared the classical COCOMO SEE approach [94] to modern approaches such as the analogy based estimation [95] and the SEE baseline model (*i.e.*, ATLM) by Whigham *et al.* [5]. The traditional Boehm's

COCOMO model [94] was found to be more accurate for the COCOMO dataset as compared to the new modelling techniques.

Whigham *et al.* [5] researched into resolving the problem of assisting software engineers with a more accurate and robust SEE model focusing on using the *growing portfolio* as the training set. They proposed a baseline SEE model namely automatically transformed linear model (ATLM). This baseline model (ATLM) is simple and can be benchmarked with newly developed SEE models for easy adoption after it proved superior to a complex ensemble method and a hybrid approach incorporating particle swarm optimization, analogy-based estimation (ABE) and clustering technique. Even though multiple linear regression (MLR) model has not been considered as the best estimation model in other studies [96][97], Whigham *et al.* [5] contend that it can form a baseline model provided relevant data-driven transformations are applied. Their argument of using MLR as a proxy for the formulation of SEE baseline model was based on previous studies [98][99] and the following seven characteristics:

- 1) simple, easily implementable and easy to interpret output;
- 2) outcomes should be deterministic;
- 3) no fine tuning of parameters during model construction;
- 4) relevant to a hybrid of quantitative and qualitative datasets;
- 5) output should be accurate than random guess;
- 6) provide relevant estimation information based on generalized dataset properties and
- 7) be made publicly available for easy implementation and execution.

Whigham *et al.* [5] argued that since there is no accepted standard for setting up a prediction model in SEE domain, their proposed ATLM can be used as a baseline model. ATLM supports the *n-fold* cross validation, the leave-one-out validation [43] and a suitable sample size with repeated training and validation sets. ATLM was benchmarked with the best ensemble method for effort estimation, namely Artificial Neural Networks (ANN) [23] and empirical analysis based on the 30-*fold* cross validation using three SEE datasets. Error evaluation measures, namely MMRE, LSD, PRED(25) and Relative Error (RE*) were used to evaluate the prediction accuracy of the models. Even though no statistical test was considered in their study [5], results indicated that ATLM performed accurately better in estimating the software effort as compared to the complex ensemble method (ANN). Again, they compared ATLM to a hybrid approach incorporating particle swarm optimization, ABE and clustering as proposed by Bardsiri *et al.* [100]. They used two benchmark datasets – Cocomo81 and Maxwell, and concluded that ATLM performed accurately better over the hybrid approach [100]. Here, evaluation was done with the MMRE based on a 10-*fold* cross-validation approach.

In another recent study by Mensah *et al.* [101], they considered a deep learning approach (specifically deep neural networks [52]) for investigating the use of *prior* and *posterior features* for software testing effort estimation. Here, the investigation study was focused on using the *growing portfolio* for empirical analysis. *Prior features* refer to the available features before project completion whilst *posterior features* refer to features available after project completion. Here, a mixed integer linear programming (MILP) optimization was setup to select relevant projects with their respective features from

within-company and cross-company projects. Mensah *et al.* [101] found that the MILP optimization selected *prior features* which yielded similar prediction accuracy as compared to *posterior features* provided the data was normalised. This result was confirmed by the Friedman's test with *p-values* less than 0.05 asymptotic significance level. It was also found that when the training and validation sets were normalised, selection of relevant cross projects improved the prediction accuracy in modelling. Data normalisation resulted in an improved prediction accuracy in a previous study by Kocaguneli *et al.* [102] when finding essential features and project instances in SEE.

Keung *et al.* [8] investigated the *conclusion instability* issue with regard to inconsistent estimated results from different SEE datasets and multiple prediction models. In their study, 20 different SEE datasets, 10 data preprocessors and 9 prediction models (learners) evaluated on 7 performance measures were used. Here, each of the 20 SEE datasets was considered as the *growing portfolio* when constructing the 9 prediction models. They realised that, aggregate analysis from multiple learners across multiple datasets assists in drawing stable conclusions in SEE. By empirical analysis and observation, Keung *et al.* [8] recommended that it is important for software engineers not to fit a single learner for a dataset in SEE.

Another study by Dejaeger *et al.* [98] using the *growing portfolio* found that the ordinary least squares (OLS) regression was the effective learner among a number of SEE learners used in predicting software effort. They used nine different datasets in their empirical analyses, namely coc81, cocnasa, desharnais, UPS05, Maxwell, ISBSG, ESA, experience and euroclear. Dejaeger *et al.* [98] concluded that for a more accurate effort

prediction model, there is the need for incorporating data transformation techniques, feature selection techniques, performance evaluation measures and statistical tests on a large training dataset (*i.e.*, the *growing portfolio*). This dissertation have incorporated the recommendations made by Dejaeger *et al.* [98] in the models developed in this study.

Another *growing portfolio* study by Sarro *et al.* [7] proposed a bi-objective algorithm, namely Confidence Guided Effort Estimator (CoGEE) for improving the prediction accuracy in SEE. This algorithm seeks to maximize accuracy and minimize uncertainties in relation to the effort estimation model. In their study, they used the MAE as one of the objective functions together with the confidence interval as the second objective function. Here, the confidence interval provides a likelihood range for the estimated effort values. They conducted 4 different experiments and evaluated their approach on 5 publicly available datasets namely Maxwell, China, Finnish, Miyazaki and Desharnais. In order to validate their proposed approach (CoGEE), they adapted a multiple-fold cross-validation method whereby the datasets were segmented into 3 test sets. Here, sampling of data points was randomly and uniformly selected without replacement. At each run, one of the partition subsets was used for testing and the remaining for training. In order to measure the accuracy of their prediction model for effort estimation, they used Mean Absolute Error and the Standardized Accuracy metrics. To test for statistical significance of their proposed model, they used the Wilcoxon Signed Rank test statistic [103]. This test statistic was utilized by them since it makes no assumptions prior to analyzing the datasets under consideration. The null hypothesis tested in their study was as follows: “The absolute errors from the estimation model, E_i

are significantly less than those from the estimation model, E_j ". This was achieved by setting a confidence limit at 5% significance level. They used the Bonferroni correction test statistic to test for multiple hypotheses. They also used the Vargha and Delayney's A_{12} test statistic [76] to test for the effect size since they realized that not all their utilized datasets followed the normal distribution. They checked the performance of their bi-objective optimization approach with three indicators, namely Contributions, Hypervolume and Generational Distance [104]. Results from their study show that, their proposed approach (CoGEE) outperforms the baseline estimation method [47] as well as the state-of-the-art methods for effort estimation (regression trees [105], case-based reasoning [106] and multiple linear regression, namely the ALTM [5]).

Another study focusing on using the *growing portfolio* as the training set, and conducted by Azzeh *et al.* [107] investigated the problem of finding the appropriate decision to achieve the best accuracy performance of individual set of cases for estimating software effort estimation. They proposed a new case adaptation approach (Local Tuning) with the aid of a multi-objective optimization algorithm specifically Particle Swarm Optimization (PSO) [108]. The multi-objective optimization algorithm by Azzeh *et al.* [107] is an extended version of Cabrera and Coello [109] multi-objective particle swarm optimizer based on a small proportion of dataset utilized. The goal of the multi-objective PSO approach by Azzeh *et al.* [107] is to obtain a set of optimal solutions from three objective functions that will improve the software effort estimation performance. They adapted three decision variables for their optimization problem, namely the number of nearest analogies, feature set and adaptation strategy weights and used their Local Tuning

adaptation approach to adapt the effort of nearest case based on minimizing the difference between its feature values and those of the new case. They then used an ordered weighted mean function to produce the effort for the new case and the predictive model was achieved based on the leave-one-out validation approach. The objective functions used in their study were three set of accuracy performance measures namely absolute error (*AE*), Inverse Balanced Relative Error (*IBRE*) and Balanced Relative Error (*BRE*). In their study, they resolved the problem of finding the best accuracy performance of individual set of cases by using the three objective functions and limiting their study to project artifacts from single companies. The artifacts used in their study were albrecht, kemerer, NASA, desharnais, cocomo, china, maxwell and telecom. Results from their study based on the 3 predictive accuracy objective measures show that Local Tuning adaptation approach outperformed 4 different CBR adaptation methods (CBR0, RTM, GA and LSE). Thus, Local Tuning significantly improved the estimation performance when cases are optimized SEearately. One key problem identified in their optimization approach is the lack of constraints and decision variable bounds utilized. This we believe can be built upon to make their effort estimation approach more robust and efficient to be adapted by practitioners.

Turhan and Mendes [110] found that stepwise regression (SWR) performed better in the field of SEE specifically cross-company modelling. They applied the SWR model on 125 web projects from 8 companies with or without filtering technique. SWR is one of the modelling techniques being employed in this study since some of the data used are cross-company datasets.

2.5. Differences between Current study and Previous studies

Even though the aforementioned studies have contributed a lot in software engineering especially in the domain of SEE, there is still more work to be done to help improve on prediction accuracy of SEE models. This dissertation acknowledges the assumption made by previous studies that *moving windows are closely related to new projects and can improve prediction accuracy in SEE modelling*. However, since prediction accuracy is affected by different window sizes and ages, there is the need for further investigation to address the aging and sizing constraints. Again, previous studies argue that prediction accuracy is affected by the existing weighting functions, namely Triangular, Epanechnikov, Gaussian and Rectangular. However, these previous studies were not able to establish a concrete evidence on which weighting function is most suitable for the moving window irrespective of the sizing constraint. Again, as a result of the relevancy constraint of sampling relevant projects to be considered as the moving window, previous studies (especially the study by Lokan and Mendes [36]) were not able to draw a stable conclusions to further acknowledge the effective use of moving windows against the *growing portfolio* benchmark. This leaves an open question for researchers to address – *should we discard the moving window assumption or further investigate whether its usage can improve prediction accuracy in most cases of predictive modelling?*

The main goal of this dissertation is to theoretically and empirically investigate the existence of exemplary projects (referred to as the *Bellwether effect*) in SEE focusing on chronological projects. Thus, the study investigates whether the use of exemplary and

recently completed projects (referred to as the *Bellwether moving window*) can improve prediction accuracy as compared to using an entire collection of historical projects (referred to as the *growing portfolio*). Thus, given p_i projects (set as hold-out) whose target is to be estimated, the study investigates whether an entire N historical projects or an m subset of recently completed projects ($m \subset N$) can yield improved prediction accuracy. Note that both the *growing portfolio* (N) and the *Bellwether moving window* (m) have their development completion dates prior to the start dates of the p_i projects. It should be noted that this dissertation is the first to make an attempt to consider investigating the concept of *Bellwethers* to address the relevancy, sizing and aging constraints of moving windows in SEE. The concept of *Bellwethers* have been elucidated to yield improved prediction accuracy in software defect prediction [26] and online analytical processing queries [28][29] hence, this study believes it is worth investigating in SEE. In order to investigate the existence of such *Bellwether* projects in SEE, the study was initialized by six theoretical postulations followed by empirical analysis using chronological datasets sourced from the ISBSG and the PROMSE repository [49].

This dissertation introduces three *Bellwether methods* (*SSPM*, *SysSam* and *RandSam*) to select the *Bellwether* projects based on the concept of *Bellwether effect* and Markov chains. These exemplary or *Bellwether* projects are obtained using well-defined stratification and sampling procedures (as elaborated in chapter 3). The *Bellwethers* can be considered as the best training set for constructing SEE models to predict the targets of *new* projects. The dissertation also introduces a guideline for sampling *Bellwethers* for predictive modelling (see section 4.4 of chapter 4).

This dissertation also proposes a novel guideline for sampling *Bellwether* projects from a given chronological dataset for setting up the SEE models. This will assist software engineers and practitioners to sample effective and relevant data samples to be used as the training set for improving prediction accuracy of *new* project estimation.

Different from previous studies, robust statistical evaluation measures are employed in this dissertation to assess the statistical significance of prediction results for the effective usage of the *Bellwether moving windows*. Thus, robust statistical measures such as Cliff's δ effect size, Brunner's ANOVA-like test and Yuen's test are used to compare the empirical results between the *Bellwether moving window* and the *growing portfolio* benchmark.

In addition to the SEE baseline model, namely ATLM [5], this study also setup an ElasticNet regression [3] and deep neural networks [52][111][101][112] as predictive models for using the *Bellwether moving windows* against the *growing portfolio* benchmark. The MAE as employed in previous studies is employed for assessing prediction accuracy of the learners or prediction models. MAE is recommended by Shepperd and MacDonell [47] to be unbiased to underestimation and overestimation. Similarly, Foss *et al.* [113] recommended MAE as an effective, reliable and unbiased evaluation measure.

This study also considers a more robust transformation approach as recommended by Kitchenham *et al.* [53] and Wilcox and Keselman [114], namely *trimming* in addition to *log transformation* as well as the Kernel density plot for data preprocessing (refer to chapter 3 for detailed description of the data preprocessing techniques employed).

With regard to the weighting function constraint, this dissertation makes use of two additional weighting functions, namely Biweight and Triweight [55][56] to further investigate their potential effect on prediction accuracy for the moving window modelling approach. Thus, an intensive study was conducted across the studied datasets to investigate if a feasible weighting function can be obtained for improved prediction accuracy.

Another unique feature of this dissertation is to provide an effort classification scheme to help reduce the *conclusion instability* challenge [8][43][46][115] in SEE. The software classification methodology introduced in this study can estimate the level of effort expended irrespective of the numerous prediction models, evaluation measures, cross or within company projects and preprocessing techniques considered. Due to the fact that several SEE models have been proposed with no clear dominating technique, the study builds variations of regression and deep neural network models incorporating different preprocessing and feature subset selection techniques in assessing prediction performance across different datasets. This section of the study is partially based on an assessment made by Whigham *et al.* [5] and Dejaeger *et al.* [98] that a regression based model can form a simple and reliable prediction model for SEE. The classification of software effort in the studied datasets is to enable self-guided interpretation of estimated effort as well as enable comparison of the characteristics of the *new* projects with past or completed projects. The extent of similarity between the *new* projects and completed projects will hopefully increase the confidence in the effort estimation.

Chapter 3

Methodology

3.1. Description of Datasets

1) Desharnais Dataset

This dataset was collected by Desharnais [116] from 10 Canadian software organisations and it has been made publicly available at the PROMISE repository [49]. It is made up of a total of 81 management information system (MIS) development projects with 10 features. Projects in the dataset were completed between 1983 to 1988 over an approximate span of 6 years. Following a similar procedure by Li *et al.* [117][118] and Shepperd and Schofield [106], eight features were selected for modelling. The independent features are *team experience (teamExp)*, *manager's experience (managerExp)*, *programming language*, *number of entities*, *number of unadjusted function points (pointsNonAdjust)*, *number of transactions* and *the FPA technical complexity scale factor (envergure)*. The dependent feature is the development *effort* variable measured in person-hours. The remaining two features within the Desharnais dataset are the *length (or duration)* and the *number of adjusted function points (pointsAdjust)*. With the exception of the *language*, *teamExp* and *managerExp* features

which are categorical variables, the remaining features are numerical variables. Note that this study considered *prior features* (i.e., independent variables known prior to project development) for setting up the SEP models. The *length* or project duration variable was therefore not used as an independent variable in the prediction models because it can only be known at the end of project development. In fact, the *length* variable could be considered as a dependent variable in some cases; hence it was not used as an independent variable in this study. Note that the *adjusted function point* variable is a *prior feature* that can equally be considered as an independent variable for setting up an SEP model, similar to the case of the *unadjusted function point* variable used in our study. The difference between the two variables is that the *adjusted function point* value is a multiplication of the *unadjusted function point* by a scale factor, as described by Desharnais [116]. The two variables are strongly correlated (with a correlation coefficient value, $r = 0.986$) and so their inclusion would negatively affect model stability and interpretability if used in the same SEP model [3]. We provide summary statistics of the project features in Table 1. Further details of this dataset is made available at the Appendix.

Table 1 Descriptive Statistics of Sampled Features in the Desharnais Dataset

Feature	N	Min	Max	Mean	SD	Skew	Kurt
TeamExp	79	0	4	2.27	1.34	-0.042	-1.26
ManagerExp	78	0	7	2.67	1.52	0.20	0.07
Transactions	81	9	886	179.90	143.32	2.36	7.73
Entities	81	7	387	122.33	84.88	1.34	1.48
PointsAdjust	81	73	1127	302.23	179.68	1.78	4.94
Envergure	81	5	52	27.63	10.59	-0.11	-0.28
PointsNonAjust	81	62	1116	287.05	185.11	1.67	4.16
Effort	81	546	23940	5046.31	4418.77	2.01	4.72

Min and Max denote Minimum and Maximum respectively

SD denotes standard deviation

Skew and Kurt denote skewness and kurtosis respectively

Since the Desharnais dataset does not have the commencement or start date of project development, we computed it by subtracting the development duration or length variable (*length*) from the end date variable (*yearEnd*). As a result of the *length* variable being in months and the *yearEnd* variable in years [116], we converted *length* to years for uniform computation of *yearStart*. Thus, start date (*yearStart*) is defined in (3) as follows:

$$yearStart = yearEnd - devLength \quad (3)$$

2) Kitchenham Dataset

Kitchenham *et al.* [42] were the first to use a dataset for moving window modelling in SEP. The Kitchenham dataset, which is publicly available at the PROMISE repository [49], includes projects undertaken between 1994 and 1999, with an approximate span of 6 years. It is a chronological dataset comprising 145 maintenance and development projects with 9 features from a single organisation, namely the Computer Science Corporation (CSC). Further details about this dataset are provided in previous studies by Kitchenham *et al.* [42] and Tsunoda *et al.* [119]. The features or variables within the dataset include *project type*, *actual duration*, *actual start date*, *estimated completion date*, *adjusted function points*, *client code*³, *first estimate*, *first estimate method* and *actual effort*. Here, we selected two *prior* features [101], namely *adjusted function points (AFP)* and *project type* as the independent variables and *actual effort* as the dependent variable, following a similar procedure to that used in our previous study [31]. We did not use the

³ The client code denotes the various categories of clients {1,2,3,4,5,6} in which the CSC organisation develops or maintains software products on their behalf.

posterior feature (i.e., *actual duration*) as an independent variable since it is impossible to determine the actual duration of a project prior to its development. Note that the *actual duration* variable is measured in days [42] and therefore for consistency, we converted it to years prior to modelling.

The 145 projects within the Kitchenham dataset are distributed across the 6 clients of the CSC organisation as follows: Client 1 ~ 16 projects, Client 2 ~ 116 projects, Client 3 ~ 4 projects, Client 4 ~ 4 projects, Client 5 ~ 1 projects and Client 6 ~ 4 project. Out of the 145 projects, 3 projects were detected as having missing entries with respect to the *estimated completion date* variable. As a result of that we computed the completion dates for those 3 project cases by adding the *actual duration* (in years) to the respective *actual start date*. Table 2 provides a descriptive summary of sampled features within the dataset. Further details of this dataset is made available at the Appendix.

3) ISBSG Dataset

The International Software Benchmarking Standards Group (ISBSG) dataset release 10 sourced from the ISBSG repository was used for this study. It is a chronological dataset with a total of 4106 projects and 105 features from over 25 countries.

Table 2 Descriptive Statistics of Sampled Features in the Kitchenham Dataset

Feature	Min	Max	Mean	SD	Skew	Kurt
Actual_duration	0.1	2.6	0.57	0.37	1.93	6.12
AFP	15.36	18137.48	527.67	1521.99	10.92	126.70
First_estimate	121	79870	2855.97	6789.29	10.34	116.91
Actual_effort	219	113930	3113.12	9598.01	10.87	125.64

AFP denotes adjusted function points

It contains variants of projects from different organisations such as communications, insurance, banking, manufacturing, government and business services. All projects are within the range of May, 1988 to November, 2007 with an approximate span of 20 years. To facilitate an effective replication of existing moving window studies, we used the same ISBSG dataset (release 10) that has previously been considered for prediction modelling [37][35][31]. We realised that new development projects tend to require higher effort than enhancement projects. There is no unique pattern with respect to specific programming languages used for development of the projects. Further details of the ISBSG dataset are provided in studies by González-Ladrón-de-Guevara *et al.* [120], Li *et al.* [117] and Lokan *et al.* [121].

The projects in the ISBSG dataset are submitted by different organisations with different policies and development methodologies. Hence, such cross-organisation projects are heterogeneous in nature [117] and do not share common characteristics such as common development team, programming skills and other development policies. Different from our original study [31] that used cross-organisation projects, this study sampled projects from a single organisation for our empirical investigation. Hence, a total of 699 projects (22.1%) from the Communication organisation has been considered in this study. We provide the summary statistics of sampled features in Table 3. Project Delivery Rate (PDR) is a feature in the ISBSG dataset that was examined in a previous study by Amasaki and Lokan [34]. They found that PDR changes with time and low PDR suggests high productivity. This finding supports the use of moving windows in prediction modelling [37]. Further details of this dataset is made available at the Appendix.

Table 3 Descriptive Statistics of Sampled Features in the ISBSG Dataset

Feature	N	Min	Max	Mean	SD	Skew	Kurt
AFP	614	3	4911	290.06	466.49	5.55	43.72
Norm_PDR	606	0.3	315.6	16.49	19.97	7.48	94.39
Team_size	260	1	50	9.12	8.79	2.16	5.48
Input_count	269	0	2075	97.15	182.09	7.05	65.35
Output_count	268	0	667	67.60	109.58	3.06	10.84
Enquiry_count	269	0	639	43.50	72.95	4.21	25.16
File_count	268	0	1208	64.71	124.75	5.05	33.57
Interface_count	265	0	747	20.09	75.43	6.91	53.97
Added_count	277	0	3738	182.73	375.78	5.10	35.62
Changed_count	276	0	1369	89.30	153.95	4.28	26.07
Deleted_count	273	0	2657	15.77	162.72	15.87	257.9
Norm_Effort	640	4	88555	3508.9	7020.4	7.02	63.20

Norm_PDR denotes normalised project delivery rate

4) *Maxwell Dataset*

In this study, we use the Maxwell dataset [122] sourced from the PROMISE repository [49] for our empirical analysis. It is a chronological dataset of a total of 62 projects with 27 features from a commercial bank in Finland. It contains variants of projects from a single organisation. All projects were undertaken between 1985 and 1993, giving an approximate age span of 9 years. This dataset has been considered for prediction modelling in previous studies [117][118][123][3] especially in the context of *growing portfolio* analysis. Apart from 4 features, namely *size*, *effort*, *time* and *development duration* which are numerical, the remaining features are nominal and ordinal. There are a total of 6 nominal features and 16 ordinal features. The dependent variable is the *effort* measured in hours and the remaining features (except the *syear* and *duration* variables) make up a feature space of independent variables. Note that, the *syear* (start year) and *duration* variables are *posterior features* and cannot be considered for predictive

modelling when sampling BMW. Descriptive statistics of sampled numerical features are provided in Table 4. Further details of this dataset is made available at the Appendix.

Since the Maxwell dataset does not have the date of completion variable, we computed it by adding the *development duration* to the *start date (syear)* as defined in (11). Thus, the completion date variable (*cyear*) was used to sort the projects in ascending order as described in the *Bellwether method* (section 3.4).

$$cyear = syear + duration \quad (4)$$

where *duration* refers to the amount of time from specification of a project to delivery. The *duration* variable is measured in calendar months and the variable *syear* in calendar years. To ensure consistency, we converted *duration* from months to years before computing *cyear* in (14).

Although the aforementioned datasets might seem old, the focus here is not on the dataset *per se* but on the possible presence and effect of *Bellwethers* in SEE modelling.

Table 4 Descriptive statistics of sampled features in the Maxwell Dataset

Variable	Minimum	Maximum	Mean	SD	Skewness	Kurtosis
Nlan	1	4	2.55	1.019	-0.039	-1.083
Duration	4	54	17.21	10.651	1.285	1.562
Size	48	3643	673.31	784.085	2.337	5.319
Time	1	9	5.58	2.131	-0.435	-0.706
Effort	583	63694	8223.21	10499.9	3.347	13.695

3.2. Data Preprocessing

1) Feature selection

Feature selection refers to the optimal subset selection of features that provides the best prediction accuracy [124]. In a previous study by Li *et al.* [118], a comparative study of

different wrapper feature selection algorithms was performed on the Maxwell dataset. They proposed an approach known as mutual information based feature selection for case based reasoning (MICBR). Their proposed MICBR feature selection algorithm was compared with forward sequential selection (FSS), hill climbing (HC) and exhaustive selection (EX). The features selected from the Maxwell dataset by MICBR were *T14*, *duration* and *size*. Three features selected by FSS were *T05*, *duration* and *size*. Twelve out of 25 features were selected by HC and they were *time*, *App*, *source*, *telonuse*, *Nlan*, *T01*, *T06*, *T07*, *T08*, *T14*, *duration* and *size*. No features were selected by the exhaustive selection (EX) algorithm. Li *et al.* [118] found that the three features selected by MICBR (*T14*, *duration* and *size*) resulted in improved prediction accuracy as compared to the others. The *duration* variable which is a *posterior* feature can only be known at the completion of a project. As a result of that, this study made use of two *prior* features (namely *T14* and *size* as the independent variables) and the target feature, *effort* (as the dependent variable) for our empirical analysis. *Prior* features refer to independent variables known before the development of a *new* project whilst *posterior* features refer to variables known after the development of a project [101].

With regard to the Desharnais dataset, this study followed a similar feature selection procedure by Li *et al.* [117][118] and Shepperd and Schofield [106], and selected eight features for the SEE modelling. The independent features are *team experience* (*teamExp*), *manager's experience* (*managerExp*), *programming language*, *number of entities*, *number of unadjusted function points* (*pointsNonAdjust*), *number of transactions*

and *development environment (envergure)*. The dependent feature is the *development effort* variable measured in person-hours.

With regard to the Kitchenham dataset, two *prior* features [101], namely *adjusted function points (AFP)* and *project type* were selected as the independent variables and *actual effort* as the dependent variable following similar procedure in a previous study by Mensah *et al.* [31]. We did not use the *posterior* feature (*i.e.*, *actual duration*) as independent variable since it is difficult to accurately determine the actual duration of a project prior to development. Note that the *actual duration* variable is measured in days [42] and therefore for consistency, we converted it to years prior to modelling.

With regard to the ISBSG dataset, the selected features were the *size* of the projects measured in Unadjusted Function Points (UFP), *language type* (3GL and 4GL), *development type* (new development, re-development and enhancement), *development platform* (PC, mainframe, midrange and multi-platform) and the normalised *development effort* measured in person-hours as done in previous studies [32][31][37][35][119]. Aside of the normalised *effort* regarded as the dependent variable [117], the remaining *prior* features are regarded as the independent variables.

2) Elimination of irrelevant instances and missing data points

A systematic analysis [23] was performed on all data instances per each dataset to address missing data values and project cases which were not relevant to the analysis. A handful of 4 missing entries were found in the Deshairnais dataset and eliminated prior to setting up the prediction model as done in previous studies [3][106][117][118]. Note that the 4 missing data entries were observed across the selected independent variables. We found

three missing data points in the Kitchenham dataset with regard to the *estimated completion date* feature. Since the dataset extracted already has the *actual start date* and the *actual duration* for each project, we estimated the three missing values by adding the *actual duration* to the *actual start date*. Thus, no data point was eliminated from the Kitchenham dataset prior to prediction modelling. With regard to the Maxwell dataset, no missing data point was found.

Similar to the original study [31], we eliminated irrelevant project cases in the ISBSG dataset as follows:

- Eliminate projects if their respective project elapsed times or implementation dates are unknown [37][35].
- Eliminate projects with low data quality rating (that is, projects not rated as A or B by ISBSG are excluded).
- Eliminate projects whose UFP rating is not A as suggested by ISBSG [125].
- Similar to previous studies [126][127][128], select projects based on functional sizing method with IFPUG version 4.0 and above.
- Eliminate projects with normalised ratios more than 1.2 as recommended by ISBSG [125]. Normalised ratio defined as the ratio of the normalised effort to the summary effort is suggested by ISBSG for project subset selection and refinement [125][126].
- Eliminate projects with unknown development team effort as reported in the ISBSG dataset. That is, projects with resource level ‘1’ denoting development

team effort are selected following similar approach in previous studies [126][129][130].

- Eliminate projects with unknown organisation types [127].
- Eliminate projects which are considered as web projects [35][37]. This is due to the established fact that web projects have different properties from the other projects in this dataset [35]. Also function point analysis does not reflect all the necessary features that have an influence in the determination of effort of web projects.
- Eliminate projects with missing data entries based on the selected features.

3) Outlier and Influential data point detection

Outliers and influential data points are known to affect the performance of prediction models when left unresolved. Following Mendes and Kitchenham [131] approach on stability analysis, we investigated the effect of outliers and influential data points in the studied datasets which can affect prediction performance during analysis. We first fit the prediction model without considering data transformation and investigate the effect of outliers and influential data points. Secondly, we transform the studied datasets and investigate the effect of outliers and influential data points. In each case, we evaluate the performance of the prediction model. Outliers and influential data points are identified and removed provided they added significant effects to the construction of the prediction model [3][5][33][61][62]. Outliers are identified based on the Cook's distance [132] and kernel density plots (recommended by Kitchenham *et al.* [53]). Cook's Distance is used following a similar approach by Seo and Bae [130] in the identification and treatment of

outliers during model construction. Thus, we considered an observation as an outlier if the Cook's distance is three times more than the mean Cook's distance.

With regard to influential data points, we used the DFFITS diagnostic [133][134] to determine such observations. We considered an observation as influential when its exclusion results in major changes in the prediction model. Thus, an observation is considered influential if its DFFITS exceeds 1 for small to medium datasets and $2\sqrt{2/n}$ for large datasets.

4) Data trimming + transformation

The use of heavy-tailed distributed data for predictive modelling does not provide reliable prediction estimates. It is very important to consider necessary data transformation measures to investigate their effect on prediction accuracy. A recent study by Kitchenham *et al.* [53] recommended the use of *trimming* instead of simple transformation techniques (*e.g. log transform, square root, box cox, etc*) as a robust method of transforming data. Simple transformations are known to transform skewed data to symmetric but does not necessarily deal with outliers [33][34][64]. Trimming of data was proposed by Wilcox and Keselman [114] to address heavy-tailed distributed data. Trimming is the process of eliminating the tails of a distribution in a population or sample under investigation prior to setting up the prediction model. This results in an average of trimmed subset (*trimmed mean*) used as a measure of central location. One *trimming* method suggested in previous studies [34][64] is to remove a fixed proportion of the smallest and largest values given that the data is sorted in order of ascendancy [114]. A 20% threshold was suggested as a plausible proportion to be trimmed. For example, given that a dataset consists of 20 data

points, 20% trimming refers to eliminating 2 smallest and 2 largest data points assuming the data is sorted. Another *trimming* method suggested by Wilcox and Keselman [114] is to detect and remove outliers from the dataset and compute the average of the remaining data subset.

Both methods alter the dataset prior to setting up a prediction model. In this study, we adopt the second method and eliminate outliers provided they significantly affect the prediction accuracy in the model construction. Afterwards, we conduct empirical analyses with and without simple transformation technique to further investigate its significant effect. We make use of the *log transformation* [136] which was proven in our original study [31] to improve prediction accuracy. Previous moving window studies [34]–[37] considered the use of *log transformation* to transform features (*e.g.* size, duration, effort, *etc*) to follow the normal distribution. We seek to confirm in this study whether the use of *log transformation* on a subset of *trimmed* data plays an important role in prediction analysis.

For example, in Table 5 three outliers and three influential data points (*i.e.*, the 10th, 19th and 64th project cases) were identified by the Cook’s distance as well as the DFFITS in the case of the *log transformed* Desharnais dataset. We found that only the 64th project case was very influential or significantly affected the predictive modelling. Hence, we *trimmed off* that culprit project case prior to investigating the *Bellwether* projects. In the Kitchenham dataset, an outlier and influential data point (*i.e.*, the 62nd project case) was detected by the Cook’s distance and DFFITS in the case of the *log transformed* data (Table 5). The influential data point was removed since it significantly

affected the predictive modelling. It should be noted that in the case of the untransformed data, the same 62nd influential data point affected the prediction model. However, in the ISBSG dataset, 3 influential data points were *trimmed off* (i.e., the 23rd, 26th and 74th project cases) when the *log transformation* was applied.

With regard to the Maxwell dataset, two outliers (the 10th and 28th project cases) were identified by the Cook's distance for the untransformed data. We investigated the effects of these two outliers by gradually including and excluding them in the prediction model (ATLM). We found that the exclusion of the 10th project case resulted in a significant change on prediction accuracy (*wrt* MAE) whilst exclusion of the 28th project case caused no significant change. In order to confirm the removal of the outlier (10th project case), we applied *log transformation* on the raw data to further investigate its effect on prediction accuracy. Here, five project cases were detected as outliers. We realised that none of the two outliers (detected in the case of the untransformed data) were identified in the case of the *log transformation*. That is, the 10th project case caused no significant change in *log transformation*. Project cases 14, 21, 45, 46 and 48 were recorded as outliers in *log transformation*. We gradually *trimmed off* (or excluded) the outliers and fitted the prediction model following the same approach in the case of the untransformed data. We found that removing two of the outliers (with Cook's distance between 0.1 and 0.12) resulted in improved prediction accuracy as compared to the untransformed case. Thus, project cases 21 and 46 were *trimmed off* in the sorted and *log transformed* data. Therefore, *trimming + log transformation* was applied on the Maxwell dataset in the empirical analysis in order to improve on prediction performance.

Table 5 Statistics of Outliers and Influential data points identified and trimmed off

Dataset	Cook's Distance		DFFITS		Trimmed off
	unlog	log	unlog	log	
Desharnais	4	3	6	3	1
Kitchenham	1	1	3	1	1
ISBSG	9	2	9	5	3
Maxwell	2	5	2	2	2

The Kernel density plots in Fig. 3.1-.4(a) and Fig. 3.1-.4(b) show the distribution of the target variable (*effort*) before and after *trimming + log transformation* respectively. The distribution of the data after *trimming + log transformation* was observed to follow the normal distribution (b) as compared to before data transformation (a). In order to make an inference that the existence of *Bellwether* should follow a normal distribution (Postulation 3), we further performed a statistical test of normality using the Shapiro-Wilk test [137] to validate the data distribution. We found that the selected features followed the normal distribution with *p*-values (>0.05) at 5% asymptotic significance level. Results from the Shapiro-Wilk tests of normality confirm that for a *Bellwether* to yield a relative superior prediction accuracy, the project data must follow the normal distribution.

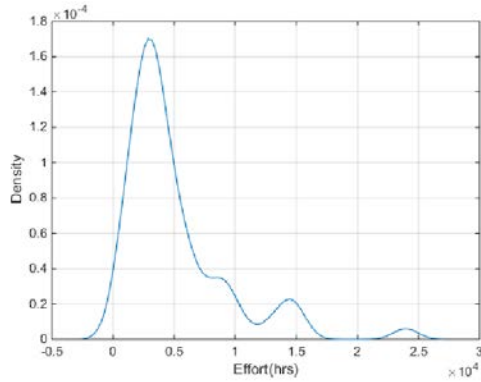


Fig. 3.1a Kernel density plot showing the distribution of the effort data before *trimming + log transform* in the Desharnais dataset

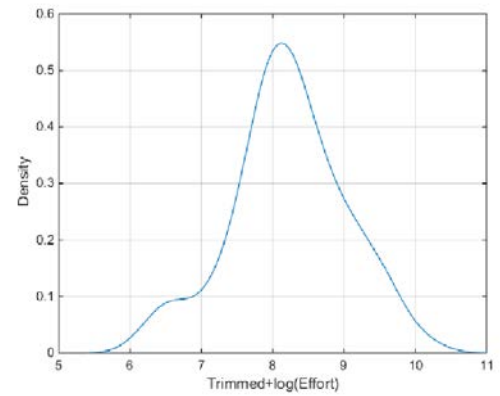


Fig. 3.1b Kernel density plot showing the distribution of the trimmed effort data after *trimming + log transform* in the Desharnais dataset

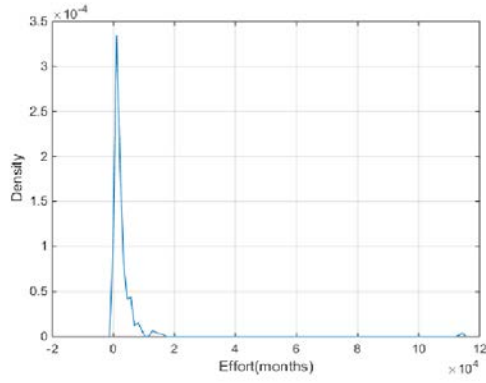


Fig. 3.2a Kernel density plot showing the distribution of the effort data before *trimming + log transform* in the Kitchenham dataset

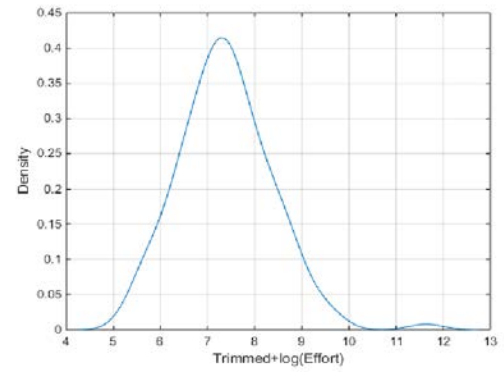


Fig. 3.2b Kernel density plot showing the distribution of the trimmed effort data after *trimming + log transform* in the Kitchenham dataset

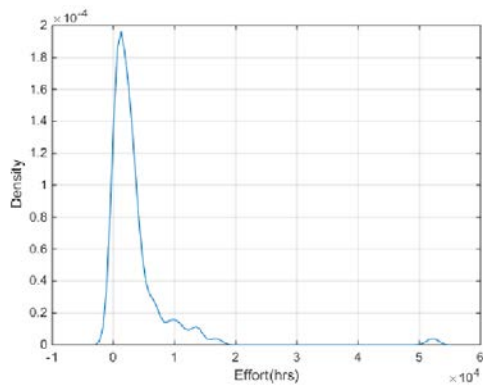


Fig. 3.3a Kernel density plot showing the distribution of the effort data before *trimming + log transform* in the ISBSG dataset

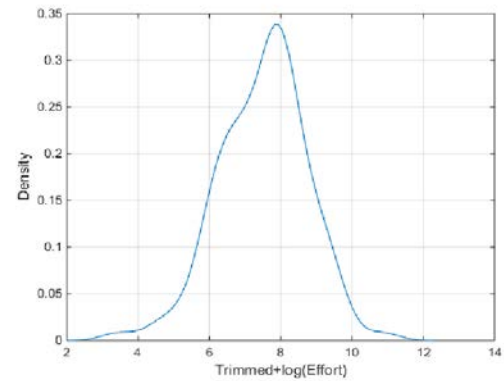


Fig. 3.3b Kernel density plot showing the distribution of the trimmed effort data after *trimming + log transform* in the ISBSG dataset

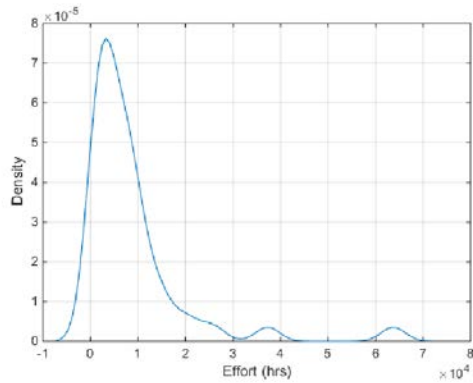


Fig. 3.4a Kernel density plot showing the distribution of the effort data before *trimming + log transform* in the Maxwell dataset

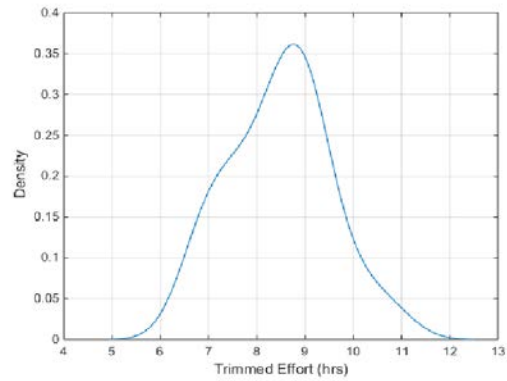


Fig. 3.4b Kernel density plot showing the distribution of the trimmed effort data after *trimming + log transform* in the Maxwell dataset

Figure 3 Kernel density plot showing the distribution of data before and after *trimming + log transformation*

Note that for normality, the *p-value* should be more than the asymptotic statistical significance level ($\alpha = 0.05$). Thus, the error evaluation measures (MAE) resulted in *p-values* of more than 0.05 for the Shapiro-Wilk tests respectively.

The aforementioned preprocessing procedure resulted in a subset of 117 projects (16.7%) out of the sample space of 699 projects from the Communication organisation in the ISBSG dataset. Out of the preprocessed subset selection, 107 projects (91.5%) were considered as the *growing portfolio* and 10 projects (8.5%) with recent completion dates were set aside as hold-out for testing the prediction accuracy of the deep learning model (DNN). Note that the *Bellwether* was sampled from the *growing portfolio* in the empirical analysis conducted in this study. The 3 *Bellwether methods* (i.e. *SSPM*, *SysSam* and *RandSam*) were used to select the *Bellwether* to be used as the training set for the predictive modelling. Similarly, with regard to the Desharnais dataset, 76 projects (93.8%) resulted from the preprocessing. Out of these 76 preprocessed projects, we used 5 projects (6.6%) with recent completion dates relative to the remaining projects as the hold-out. Thus, we selected the 5 projects which has 1988 as their completion dates. The remaining 71 projects (93.4%) ranging from 1982 to 1987 were considered as the *growing portfolio* out of which the *Bellwether* was sampled.

With regard to the Kitchenham dataset, 144 projects resulted from preprocessing. Out of this total, 134 projects (93.1%) were considered as the *growing portfolio* and 10 projects (6.9%) as the hold-out. Lastly, with regard to the Maxwell dataset, a total of 60 trimmed projects resulted from preprocessing, whereby 57 projects (91.9%) were

considered as the *growing portfolio* and the remaining 3 projects (4.8%) considered as the hold-out.

3.3. Theoretical Postulations

In order to establish the theoretical foundation for the existence and effective usage of *Bellwether moving windows* from a given chronological dataset, this study states and proves the following six postulations:

Postulation 1. *If Ω is a sample space of a set of chronological projects, then the prediction probabilities of its partition sample sets $\{X_1, \dots, X_q\}$ are independent.*

Proof. Suppose the sample space, X is partitioned into q partition sample sets where each $X_i \in \Omega$ is drawn independently, then the prediction probabilities⁴ of two events, X_i and X_j can be defined as follows:

Assume that the prediction probability of event X_i occurs after event X_j has occurred (5) and vice versa (6), then according to the conditional probability rule,

$$P(X_i|X_j) = \frac{P(X_i \cap X_j)}{P(X_j)} \quad (5)$$

$$P(X_i|X_j)P(X_j) = P(X_i \cap X_j)$$

$$P(X_j|X_i) = \frac{P(X_i \cap X_j)}{P(X_i)} \quad (6)$$

$$P(X_j|X_i)P(X_i) = P(X_i \cap X_j)$$

⁴ Prediction probability is the ratio of the prediction outcome to the actual outcome.

where $P(X_i) > 0$ and $P(X_j) > 0$.

To show that the events X_i and X_j are independent, then the prediction probability of X_i does not affect the prediction probability of X_j and vice versa. Thus, $P(X_i|X_j) = P(X_i)$ and $P(X_j|X_i) = P(X_j)$.

From the multiplication rule which states that the probability that two events X_i and X_j occur is the product of the probability of X_i and the probability of X_j . That is,

$$P(X_i \cap X_j) = P(X_i) P(X_j) \quad (7)$$

Substituting (7) into (5) gives

$$P(X_i|X_j) = \frac{P(X_i) P(X_j)}{P(X_j)} = P(X_i) \quad (8)$$

Similarly, substituting (7) into (6) gives

$$P(X_j|X_i) = \frac{P(X_i) P(X_j)}{P(X_i)} = P(X_j) \quad (9)$$

From (8) and (9) it is confirmed that events X_i and X_j from a given sample space, Ω are independent. For example, $P(X_i|X_j) = P(X_i)$ means that the prediction probability of the occurrence of event X_i given that event X_j has already occurred is the same as the prediction probability of event X_i . This implies that $P(X_i \cap X_j) = P(X_i)P(X_j)$ since $P(X_i|X_j) = P(X_i)$ as shown in (6).

Generally, $P(X_i \cap \dots \cap X_q) = P(X_i) \dots P(X_q)$ which implies that the prediction probabilities of the partition samples, $X_i \in \Omega$ are independent of each other.

This postulation is validated based on empirical analysis whereby each partition from the sample space resulted in different prediction probabilities of the target variable

(software effort). For example, in order to predict the software effort (Y) of p_i projects using three partition samples (training sets), X_1 , X_2 and X_3 from the ISBSG dataset, we found that their respective prediction probabilities are as follows: $P(X_1) = 0.58$, $P(X_2) = 0.43$ and $P(X_3) = 0.69$. Thus, the training set, X_1 can predict about 60% of the actual effort of the p_i projects, X_2 can predict about 40% whilst X_3 can predict about 70% when such samples are used for building predictive models in the context of software effort estimation.

Postulation 2. *Given that the independent partition sample sets $\{X_1, \dots, X_q\}$ have respective means $\{\bar{x}_1, \dots, \bar{x}_q\}$, then there exists a sample mean, $\bar{X}$ which can be used as an unbiased estimate of the population mean, μ .*

Proof. Let $\{X_1, \dots, X_q\}$ be an independent and identically distributed (*iid*) sequence with finite means, $\bar{x}_1 < \infty, \dots, \bar{x}_q < \infty$ respectively, then according to the strong law of large numbers [30][67], as the sample size increases then there exists a sure probability of one [30] that

$$\lim_{q \rightarrow \infty} \frac{X_1 + X_2 + \dots + X_q}{q} = E\left(\frac{\sum X_i}{q}\right) \approx E(\bar{X}) = \mu \quad (10)$$

Thus, according to Dobrow [30], the limit of the average of samples, X_i as the number of samples, q approaches infinity is the expectation of the sample mean which can be used as the population mean. Similarly, the confidence interval of the sample, X can be computed as shown in (11).

$$\bar{X} \pm Z_{\alpha/2} \left(\frac{S}{\sqrt{q}} \right) \quad (11)$$

where $\bar{X}$ is the sample space mean, S is the standard deviation of the sample space, q is the number of independent partition subsets in the sample space and $Z_{\alpha/2}$ is the value from the standard normal distribution based on the desired confidence level (α). Confidence interval computation assists in verifying the amount of uncertainty associated with the sample mean estimate to the population mean parameter. The lower and upper limits of the confidence interval forms the range for $\bar{X}$ and can be used to infer the range for the population mean, μ .

Postulation 3. *If the sample space (Ω) follows a normal distribution, then there exists a partition sample, X_i that can be used as the Bellwether.*

Proof. Let the sample space (Ω) in which the respective sample (X_i) to be drawn from, follow a normal distribution with mean, $\bar{X}$ and variance, S^2 . That is, $X_i \sim N(\bar{X}, S^2)$. To achieve this assumption, we apply the *log transformation*, the *z-score* normalisation and other preprocessing techniques such as the kernel density plot [138] and the Cook's Distance [130] for treatment of outlier and influential data points in the sample space. We investigated if the sample (X_i) followed the normal distribution based on the skewness and kurtosis statistics. For normality, the skewness and kurtosis of X_i should be close to zero and three respectively [139]. Also for normality, as the sample size increases, its sample mean will follow the normal distribution (central limit theorem [66]). Results from the empirical analysis show that the *log transformation* converted the features or attributes of the sample space to follow the normal distribution with skewness approximately zero

and kurtosis approximately three. It was realised that as the size of the sample space turns to be close to the size of the *growing portfolio* of a given dataset, the normality assumption is not only met but there exists a *Bellwether effect* in such a sample space.

For $\{X_1, \dots, X_q\} \in X$, then $\exists X_i$ such that: $P(X_i) \approx P(X)$. If $P(X_i) \approx P(X) = \frac{\text{predicted outcome}}{\text{actual outcome}}$, then X_i follows a normal distribution with mean, $\bar{X}_i$ and variance, S_i^2 . That is, $X_i \sim N(\bar{X}_i, S_i^2)$. The X_i with the best prediction accuracy among the remaining $X_j \forall j \neq i$ forms the *Bellwether* sample yielding approximately similar prediction accuracy as compared to the overall sample space, X . This postulation is validated based on empirical analysis in chapter 4.

Postulation 4. *Given a set of N chronological projects with a finite mean (μ) and variance (σ^2), then there exists a potential Bellwether which can be used as a moving window.*

Proof. Let $\{X^{(1)}, \dots, X^{(t)}\}$ be independent and identically distributed (*iid*) random samples chronologically drawn from a given population, N . If a random sample $X^{(i)} > 30$ is drawn from N , then (a) according to the central limit theorem [66] and the law of large numbers [67], $X^{(i)}$ will follow the normal distribution with mean, μ and variance, σ^2 that is, $X^{(i)} \sim N(\mu, \sigma^2)$ and (b) the $X^{(i)}$ with the minimum accuracy measure(s) to be considered as the *Bellwether* can then be used to successively predict the remaining samples, $X^{(j)} \forall j \neq i$. That is the i^{th} *Bellwether* sample predicts the remaining j^{th} samples.

In order to confirm the existence of a potential *Bellwether* from a given population set, N empirical analysis was performed based on statistical stratification of N using the *X-means* clustering algorithm [140], and samples were drawn from each stratum. It was

realised that the best population stratum in which potential *Bellwether* samples can be drawn from was of size not less than 100 projects as observed in an investigation study by Mensah *et al.* [31] using the ISBSG and the Kitchenham datasets. Note that these results were empirically validated by applying the *Bellwether methods* as detailed in section 3.4.

Postulation 5. *If ‘P’ denotes a regular transition probability matrix (TPM) of a Markov chain, then there exists an ergodic Markov chain (EMC) whose respective window can be used as the Bellwether.*

Proof. The Markov chain is defined as a set of random variables $\{X^{(1)}, \dots, X^{(t)}\}$ or a collection of stochastic events $\{X(t) | t \geq 0\}$ whereby given the present event of a state at time t , the prediction of the future event at $t+1$ is independent of the past events. The *iid* sample space $\{X^{(1)}, \dots, X^{(t)}\}$ for all states at their respective times can be described as a Markov chain if

$$\begin{aligned} P(X_{t+1} = i_{t+1} | X_t = i_t, X_{t-1} = i_{t-1} \dots X_1 = i_1, X_0 = i_0) \\ = P(X_{t+1} = i_{t+1} | X_t = i_t) = p_{ij}^{(k)} \end{aligned} \quad (12)$$

Assume there exists a variable ‘P’ that can be formulated as a matrix of transition probabilities of a Markov chain [30] from the sample space $\{X^{(1)}, \dots, X^{(t)}\}$, where $t \in T$ denotes the project ages, then the ij^{th} element denoted as $p_{ij}^{(k)} \in P$ is the probability that the Markov chain starting from a particular state, t_i will transition to t_j after k steps. If $p_{ij}^{(k)}$ is homogeneous (or regular), then (13) holds and there exists a unique probability matrix,

θ_u that can form the ergodic Markov chain (EMC) such that for any θ_o and for large values of u , (14) can be defined.

$$P(X^{(k)} = j | X^{(k-1)} = i) = p_{ij}^{(k)} \quad (13)$$

$$\lim_{n \rightarrow \infty} \theta_{u+1} = P^u \theta_1 \quad (14)$$

Thus, the EMC can be obtained given a non-negative power of ‘ P ’ by making all entries of the probability matrix non-zero and irreducible (*i.e.*, systematically squaring P till it converges to irreducible or limiting state). If the limiting state probability matrix (*i.e.* the EMC) exists from the TPM constructed with the moving window sample, then that particular sample can be used as the *Bellwether*. Such moving window sample is considered as a *Bellwether* since its limiting or stationary distribution has been reached and hence can form a potential training set for modelling, given it has the best prediction accuracy (see details in chapter 4).

Postulation 6. *Given a Bellwether whose ergodic Markov chain is known, then its size and age can be defined.*

Proof. Let a sample X with projects $\{p_1, \dots, p_n\}$ be classified into t_i states based on their respective project ages. Assume the projects $\{p_1, \dots, p_n\}$ from a particular sample space is sorted in non-decreasing order based on the completion dates of projects and the EMC of the sample is known. Then, the age of the *Bellwether* can be computed as the difference between the maximum and minimum project ages of the sample whose EMC is known.

Similarly, the size of the *Bellwether* can be found as the total number of projects within that particular sample (refer to chapter 4 for details).

3.4. The Bellwether Methods

In order to support the possible replication of this study, this section describes the *Bellwether methods* used to obtain the *Bellwethers* from the studied chronological datasets. That is, the section provides detailed step-by-step procedures to find the existence of *Bellwethers* and how they can be used in training a predictive model for *new* projects. Three *Bellwether methods*, namely *SSPM*, *SysSam* and *RandSam* are used to sample the *Bellwether* projects from the four studied datasets.

3.4.1. Stratification, Sampling and Prediction Method

The stratification, sampling and prediction model (*SSPM*) operates based on a probabilistic approach, namely Markov chain Monte Carlo (MCMC) [30] to sample exemplary projects from a given dataset for prediction purposes. Given a *new* project with unknown software effort, select exemplary projects to be considered as the *Bellwether moving window* (BMW) from a historical dataset (*D*) using the following three main operators (*SORT+CLUSTER*, *GENERATE TPM* and *APPLY*):

SORT + CLUSTER

Given a set of N historical and completed projects from a historical dataset (D), sort based on the project completion dates and stratify the data into q clusters.

1. For all N projects from D , sort in increasing order using their respective project completion dates.
2. Apply the X -means⁵ clustering algorithm [140] to the sorted data to obtain q clusters. Perform data stratification based on the q clusters obtained. That is, stratify N into q clusters whereby each cluster or window (partition) has a set of chronologically arranged projects.
3. For each cluster or window, compute the weighted window by applying the respective weighting functions on all the q windows (see Fig. 2 in chapter 2). We define the resulting q weighted windows in ascending order (with respect to project age) as $w_1, w_2 \dots w_q$.
4. Use w_q as a baseline window. It should be noted that, w_q contains recently completed projects in a chronological order and so can form a *baseline weighted moving window*.

GENERATE TPM

Generate the transition probability matrix (TPM) for the resulting weighted moving window and find the respective ergodic Markov chain.

5. For the resulting baseline window (w_q), generate a transition probability matrix (TPM) of the Markov chain following an approach similar to that reported by Dobrow [30]. Here, we use the project ages⁶ in w_q (as transition states) with their

⁵ X -means automatically estimate the number of optimal clusters with respect to the Bayesian Information Criterion [140].

⁶ Age refers to the elapsed time or duration span of projects that have existed for not more than t calendar years or months. It is computed as the difference between the completion date of a given project (in the *growing portfolio*) and the start date of the hold-out (or *new*) project.

respective targets (*i.e.*, software effort) to generate the TPM. The sum of the p_{ij} in each row of the TPM is approximately equal to 1.

6. Compute the ergodic Markov chain (EMC) for the generated TPM in order to validate the limiting (or stationary) distribution of w_q . Thus, using the TPM with t_i states, successfully perform the squaring of TPM until TPM approaches the limiting or stationary distribution. That is, when individual probability elements, p_{ij} of TPM are positive and cannot be reduced further, we say that TPM is ergodic and regular (or stationary) [30].
7. Report w_q^* as a BMW if the weighted moving window, w_q is stationary and provides the best prediction accuracy for majority of the remaining windows, $w_1, w_2 \dots w_{q-1}$. Else go to step 4 to update w_q with additional project(s) from w_{q-1} and repeat steps 5 and 6. Conversely, update w_q by sequentially removing project(s) from w_q and adding to w_{q-1} and repeating steps 5 and 6.

APPLY

Apply the BMW, w_q^ to the new project data whose target (*i.e.*, software effort) is to be predicted.*

8. Once w_q^* is obtained as a BMW in step 7 prior to its application to the *new* p_i project data (set as a hold-out), its size and age can be defined from its dimensions. The size of the moving window is computed as the total number of projects within the BMW while the age of the moving window is computed as the difference between the maximum and minimum project ages.

9. Predict the target (*i.e.*, software effort) of the p_i project(s) using the resulting BMW.

3.4.2. Benchmark Sampling Methods

Aside of using the *SSPM* to sample *Bellwether* projects, we further explored two additional benchmark sampling methods. These are the *systematic* and *random sampling* methods. Note that the sampled *Bellwethers* from the three *Bellwether methods* (*SSPM*, *systematic* and *random*) were all benchmarked against the *growing portfolio* for setting up the predictive models in the context of software effort estimation.

3.4.2.1. Systematic Sampling (*SysSam*)

This type of sampling method, namely systematic sampling (*SysSam*) refers to the sequential selection of projects from the sorted *growing portfolio* given that the selected projects have the most recent ages. Thus, the sampled projects forming the *moving window* is selected from recently completed projects to oldest completed projects. For example, given that N projects are sorted chronologically (*i.e.*, arranged in ascending order of project completion dates), then systematic sampling (*SysSam*) deals with the sequential selection of recent project cases from bottom (*i.e.*, recently completed) to top (*i.e.*, oldest) of N with respect to the sorted projects. A simple *for loop* is implemented whereby each sequential m_i subset is considered as the training set (potential *Bellwether*) and tested with the remaining $N-m_i$ projects (validation set). In each iteration phase, the mean absolute error (MAE) is used to assess the prediction performance of each sampled subset. The subset with the best prediction performance (*i.e.*, minimum MAE) is used to

predict the p_i projects set as hold-out. Here, we consider the subset with the best prediction performance as the *Bellwether*. We benchmarked the *Bellwether* sampled by *SysSam* against the *Bellwethers* sampled by *SSPM* and the *RandSam* discussed below.

3.4.2.2. Random sampling (*RandSam*)

Random sampling (*RandSam*) refers to the stochastic selection of projects without necessarily following a definite order. Here, a random sampling algorithm is implemented to select project subsets from the *growing portfolio*. The project subsets selected at each iteration did not necessarily contain recently completed projects or had the same sample sizes. Here, each m_i project subset is considered as a training set and used to predict the targets of the remaining $N-m_i$ projects considered as the validation set. Following similar procedure in the *SysSam*, we assess the performance of the random subset selection with MAE and considered the subset with the best prediction accuracy as the *Bellwether*. The *Bellwether* was used to make predictions of the p_i projects. The *Bellwether* sampled by *RandSam* was benchmarked against the *Bellwethers* sampled by *SSPM* and *SysSam* as well as the *growing portfolio*.

3.4.3. Assumptions

The following assumptions are made regarding the effective functioning of the *Bellwether methods*:

- Each project used in modelling has a start and/or completion date. Thus, we pruned off non-dated projects prior to modelling.

- In the case whereby a project lacks the completion date (or start date), the development duration together with the start date (or completion date) is used to compute the completion date or vice versa.
- All sampled projects from N have the same set of features.
- Each p_{ij} element in the transition probability matrix lies within 0 and 1. That is, $0 \leq p_{ij} \leq 1$.
- The sum of each i^{th} row of the transition probability matrix should be approximately 1. That is, $\sum p_{ij} = 1$.
- All entries of the ergodic Markov chain are non-zero.

3.5. Experimental Setup

This study uses the *SSPM* approach together with the *SysSam* and *RandSam* methods (section 3.4) to investigate the selection of exemplary projects to be considered as the *Bellwether*. The sampled *Bellwether* (with defined window size and age) from each *Bellwether method* is benchmarked against the *growing portfolio* for predicting the targets of p_i hold-out projects. Following the *Bellwether methods*, comparison is made across the application of the weighted functions (Triangular, Epanechnikov, Gaussian, Biweight and Triweight) and the unweighted function (Rectangular or Uniform) on the selection partitions in each dataset. For each sampled partition set, we evaluate its accuracy performance (MAE) by using the sampled partition as the training set and performing successive predictions on the remaining unselected partitions (validation set). Thus, for each iteration, the selected partition set is used as the training set and the unselected

partition considered as the validation set until the best partition with relatively minimum MAE is obtained. The best partition set is considered as the *Bellwether* following the detailed procedure in section 3.4. The *Bellwether* benchmarked against the *growing portfolio* is used for predicting the targets of the p_i projects. We used the leave-one-out cross validation [43] as done in our original study [31] to obtain the training and validation sets for the *growing portfolio*. Thus, at each run, $N-1$ projects are used for training the prediction model and the remaining project for validation. The best $N-1$ partition set of the *growing portfolio* is used to predict the p_i projects and benchmarked against the prediction performance of the *Bellwethers*.

Three prediction models or learners are considered in this study, namely the ATLM [5], ElasticNet regression [141] and Deep neural networks [52][101].

3.5.1. The Automatically Transformed Linear Model

The Automatically Transformed Linear Model (ATLM) proposed by Whigham *et al.* [5] is a baseline model in SEE and was developed using multiple linear regression. Multiple linear regression or ordinary least squares (OLS) regression is one of the most popular and widely used SEE models [98][37][34][36]. OLS was considered optimal in an effort estimation study by Dejaeger *et al.* [98]. Whigham *et al.* [5] built on multiple linear regression to propose the ATLM, state-of-the-art baseline SEE model. With regard to ATLM [5] and the OLS by Dejaeger *et al.* [98], the dependent variable is continuous whilst the independent variable(s) can be continuous or discrete. In this study, we use a continuous dependent variable (effort) with its respective effort classes in discrete form

in addition to multiple independent variables for the regression analysis. Hence, the multiple OLS used in this study is defined in (15) as follows:

$$Effort_j = \beta_0 + \sum_{i=1}^p \beta_i x_{ji} + \varepsilon_j \quad (15)$$

where β_0 and β_i are the regression coefficients, x_{ji} is the i^{th} independent (or predictor) variable of the j^{th} project out of a total p independent variables. $Effort_j$ is the dependent (or response) variable of the j^{th} project.

Whigham *et al.* [5] compared ATLM to other complex models such as ensemble methods [23] and a hybrid approach (incorporating particle swarm optimisation, analogy-based estimation and clustering [100]). It was found that, although comparatively simple, ATLM yielded better prediction performance as compared to the other learners and so was considered a baseline model for SEE.

3.5.2. The ElasticNet Regression

The ElasticNet regression proposed by Zou and Hastie [141] is a regularization and feature selection technique which helps in eliminating highly correlated predictor variables from the model. It is a hybrid of ridge regression [142] and least absolute shrinkage and selection operator (LASSO) regression [143]. This is useful when there are multiple predictors which are highly correlated. ElasticNet searches for the optimal correlation coefficients (β) by solving the regularisation and optimisation problem defined by Friedman *et al.* [144] and using β defined in (16) to setup the regression.

$$\hat{\beta} = \arg \min_{b \in \mathbb{R}^{p+1}} \left[\frac{1}{2n} \sum_{i=1}^n (y_i - x_i^T b)^2 + \lambda P_\alpha(b_1, \dots, b_p) \right] \quad (16)$$

$$\text{where } P_\alpha = \sum_{i=1}^p \left[\frac{1}{2} (1 - \alpha) b_j^2 + \alpha |b_j| \right]$$

P_α is the elastic-net penalty [141] which is useful in solving the highly correlated predictor variables and also works best when the number of observations (n) is less than the number of predictors (p) for $\alpha \in (0,1)$ [144]. Ridge regression and LASSO regression are obtained from (16) for $\alpha = 0$ and $\alpha = 1$ respectively [144].

ElasticNet regression was found in our recent study [3] to significantly improve prediction accuracy when benchmarked against five other regression techniques including ATLM. It generates zero-value coefficients (β) and outperforms other regression techniques when the independent variables are highly correlated. Note that details of setting up the *duplex output SEE* models are provided in our previous study [3].

To the best of our knowledge the study by Mensah *et al.* [3] was the first study in software effort estimation to employ the ElasticNet regression which has been successfully used in other domains such as the Biomedical field [145] and image processing [146].

3.5.3. The Deep Neural Network (DNN)

A deep neural network is a feedforward artificial neural network architecture which has more than one hidden layer with their respective interconnection of neurons. Deep learning models have yielded improved prediction accuracy in a range of previous studies [101][52][147][111][148][149][150]. Deep neural networks (DNN) resulted in a relatively higher prediction accuracy in a recent study [31] against two benchmarks,

namely the ATLM and the ordinary least squares [98]. DNN models have yielded improved prediction accuracy in previous studies [101][147][111][148].

We setup a DNN model which makes use of multiple hidden layers and an output layer with their respective neurons to automatically learn from a set of projects and provide the resulting prediction for the target. Thus, given a supervised learning problem with input data (X_i) and its respective label (Y_i), we setup a DNN model with h hidden layers containing a set of neurons and an output layer with a single neuron. The single neuron in the output layer is responsible for *firing* the estimated variable/label (*i.e.*, software effort). Each neuron uses a non-linear activation function to map the output from one neuron in a given layer to an interconnected neuron in the next layer. We made use of the hyperbolic tangent function [151][52] as the activation function in each of the neurons for *firing* the respective outputs based on given input values, z . The hyperbolic tangent function, $\tanh(z)$ defined in (17) has a *firing* output range of $(-1, 1)$ that prevents the DNN model getting *stuck* during the training process. That is, unlike the sigmoid activation function with an output range of $(0, 1)$ that results in an output value close to 0 when the z value is negative, $\tanh(z)$ is able to output the corresponding negative value for a given z due to its wide range of output $(-1, 1)$. This makes the use of the $\tanh(z)$ an efficient activation function when setting up the DNN.

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{e^{2z} - 1}{e^{2z} + 1} \quad (17)$$

In order to obtain the best DNN model, we started with a single hidden layer and an output layer each containing a single neuron and iterated through a series of fine-tuning processes to obtain the best model parameters at a minimal loss. At each iteration, the

weights w_{ij} connecting the neurons were initialised to 0. The new weights $w_{ij}^{(t+1)}$ were updated after each iteration process using the function defined in (18) whereby the change $\Delta w_{ij}^{(t+1)}$ in weight for each new weight is defined in (19). The parameter t denotes the level of iteration, α is the learning rate, γ is the momentum and $L(\cdot)$ is the loss function or the error between the predicted output and the actual label. Note that the Levenberg-Marquardt backpropagation optimisation [152][153] training function was employed to update the weights of the neurons in the hidden and output layers respectively. This training function, which is a supervised learning algorithm, is relatively efficient [154] when training with small to moderate-sized datasets.

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t+1)} \quad (18)$$

$$\Delta w_{ij}^{(t+1)} = \gamma \Delta w_{ij}^{(t)} - \alpha \frac{\partial L(\cdot)}{\partial w_{ij}} \quad (19)$$

The choice of the learning rate (α) can significantly affect the performance of a DNN during the training process. Thus, there is the need for an optimal selection of α in order for the DNN to converge to relatively better global optima. After commencing the training of the DNN with a relatively large learning rate of $\alpha = 0.1$, we ended up with a relatively low learning rate of $\alpha = 0.01$ which yielded the best performance at 100 training epochs. Note that if the learning rate is too small (*e.g.* $\alpha = 0.001$), the gradient descent can be slow and training will take a relatively long time to converge. Alternatively, if α is too large (*e.g.* $\alpha = 0.2$), the gradient descent can wave-off the minimum and training might fail to converge. We recommend training of the DNN to commence with a relatively large learning rate and then gradually decrease it to a relatively lower learning rate to obtain the best prediction accuracy. The momentum (γ) is used for smoothing out the weight updates

in the interconnected neurons. Even though momentum also affects the training process, its impact is less significant relative to the learning rate. The best momentum value used in our DNN was $\gamma = 10^{-6}$ after commencing the training process with a $\gamma = 0.1$ [127].

After iteratively fine tuning the hidden layers with their respective neurons, we found that the best DNN that resulted in improved prediction accuracy *wrt* to MAE was a 2-hidden layered architecture with 5 and 2 neurons respectively, in addition to an output layer with a single neuron. Note that the sampled data from each *Bellwether method* was considered as the training set and the remaining un-sampled data as the validation set.

3.6. Performance Measures

We employ a robust performance measure (*i.e.*, the mean absolute error (MAE)) which has been proven reliable by Shepperd and MacDonell [47] and Foss *et al.* [113] for setting up SEE models. MAE has been considered in previous studies [34][36][3][5][7][44] to evaluate prediction accuracy of SEE models. It is a risk function that measures the average absolute deviation of the estimated effort (E_E) values from the actual or true effort (E_A) values and it is defined in (20). Minimum values from the performance measures are considered superior in terms of prediction accuracy.

$$MAE = \frac{1}{n} \sum_{i=1}^n |E_{Ai} - E_{Ei}| \quad (20)$$

The robust statistical tests employed in this study for determining the statistical differences among the weighted/unweighted moving windows and *growing portfolio* are the Yuen's test, Brunner's test and Cliff's δ effect size. Statistical significant differences

are considered at $\alpha = 0.05$ asymptotic significance level as well as a confidence interval of 95%. Details of the aforementioned statistical tests are provided in chapter 2.

Chapter 4

Results and Discussion

This chapter presents the empirical results and discussion of the existence of *Bellwether* against the *growing portfolio* benchmark in each of the four studied datasets – the Desharnais dataset, the Kitchenham dataset, the ISBSG dataset and the Maxwell dataset. The results are presented based on the three *Bellwether methods* (*SSPM*, *SysSam* and *RandSam*) and performance evaluation assessed using mean absolute error (MAE), Cliff's δ effect size, Yuen's test and Brunner's test at the 5% asymptotic significance level.

4.1. Bellwether effect in chronological datasets

1) Desharnais Dataset

Results from the *SSPM* show that after subjecting the sorted chronological dataset to the *X-means* clustering algorithm, three partitions (with initial approximate sizes of 41, 23 and 8 projects respectively) were recorded. These initial partitions were used to predict the targets of the p_i projects in order to validate Postulation 1 (see section 3.3). We found that the prediction probabilities of the three partitions with respect to the p_i projects were different and independent from each other. In order to empirically investigate the existence of a *Bellwether effect* in the Desharnais dataset, we followed the *SSPM* procedure as well as the *SysSam* and the *RandSam* methods as elaborated in chapter 3.

The partition containing 41 recently completed projects relative to partition 2 and 3 was used as the initial baseline window (in the case of *SSPM*). In each iteration, the window size was updated and the respective transition probability matrix computed. We assessed the prediction performance using MAE for each updated window in each iteration. Prediction performance results of the stationary and potential *Bellwethers* with known ergodic Markov chains are recorded in Table 6. Similarly, the best prediction performance results of the sampled *Bellwethers* from the *SysSam* and *RandSam* methods are also recorded in Table 6. It should be noted that three learners namely, the ATLM, ElasticNet and Deep neural networks (DNN) were used in this study. In all cases, it was found that the DNN yielded superior prediction accuracy as compared to the ATLM and the ElasticNet. Thus, results from the DNN learner evaluated based on the MAE are recorded in Table 6.

Results of the weighted/unweighted windows (benchmarked against the *growing portfolio*) which yielded the best prediction accuracy for each sampling approach are summarised in Table 6. For the sake of easy understanding of the summarised results, we used shaded or *greyed cells* to denote the best prediction window across the six weighting/unweighting functions. Note that comparisons are made across the window ages and sizes for each sampling approach (*SSPM*, *SysSam* and *RandSam*). Lastly, we used ‘**’ to further denote the window which can act as the *Bellwether* in each of the sampling approaches. For example, in the case of the *SSPM*, we found that the Triweight window (BMW) of size 46 exemplary projects and age 2 years resulted in the best prediction accuracy (MAE) of 0.3170 as compared to the remaining windows and the

growing portfolio benchmark. Similarly, in the case of the *SysSam* method, we realised that the Triweight window of size 44 projects and age 1.5 years with MAE of 0.6241 constituted the *Bellwether*. Lastly, the Triweight window of size 48 projects and age 2 years with MAE of 0.8549 constituted the *Bellwether* in the case of the *RandSam* method.

We found that exemplary projects forming the *Bellwether* in the Desharnais dataset should not be older than 2 years and their window sizes should be in a range of 44 to 48 projects. In all three cases of the *Bellwether methods*, we found that the *Bellwether* yielded a relatively superior prediction accuracy as compared to the *growing portfolio* (GP).

Table 6 MAE evaluation of potential *Bellwethers* sampled by *SSPM*, *SysSam* and *RandSam* against the *growing portfolio* benchmark in the Desharnais dataset

Window		Stratification, Sampling and Prediction Method (<i>SSPM</i>)					
Age (years)	Size	Rectangular	Triangular	Epanechnikov	Gaussian	Biweight	Triweight
1	38	1.6926	1.2109	1.8054	0.9257	1.0748	0.8947
1.5	42	0.9914	1.5862	0.7931	0.7035	0.8675	0.7184
2	46	0.8351	1.5682	0.7841	0.9801	0.8577	0.3170**
2	47	1.1288	1.2563	0.8628	1.0785	0.9437	0.7089
5	71(GP)	2.4018	2.9906	2.4953	1.8691	1.6355	1.2318
Age	Size	Systematic Sampling Approach (<i>SysSam</i>)					
1.5	40	1.8126	1.5444	1.0889	0.8056	0.9552	0.7104
1.5	44	1.7410	1.4964	0.9929	1.2055	1.4300	0.6241**
2	45	1.9154	2.0302	1.0604	1.2878	1.1268	0.9658
2	49	1.0103	1.6904	1.3809	0.6504	0.7552	0.8473
5	71(GP)	2.4018	2.9906	2.4953	1.8691	1.6355	1.2318
Age	Size	Random Sampling Approach (<i>RandSam</i>)					
1.5	36	1.9190	1.2626	1.5251	1.3282	1.2872	1.2462
2	41	1.1594	1.0678	2.1356	1.3347	1.1680	1.0011
2	45	1.7626	1.2521	1.5042	0.3151	1.2757	1.2363
2	48	1.1361	0.9119	1.8239	1.1399	0.9975	0.8549**
5	71(GP)	2.4018	2.9906	2.4953	1.8691	1.6355	1.2318

GP denotes the growing portfolio.

Greyed cells used to indicate best prediction performance across weighted/unweighted windows for each aged/sized sample set. Thus, the minimum MAE signifies the best prediction accuracy.

*** denotes the best weighted/unweighted window for each sampling approach (*SSPM*, *SysSam* and *RandSam*). That is, the window with the minimum MAE for all greyed cells per each sampling approach.

Result from the Desharnais dataset supports result from our previous study [31] that *Bellwethers* are evident and yield a relatively higher prediction accuracy against the *growing portfolio* benchmark.

2) Kitchenham Dataset

Results from the *SSPM*, *SysSam* and *RandSam* for this dataset are summarised in Table 7. Here, we found that among all the three *Bellwether methods*, the Triweight window selected by the *SSPM* with window size of 83 exemplary projects and age of 2 years provided superior prediction accuracy (*i.e.*, MAE of 0.5267). Thus, the Triweight window (BMW) can result in improved prediction of *new* projects as compared to the other weighted/unweighted windows and the *growing portfolio*.

Table 7 MAE evaluation of potential *Bellwethers* sampled by *SSPM*, *SysSam* and *RandSam* against the *growing portfolio* benchmark in the Kitchenham dataset

Window		Stratification, Sampling and Prediction Method (<i>SSPM</i>)					
Age (years)	Size	Rectangular	Triangular	Epanechnikov	Gaussian	Biweight	Triweight
1.5	76	1.2534	1.3421	0.8437	0.7666	1.0742	0.8207
2	83	1.0532	1.0882	0.7637	0.9102	1.0965	0.5267**
2	86	1.1006	1.0430	0.8617	0.8147	0.8521	0.7216
2	87	0.9507	1.2288	1.4577	0.8604	1.2503	0.6453
3	134(GP)	3.2410	2.5300	2.0601	1.6625	2.5797	2.4969
Age	Size	Systematic Sampling Approach (<i>SysSam</i>)					
2	80	1.3978	1.5370	1.0740	1.2607	1.0874	0.9034
2	82	1.4591	1.6198	1.2396	1.1748	1.1779	1.0811
2	84	1.6481	1.8749	1.7497	1.0936	0.9569	0.8202**
2	89	1.8680	1.1718	1.3436	1.4647	1.2817	1.0985
3	134(GP)	3.2410	2.5300	2.0601	1.6625	2.5797	2.4969
Age	Size	Random Sampling Approach (<i>RandSam</i>)					
2	80	2.6738	1.5264	2.0529	1.4080	1.6696	1.3103
2	81	1.6198	0.7169	1.4339	0.8962	0.7842	0.6721**
2	82	1.8205	1.5922	2.1845	1.9903	1.7416	1.4927
2	83	2.1799	2.7044	2.4089	1.3054	1.8643	1.5979
3	134(GP)	3.2410	2.5300	2.0601	1.6625	2.5797	2.4969

Results for the remaining weighted/unweighted windows and the *growing portfolio* are summarised in Table 7. With regard to the *growing portfolio* (GP), we found that the Gaussian GP resulted in improved prediction accuracy as compared to the remaining weighted/unweighted GP.

Similar to the original study [31], there exists exemplary projects (with defined window size and age) that yields a relatively improved prediction accuracy irrespective of the *Bellwether method* used. Also result validates a previous study by Kitchenham *et al.* [42] that there exists a moving window sampled from this dataset that provides a better prediction accuracy as compared to the *growing portfolio*. Note that results from the DNN learner is recorded in Table 7 since it resulted in superior prediction accuracy.

3) ISBSG Dataset

Results from the ISBSG dataset summarised in Table 8 show that the Triweight window selected by the *SSPM* yielded the best prediction accuracy as compared to those selected by *SysSam* and *RandSam*. The Triweight window (*SSPM*) has a window size of 84 exemplary projects and an age of 2 years with a minimum MAE value of 0.6325 (Table 8). This confirms results from previous study [31] that the BMW should not be older than 2.5 years relative to the p_i projects. Note that the original study used multi-organisation projects from the ISBSG repository and found that the BMW should have a window size of 257 exemplary projects with an age of 2.5 years old. With the exception of one Gaussian window in the case of the *SysSam* that yielded a relatively improved prediction, the Triweight window was the best across all windows.

Table 8 MAE evaluation of potential *Bellwethers* sampled by *SSPM*, *SysSam* and *RandSam* against the *growing portfolio* benchmark in the ISBSG dataset

Window		Stratification, Sampling and Prediction Method (<i>SSPM</i>)					
Age (years)	Size	Rectangular	Triangular	Epanechnikov	Gaussian	Biweight	Triweight
1.5	82	1.1123	1.2020	1.4039	1.0244	1.3147	0.9268
2	83	1.2013	1.4199	1.8398	1.7749	1.5531	1.0312
2	84	1.1438	1.5387	2.0774	0.9234	1.0830	0.6325**
2.5	86	1.0178	1.9018	1.8036	1.1887	1.0802	0.9829
8	107(GP)	4.6061	3.6486	2.2971	2.8107	3.7094	2.1080
Age	Size	Systematic Sampling Approach (<i>SysSam</i>)					
1.5	81	2.9465	2.6376	3.2752	2.2970	2.8850	2.4728
1.5	82	2.2105	2.2522	4.5045	2.8153	2.4635	2.1115
1.5	83	2.3193	2.3416	2.6832	1.4270	2.3737	1.3203**
2	84	3.2292	3.0658	2.9053	2.4527	2.6827	2.2994
8	107(GP)	4.6061	3.6486	2.2971	2.8107	3.7094	2.1080
Age	Size	Random Sampling Approach (<i>RandSam</i>)					
1.5	79	1.6080	1.1351	2.2701	1.4188	1.2415	1.0641
2	86	2.0130	1.3918	2.7837	1.7398	1.5224	1.3049
2.5	86	3.0188	2.0160	2.0319	2.5200	2.2051	1.8900
2.5	87	1.7900	0.8453	1.6907	1.0567	0.9246	0.7925**
8	107(GP)	4.6061	3.6486	2.2971	2.8107	3.7094	2.1080

It can be observed from Table 8 that the application of the Gaussian weighting function also resulted in improved prediction accuracy (but not as compared to the Triweight function) as observed in the original study [31]. With regard to the ISBSG dataset, the DNN resulted in superior prediction accuracy followed by the ElasticNet and the ATLM. Hence, results presented in Table 8 are based on using the DNN and evaluation done using the MAE.

4) *Maxwell Dataset*

Results from the *SSPM* show that after subjecting the sorted chronological dataset to the *X-means* algorithm, two partitions were recorded. The initial partition sizes (number of

instances per cluster) were 30 projects (52.6%) and 27 projects (47.4%) with a BIC⁷ of -32.619. Thus, the *growing portfolio* of 57 sorted projects was partitioned into 30 and 27 (steps 1-2 of the *SSPM*). We made use of the 27 projects (containing more recently completed projects relative to the 30 projects) as the initial baseline window for training the three learners. The remaining 30 projects acted as the initial validation set after the application of weighting functions following steps 3 to 4.

We present the potential BMWs benchmarked against the *growing portfolio* across the 6 weighting/unweighting functions. We found that, in all cases of benchmarking the potential BMW against the *growing portfolio* (GP), the BMW yielded relatively superior prediction accuracy. On average, we also found that when the Triweight function was applied to the BMW and GP, the prediction accuracy was relatively improved as compared to the other functions across all three learners. The potential BMWs were of respective sizes ranging from 39 to 47 projects (68.4% - 82.5%) as recorded in Table 9. These windows were not older than 2 to 3 years respectively. That is, the BMW(39) has a window age of 2 years, BMW(40) ~ 2 years, BMW(41) ~ 2 years, BMW(42) ~ 2 years, BMW(43) ~ 3 years and BMW(47) ~ 3 years. Out of these 6 potential BMWs, we found that on the average, BMW(43) ~ 3 years resulted in relatively superior prediction accuracy. On average, it was found that the application of the Triweight function on the BMW yielded improved prediction accuracy. Note that the deep neural network (DNN) model was used for setting up the prediction model.

⁷ A Bayesian Information Criterion (BIC) with a relatively low value is preferred for an effective modelling.

Table 9 MAE evaluation of potential *Bellwethers* sampled by *SSPM*, *SysSam* and *RandSam* against the *growing portfolio* benchmark in the Maxwell dataset

Window		Stratification, Sampling and Prediction Method (<i>SSPM</i>)					
Age (years)	Size	Rectangular	Triangular	Epanechnikov	Gaussian	Biweight	Triweight
2	39	3.3116	4.0982	2.5551	3.1830	3.3021	2.3816
2	40	4.3964	4.3662	3.1703	3.4492	3.5501	2.5112
2	41	4.4661	4.4362	3.2685	3.3898	3.4506	2.1717
2	42	3.5549	4.0253	3.4278	3.1534	3.0097	2.2005
3	43	4.2742	4.1452	3.4478	3.0308	3.4560	2.0819
3	47	4.0356	4.0114	3.2312	2.5658	3.0283	2.7892
9	57 (GP)	7.0526	9.6895	6.7545	6.5615	6.9400	5.7277
Age	Size	Systematic Sampling Approach (<i>SysSam</i>)					
2	39	4.0021	3.8116	3.4011	4.1580	3.5501	3.8300
2	42	4.5097	4.8980	4.4506	3.1717	3.3021	2.8160
3	43	3.4560	3.1534	3.5097	4.2005	2.5112	2.4100
3	47	4.4283	4.3308	5.2560	4.0819	3.4506	3.9492
9	57 (GP)	7.0526	9.6895	6.7545	6.5615	6.9400	5.7277
Age	Size	Random Sampling Approach (<i>RandSam</i>)					
2	39	4.5526	3.8950	4.2545	4.0615	3.8643	3.0715
2	42	5.6398	3.5639	4.2303	3.5550	2.6895	2.0526
3	43	3.5540	3.6257	3.1281	2.9010	1.7545	1.5615
3	47	4.2312	3.5658	3.0283	2.7892	2.3717	2.1478
9	57 (GP)	7.0526	9.6895	6.7545	6.5615	6.9400	5.7277

With regard to using the *SysSam* and *RandSam*, it was found that on average the Triweight function applied on the *Bellwether* of 43 projects ~ 3 years resulted in improved prediction accuracy. Thus, in all cases of the three *Bellwether methods* (i.e., *SSPM*, *SysSam* and *RandSam*), the *Bellwether* of 43 projects not older than 3 years yielded superior prediction accuracy as compared to the other *Bellwethers*. This result improves on previous inference made in an original study by Mensah *et al.* [31] that *Bellwethers* exist in a population of size not less than 100 projects. Thus, even though the Maxwell dataset has a population size of 62 projects, we found that *Bellwethers* were not rare in

such a small-sized dataset. In other words, *Bellwethers* exist in a small-sized population of size less than the 100-project threshold in the original study [31].

It may thus be deduced from the Maxwell dataset that the minimum population size for obtaining the BMW should not be less than the size of the *growing portfolio* provided there exists such exemplary projects with improved prediction accuracy. In this particular study, a minimum threshold of 57 projects (*i.e.*, *growing portfolio* size in the Maxwell dataset) was required to obtain the BMW.

Results show that there exists exemplary projects in the four studied datasets which can form the Bellwether (BMW). Such exemplary projects should not be older than 3 years relative to the p_i hold-out projects and should have window size of 39 to 84 projects for improved prediction accuracy. The BMW sampled by the SSPM yielded a relatively superior prediction accuracy as compared to the SysSam and RandSam methods.

4.2. Statistical significance of Bellwethers with respect to weighting/unweighting functions

We performed empirical analysis to further investigate whether *there exists statistical significant differences across the weighted/unweighted windows*. Thus, previous studies [34][41] found that prediction accuracy is affected by different weighting functions. Therefore there is the need to further investigate whether or not this assessment made in previous studies is true. We first used Brunner's method to test the alternative hypothesis for each of the four datasets examined. Here, we found that there exists significant differences across the weighted/unweighted windows benchmarked against the *growing portfolio*. This was due to the fact that the *F-values* from the *Brunner's* ANOVA-like test were all significant at 5% asymptotic significance level in the four datasets. Thus, *F-*

values were $F_1 = 332.68$ ($p < 0.05$), $F_2 = 554.39$ ($p < 0.05$), $F_3 = 168.64$ ($p < 0.05$) and $F_4 = 598.53$ ($p < 0.05$) for the Desharnais, Kitchenham, ISBSG and Maxwell datasets respectively.

We further performed pairwise statistical tests between the weighted/unweighted windows and the *growing portfolio* since the Brunner's test resulted in significant differences across the windows and the *growing portfolio*. This was to find out if the *Bellwether* yielded improved prediction accuracy across all the four datasets. The pairwise statistical tests were done using Yuen's test and Cliff's δ effect size as recorded in Table 10. Since the *Bellwether* sampled by the *SSPM* yielded relatively better prediction performance (*i.e.*, minimum MAE) on average (in Table 6), the pairwise statistical test was evaluated between *SSPM* and the *growing portfolio* (Table 10).

Results from Table 10 show that there are statistical and practical significant differences between the weighted/unweighted *Bellwether* and the *growing portfolio* across all four datasets. Thus, both p (< 0.05) and δ (≥ 0.276) values for the Yuen's test and Cliff's δ effect size (respectively) are all significant at 5% asymptotic significance levels. In order to know the *Bellwether* or *growing portfolio* that yielded the best prediction accuracy in each paired comparison test, we made use of the *trimmed means* of their respective MAE values. That is, we evaluated the prediction performances of the *Bellwether* and *growing portfolio* and averaged their respective MAE values. The minimum *trimmed mean* yielded the best prediction accuracy. Note that LHS denotes the *trimmed mean* of the weighted/unweighted *Bellwether* (BMW) and RHS denotes the *trimmed mean* of the *growing portfolio* (Table 10).

Table 10 Yuen's test and Cliff's δ effect size for pairwise differences between the weighted/unweighted *Bellwether* (SSPM) and *growing portfolio*

Window	Trimmed mean		Yuen's test		Cliff's δ effect size
	LHS	RHS	t-value	p-value	
Desharnais dataset					
<i>BMW(Tw)</i> vs. <i>GP(R)</i>	8.0903	15.2934	-8.7691	0.0368	0.9243*
<i>BMW(Tw)</i> vs. <i>GP(T)</i>	8.0903	15.7650	-13.1157	0.0119	0.4453*
<i>BMW(Tw)</i> vs. <i>GP(E)</i>	8.0903	24.3122	-27.7224	0.0323	0.9925*
<i>BMW(Tw)</i> vs. <i>GP(G)</i>	8.0903	12.4660	9.6117	0.0164	0.3950*
<i>BMW(Tw)</i> vs. <i>GP(B)</i>	8.0903	16.4730	-14.3255	0.0278	0.6120*
<i>BMW(Tw)</i> vs. <i>GP(Tw)</i>	8.0903	14.6436	-12.3096	0.0345	0.4324*
Kitchenham dataset					
<i>BMW(Tw)</i> vs. <i>GP(R)</i>	7.1480	15.0463	-14.3050	0.0178	0.3604*
<i>BMW(Tw)</i> vs. <i>GP(T)</i>	7.1480	14.2752	-17.5349	0.0208	0.9812*
<i>BMW(Tw)</i> vs. <i>GP(E)</i>	7.1480	22.3427	-37.3831	0.0362	0.3407*
<i>BMW(Tw)</i> vs. <i>GP(G)</i>	7.1480	12.8766	8.0414	0.0409	0.4175*
<i>BMW(Tw)</i> vs. <i>GP(B)</i>	7.1480	14.9382	-19.1659	0.0362	0.6441*
<i>BMW(Tw)</i> vs. <i>GP(Tw)</i>	7.1480	13.8335	-16.4482	0.0273	0.5832*
ISBSG dataset					
<i>BMW(Tw)</i> vs. <i>GP(R)</i>	6.4601	11.3314	9.4064	0.0222	1.2842*
<i>BMW(Tw)</i> vs. <i>GP(T)</i>	6.4601	13.1866	-10.5093	0.0149	0.3746*
<i>BMW(Tw)</i> vs. <i>GP(E)</i>	6.4601	20.0357	8.7005	0.0378	1.4610*
<i>BMW(Tw)</i> vs. <i>GP(G)</i>	6.4601	13.7353	-11.4902	0.0118	0.6054*
<i>BMW(Tw)</i> vs. <i>GP(B)</i>	6.4601	14.6537	-7.6912	0.0375	0.4308*
<i>BMW(Tw)</i> vs. <i>GP(Tw)</i>	6.4601	12.8210	-10.0464	0.0317	0.6912*
Maxwell dataset					
<i>BMW(Tw)</i> vs. <i>GP(R)</i>	4.1452	7.0526	-3.9822	0.0189	0.4421*
<i>BMW(Tw)</i> vs. <i>GP(G)</i>	4.1452	6.5615	-0.8315	0.0295	0.3430*
<i>BMW(Tw)</i> vs. <i>GP(B)</i>	3.4478	6.5615	1.1830	0.0268	0.5450*
<i>BMW(Tw)</i> vs. <i>GP(E)</i>	3.4478	6.7545	-0.8478	0.0057	0.4950*
<i>BMW(Tw)</i> vs. <i>GP(T)</i>	3.0308	9.6895	2.2083	0.0158	1.0120*
<i>BMW(Tw)</i> vs. <i>GP(Tw)</i>	3.0308	6.5615	2.0609	0.0275	0.6270*

Growing portfolio (GP); Bellwether moving window (BMW); Rectangular (R); Triangular (T); Epanechnikov (E); Gaussian (G); Biweight (B); Triweight (Tw)

Statistical Significance: $p < 0.05$; Practical Significance: $*\delta \geq 0.276$

LHS denotes the trimmed MAE of the window on the left hand side in the first column; RHS denotes the trimmed MAE of the window on the right hand side

Again, consider the pairwise comparison between the Triweight BMW(Tw) and the Rectangular GP(R) in the first row of Table 10 with regard to the Maxwell dataset. It

can be seen that the Cliff's δ effect size value of 0.4421 ($\delta > 0.276$) validates the statistical significant difference between the two weighted windows. The *trimmed* MAE of 4.1452 (on the LHS) corresponding to BMW(Tw) is less than that of GP(R) with a *trimmed* MAE of 7.0526 (on the RHS). Hence, the BMW(Tw) yielded a relatively higher prediction accuracy as compared to the GP(R). Note that results presented in Table 10 are associated with the DNN learner since it resulted in improved prediction accuracy relative to the ATLM and ElasticNet [31]. Similar result was observed for the remaining paired comparison tests as presented in Table 10.

Results show that the Triweight Bellwether (BMW) sampled by the SSPM was relatively significant wrt prediction accuracy as compared to the weighted/unweighted growing portfolio and the remaining windows. Validation of results assessed using Brunner's ANOVA-like test, Yuen's test and Cliff's δ effect size at 5% asymptotic significance level showed that Bellwethers yield superior prediction accuracy.

4.3. Prediction of the p_i hold-out projects by the Bellwether and the growing portfolio benchmark

To investigate how significant the *Bellwether* yielded with respect to prediction accuracy, we benchmarked its prediction performance against the *growing portfolio*. This provided further evidence of considering *Bellwethers* when predicting the targets of p_i projects. As recorded in Tables 6-9, the Triweight *Bellwether* yielded the best prediction performance and hence was used for predicting the targets of the p_i projects. Results are summarised in Fig. 4-7 for the Desharnias, Kitchenham, ISBSG and Maxwell datasets respectively.

Note that, the *Bellwethers* sampled by the *SSPM* in each chronological dataset was used for the prediction of the p_i projects since it resulted in superior accuracy. With regard to predicting the targets (effort) of p_i in the Desharnais dataset, we found that the *Bellwether* (BMW) yielded improved prediction accuracy across all the five p_i projects (Fig. 4). That is, the predicted effort by the BMW (circular shapes) resulted in similar effort values as compared to the *actual* effort values (diamond shapes) of the p_i hold-out projects. Alternatively, the predicted effort of the p_i 's by the *growing portfolio* (GP) denoted by square shapes resulted in less prediction accuracy relative to the BMW. Similar results were observed in the Kitchenham dataset (Fig. 5), the ISBSG dataset (Fig. 6) and the Maxwell dataset (Fig. 7).

We further performed statistical tests using Yuen's test and Cliff's δ effect size to confirm the significant differences between the prediction results of the BMW and GP relative to the actual effort values of p_i . As illustrated in Table 11, we found that there were no statistical and practical significant differences between the prediction results by the BMW and the actual effort values of the p_i 's in all four studied datasets.

Results show that the Bellwether (BMW) yielded a relatively superior prediction accuracy in estimating the targets of hold-out projects (p_i). We validated the prediction of the p_i using the Yuen's test and Cliff's δ effect size.

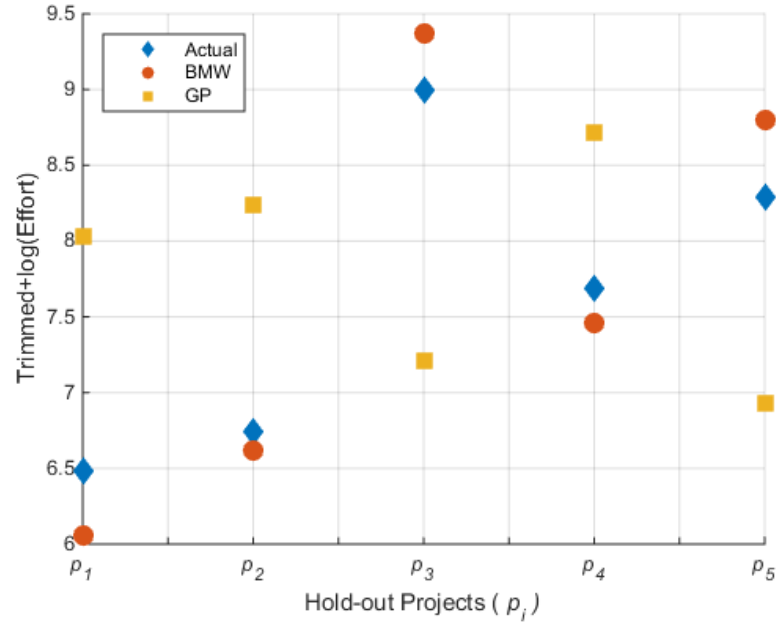


Figure 4 Prediction of the p_i hold-out projects by the Bellwether (BMW) and the growing portfolio (GP) in the Desharnais dataset.

Note: The diamond shape denotes the actual effort of p_i as recorded in the dataset. The circular shape denotes the predicted effort of p_i by the Bellwether (BMW) and the square shape denotes the predicted effort of p_i by the growing portfolio (GP).

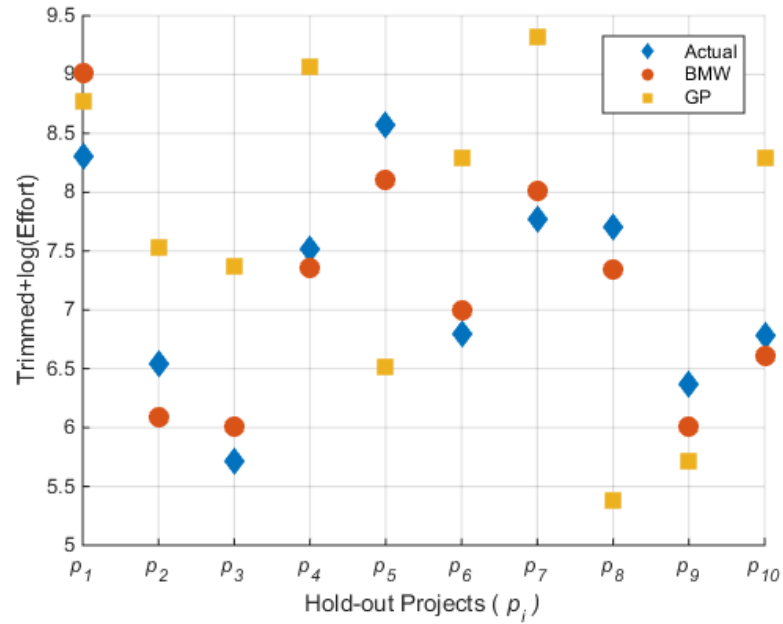


Figure 5 Prediction of the p_i hold-out projects by the Bellwether (BMW) and the growing portfolio (GP) in the Kitchenham dataset.

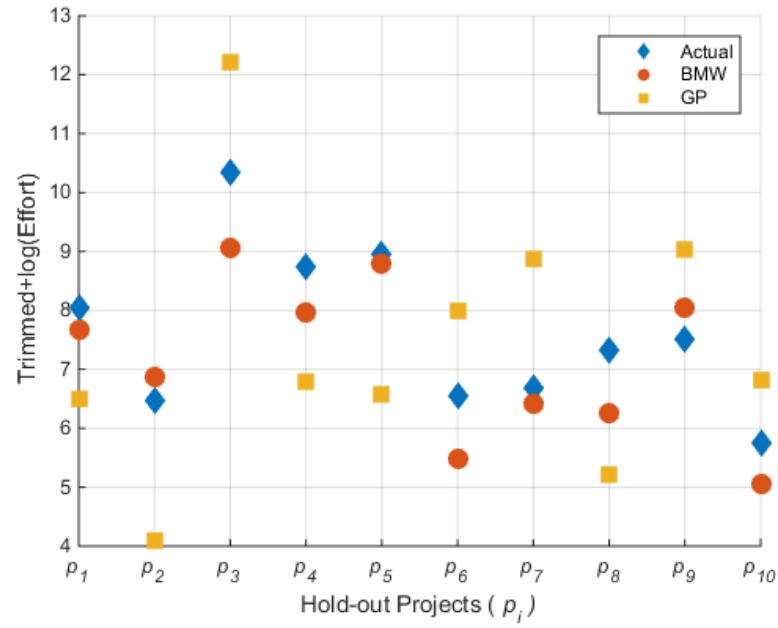


Figure 6 Prediction of the π_i hold-out projects by the Bellwether (BMW) and the growing portfolio (GP) in the ISBSG dataset.

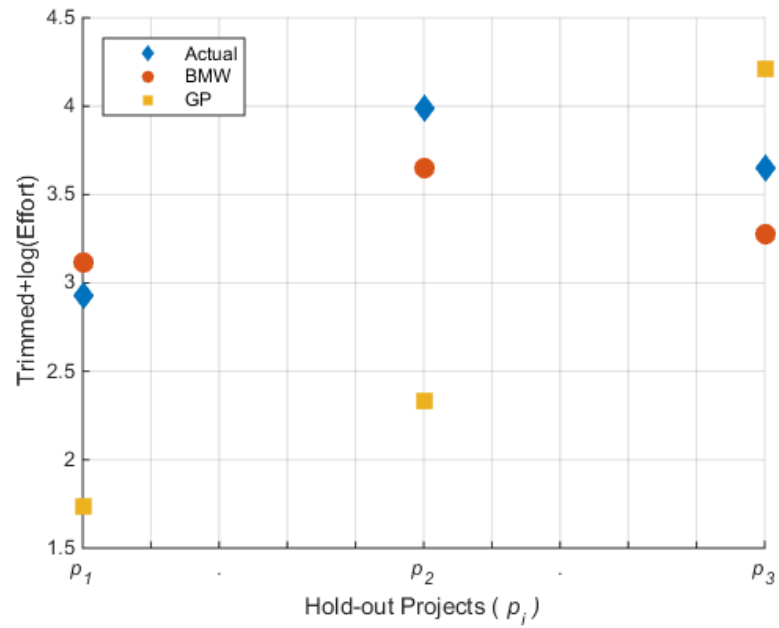


Figure 7 Prediction of the π_i hold-out projects by the Bellwether (BMW) and the growing portfolio (GP) in the Maxwell dataset.

For example, the Yuen's t -test between the predicted effort by BMW and the actual effort of p_i in the Desharnais dataset shows no statistical significant difference at 5% asymptotic significance level (*i.e.*, $p = 0.9058 > 0.05$). The effect size between the BMW and the p_i 's is $\delta = 0.1789$ which is less than the threshold of 0.276 hence the difference is not practically significant (Table 11). Alternatively, we found statistical and practical significant difference between the *growing portfolio* (GP) and the p_i (Table 11). This shows that the BMW yielded a superior prediction accuracy of the targets of the hold-out projects (p_i).

Table 11 Statistical significant differences between prediction results of the *Bellwether* (BMW) and the *growing portfolio* (GP) relative to p_i

BMW/GP vs p_i	Yuen's test		Cliff's δ effect size
	t -test	p -value	
Desharnais dataset			
BMW vs p_i	0.1260	0.9058	0.1789
GP vs p_i	3.4149	0.0269*	0.8748**
BMW vs GP	-2.8552	0.0462*	1.0384**
Kitchenham dataset			
BMW vs p_i	-0.4416	0.6692	0.1241
GP vs p_i	2.8966	0.0177*	0.4898**
BMW vs GP	-3.3441	0.0086*	0.4407**
ISBSG dataset			
BMW vs p_i	-1.3263	0.2174	0.2113
GP vs p_i	2.8355	0.0195*	0.6265**
BMW vs GP	-3.0039	0.0149*	0.6847**
Maxwell dataset			
BMW vs p_i	-0.4722	0.5667	0.1660
GP vs p_i	2.4730	0.0263*	0.5734**
BMW vs GP	1.1215	0.0408*	0.3623**

Statistical Significance: $p < 0.05^*$ Practical Significance: $\delta \geq 0.276^{**}$

4.4. Guideline for Sampling Bellwethers

This section introduces the following guideline for the practical usefulness and sampling of *Bellwether moving windows* (BMW) to be considered for prediction purposes. These guidelines are validated using the four studied datasets.

- The BMW will continue to be useful for prediction purposes provided the phenomenon being measured shares similar characteristics with the exemplary projects. In other words, if the *new* project to be developed does not share similar characteristics (such as the number of features, development and testing procedures) with the projects within the BMW, then the BMW might not be useful for predictive modelling.
- The feature set, f_{new} of the *new* project whose software effort is to be predicted should be a subset of the feature set, f_{bell} of the *Bellwether moving window* (BMW). That is, $f_{new} \subseteq f_{bell}$. Thus, f_{new} should have the same or a subset of *prior features* in f_{bell} when seeking to use the BMW to predict the software effort of *new* project(s). For example, given that the *prior features* such as $f_{bell}^{(1)}, \dots, f_{bell}^{(n)}$ are considered during the training/validation process for obtaining the BMW, then the features, f_{new} of the *new* project should have the same features or a subset of features considered in the selection of the BMW. That is, $f_{new}^{(i)} \subseteq f_{bell}^{(j)}$ where $i \leq j$ and $i, j \neq k$. k is a set of *posterior features* (e.g. lines of code (LOC), number of source files, number of commits, and the like) that is not considered during the training/validation process, and therefore cannot be used when predicting the software effort of *new* project(s).

- *Posterior features* are not considered for sampling the BMW since the *new* project to be predicted does not share such characteristics in the initial stage of development. We therefore recommend the use of *prior features* for obtaining the BMW. *Prior features* refer to the independent variables known before (or at the beginning of) the development of a *new* project whilst *posterior features* refer to the independent variables known after the development of a project [101]. Examples of *prior features* are function points, number of developers, experience of development team, development platform, *etc* and examples of *posterior features* are LOC, number of source files, total number of commits, number of defects, *etc*.
- Historical project datasets can be *noisy* and heterogeneous which affects data quality [155] and as such, there is a need to preprocess the data [126] prior to sampling the BMW. We recommend the use of *trimming + log transformation* as an effective data preprocessing method for obtaining the BMW. This method has proved an effective approach in our empirical analysis resulting in improved prediction accuracy.
- Historical project datasets collected from different industries or organisations are heterogeneous in nature with different methodological, development and security policies. As a result, single organisational projects which share similar characteristics with the p_i hold-out projects are to be preferred for sampling the BMW for improved prediction accuracy. In cases where an organisation lacks historical projects, the introduced *Bellwether methods* (*SSPM*, *SysSam* and

RandSam) can be adopted to sample cross-company projects as shown using the Desharnais dataset (in this study) and the ISBSG dataset (in the original study [31]). These *Bellwether methods* can be used sample cross company projects from known repositories that shares similar characteristics with the new project to be estimated. It is evident from results in this dissertation that there exists exemplary projects as selected by the *SSPM*, *SysSam* and *RandSam* yielding improved prediction accuracy for the hold-out projects.

- From our empirical investigation of the existence of *Bellwethers* in chronological datasets, we found that such exemplary projects exist in at least 71 project cases (*growing portfolio* size of the preprocessed Desharnais dataset).
- Results from this study have shown that the window age of the BMW should not be older than 3 years. Even though we still have limited empirical evidence based on sampling from different chronological datasets, we recommend that in the use of the studied datasets, the age of the BMW should not exceed 3 years. Note that the 3 year threshold is a plausible window age for sampling the BMW based on our empirical evidence. However, this threshold is subject to change provided we have enough chronological datasets to make a sound generalisation.
- With regard to window size of the BMW, we found an approximate range of 40 to 80 exemplary projects. This range has proven to form a relatively reasonable window size for predicting the target of *new* or hold-out project data based on our empirical evidence. Note that from the law of large numbers and the central limit theorem, as the sample size increases ($n > 30$), the sample variance, s^2 turns to

decrease resulting in the sample mean, $\bar{x}$ being used as an unbiased estimator of the population mean, μ [67].

Chapter 5

Duplex Output Prediction Modelling

5.1. Software Effort Classification Scheme

The process of categorising estimated *software effort* into its respective class is referred to as *software effort classification*. The classification scheme aims to provide a self-guided benchmark to help determine the level of effort expended during software development. The scheme was first introduced in a previous study by Mensah *et al.* [3] using 14 historical datasets (*growing portfolio*) as training set for software effort estimation (SEE) modelling.

According to Mair *et al.* [156] and Qi *et al.* [9], it is difficult to obtain relevant datasets for studying software effort estimation, as the appropriate ones might be proprietary. Biases in the project data selection and model training criteria significantly contributes to unreliable prediction results [157]. The sampling procedure used [158] and the initial data preprocessing methods that lead to removal of outliers affect prediction results [48]. Inappropriate tuning of model parameters [159] and the lack of robust benchmark models [5] also affect the reliability of prediction accuracy in software effort estimation. Previous studies have proposed a simple baseline estimation model (automatically transformed linear model (ATLM)) based on a regression approach [5],

and a complex estimation model (confidence guided effort estimator (CoGEE)) based on a bi-objective optimisation approach [7]. However, there are still discrepancies between these models, and it is difficult for software engineers to adapt them, due to the lack of guidance on the interpretation of the estimated results [158]. The state-of-the-art baseline model (ATLM) by Whigham *et al.* [5] only considers the estimation of effort and does not offer a corresponding interpretation guide for the estimated results. Irrespective of the SEE model, software researchers normally fail to consider the difficulties faced by software practitioners in comparing the results from different prediction models. This limits the practical usefulness of such estimation models [158][157]. Thus, current prediction models are trained from SEE datasets that contain independent variables (such as size, development duration, *etc*) and a dependent variable (software effort), but do not consider the effort classes (see Fig. 8). Our contribution is a classification scheme that provides a self-guided interpretation of estimated effort (Fig. 9).

Proj_ID	KLOC	Duration	...	Effort	Class
1	449.9	13	...	336.3	?
2	289	14	...	116	?
3	254.2	13	...	258.7	?
4	214.4	18	...	86.9	?
5	40.5	7	...	82.5	?
....	...	...	...	...	...
n	200	11	...	130.3	?

Proj_ID	KLOC	Duration	...	Effort	Class
NewProj	60.2	14	...	???	???

Figure 8 Effort Estimation of projects without a classification scheme

In this dissertation, recently completed and exemplary projects (*Bellwethers*) are considered for the SEE modelling. In order to achieve the levels of classification – *low*, *moderate* and *high*, we first discretise the actual efforts recorded in the historical chronological datasets into three classes based on the optimal spacing theory (introduced by Eubank [51]). This theory was utilised because it gives rise to an optimal estimation of location thresholds for discretising a continuous variable into respective classes.

Proj_ID	KLOC	Duration	...	Effort	Class
1	449.9	13	...	336.3	H
2	289	14	...	116	M
3	254.2	13	...	258.7	H
4	214.4	18	...	86.9	M
5	40.5	7	...	82.5	L
....	...	...	...	...	...
n	200	11	...	130.3	M

Proj_ID	KLOC	Duration	...	Effort	Class
NewProj	60.2	14	...	???	???

Figure 9 Effort Estimation of projects with a classification scheme (H–high effort, M–moderate effort, L–low effort)

Eubank [51] introduced an asymptotically optimal spacing theory based on the density-quantile function over the interval $[0, 1]$ for estimating the location (μ) and scale (σ) parameters of a given continuous variable (Y). Let $Y_{(1)}, Y_{(2)}, \dots, Y_{(n)}$ denote independent and identically distributed targets of a sample of projects in a given dataset. Then, the cumulative distribution function (cdf), $F(y)$ is defined as

$$F(y) = F_o\left(\frac{y - \mu}{\sigma}\right) \quad (21)$$

where F_o denotes the distributional form [160], μ and σ denote the location and scale parameters estimated from the sample data. F_o is assumed to be a continuous distribution function of the target variable (y) with associated probability density function (pdf), f_o . The respective pdf, f_o is obtained by differentiating F (i.e., $f(y) = F'(y)$) defined in (22).

$$f(y) = \frac{1}{\sigma} f_o \left(\frac{y - \mu}{\sigma} \right) \quad (22)$$

In order to obtain the density-quantile function (q_o), we define a quantile function (Q) in (23) following a similar procedure by Eubank [51] where Q_o and f_o are the quantile function and the probability density function of F_o respectively.

$$Q(u) = F^{-1}(u) = \mu + \sigma Q_o(u), \quad 0 \leq u \leq 1 \quad (23)$$

Thus, Eubank's density-quantile function is defined in (24). In order to estimate the optimal location and spacing parameters, Parzen [161] has shown that for a large sample set, this problem can be seen as a continuous time series regression as defined in (25).

$$q_o(u) = f_o(Q_o(u)), \quad 0 \leq u \leq 1 \quad (24)$$

$$f_o Q_o(u) \tilde{Q}(u) = \mu f_o Q_o(u) + \sigma Q_o(u) f_o Q_o(u) + \sigma_B B(u) \quad (25)$$

where $\sigma_B = \sigma/\sqrt{n}$ and $B(\cdot)$ is the Brownian bridge process [160]. The ordered sample statistics of Y is defined as $Y_{(1)} \leq Y_{(2)} \leq \dots \leq Y_{(n)}$. Using the ordered sample statistics, we define the sample quantile function, $\tilde{Q}(u)$ together with the interval as given by Eubank [51] in (26).

$$\tilde{Q}(u) = Y_{(j)}, \quad \frac{j-1}{n} < u \leq \frac{j}{n}, \quad j = 1, \dots, n \quad (26)$$

In order to estimate the μ and σ parameters, $Q(u)$ over the interval $0 \leq u \leq 1$ is assumed to be known from the dataset. Then, by computing the expectation of Y in (27) as defined by Parzen [161], μ and σ can be estimated as in (28).

$$E[g(Y)] = E[gQ(U)] = \int_0^1 gQ(u)du \quad (27)$$

$$\mu = \int_0^1 Q(u)du \quad \text{and} \quad \sigma = \left[\int_0^1 \{Q(u) - \mu\}^2 du \right]^{\frac{1}{2}} \quad (28)$$

In summary, given a sample set of random variables $Y_1, \dots, Y_n$ denoting the targets (software effort) of variant of projects from a given dataset, we seek to classify the Y_j s into three distinct classes using Eubank's optimal spacing theory based on the quantile-density function. We model the cumulative distribution function, $F(y) = P(Y \leq y)$ and the probability density function, $f(y) = F'(y)$ under the assumption that the Y_j s are independent and identically distributed with $F(y)$ being continuous. We follow the mathematical modelling approach by Eubank [51] to define the quantile function, $Q(u)$ over the interval $0 \leq u \leq 1$ and estimate the location (μ) and scale (σ) parameters for the optimal discretisation of the Y_j s. Further details of the *software effort classification* scheme is provided in a previous study by Mensah *et al.* [3].

5.2. Duplex Output Prediction Algorithm

The *duplex output SEE* model is constructed based on Algorithm 1. The input comprises of training and validation sets with predictor variables - *Features(X)*, the response variable - *Effort(Y₁)* with its respective level of classification variable - *Classification(Y₂)*. We first load each dataset with N number of instances in step 1 and apply preprocessing

techniques to resolve all forms of variations in the dataset (step 2) as elaborated in chapter

3.

Algorithm 1: Duplex Output Model (X_r, Y_R, Y_Q)

Input: Features (X), Effort (Y_I) and Classification (Y_2)

Outputs: Output (Y_R) from prediction model and Classification Output (Y_Q)

$N \leftarrow$ total number of dataset instances

$p \leftarrow$ total number of features

$r \leftarrow$ subset of p features

$\beta_r \leftarrow$ Regression Coefficients

$Q_l \leftarrow$ Threshold for Low Effort class

$Q_m \leftarrow$ Threshold for Moderate Effort class

$Q_h \leftarrow$ Threshold for High Effort class

Begin

1: Load project dataset of size N

2: Preprocess dataset

 // Feature selection

3: **for** $i, i=1, \dots, p$ **do**

4: select $X[r] \leftarrow X[1]$ to $X[i]$

5: **end for**

6: Apply normalisation technique

7: **for** $j, j=1, \dots, N$ **do**

8: Sample Bellwethers using SSPM

9: **end for**

10: **for** $k, k=1, \dots, r$ **do**

11: Compute $\beta_r \leftarrow \text{inv}(X'X)X'Y$

12: **end for**

13: Test for β_r using AIC and Welch's t -test

14: Select best β_r for model construction based on VIF

 // Construct Regression model

15: $Y_I \leftarrow \beta_0 + \sum \beta_i X_i$

16: Output(Y_R) // Estimated Effort

 // Effort classification

17: Effort classification based on discretization function

18: **If** Output(Y_R) $< Q_l$ **then**

19: Output(Y_R) as "Low Effort"

20: **else if** Output(Y_R) $\geq Q_h$ **then**

21: Output(Y_R) as "High Effort"

22: **else** Output(Y_R) as "Moderate Effort"

23: **end if**

24: Output(Y_Q) // Effort classification

End

In steps 3 to 5, a subset of r *prior* features are selected from a total of p features. Here, the *prior* feature subset with the minimum MAE is used in constructing the *duplex output model*. The selected features are normalized as done in previous study [3] (step 6). We apply the *Bellwether method* specifically the *SSPM* which proved superior over *SysSam* and *RandSam* (in chapter 3) to sample the *Bellwether moving window* for prediction modelling (steps 7–9). The regression coefficients, β_r were computed in steps 10–12 for each of the SEE model. In step 13, the β_r values are tested using Yuen’s test and Cliff’s δ effect size statistics as recommended by Kitchenham *et al.* [53] for unequal data samples assumed to have inconsistencies or unequal variances which are statistically different from each other. The number of predictor variables can affect the model performance, hence preprocessing techniques and multicollinearity diagnostic tests are incorporated in order to obtain the best feature subset for the SEE models implemented for this study. Akaike Information Criterion (AIC) and Variance Inflation Factor (VIF) are used to check for multi-collinearity among the X_r to obtain the best β_r (step 14). We consider multicollinearity as an issue if the VIF is more than a specified threshold of ten [162] in each AIC model. The multicollinearity diagnostic analysis is confirmed based on the use of t -test statistic [162][68]. The β_r values are used to construct the regression model in step 15 and the predicted effort values are denoted as $Output(Y_R)$ in step 16. The $Output(Y_R)$ of the regression model is then classified into its respective class (step 17). The estimated effort, $Output(Y_R)$ are categorized into their respective effort classifications, namely *low (L)*, *moderate (M)* and *high (H)* efforts (steps 18–23). This is done based on the Eubank’s density-quantile function [51] thresholds to classify the effort values into

low (Q_l) and high (Q_h) classes respectively (step 18-23). Thus, estimated effort values less than Q_l are classified as *Low Effort* and estimated effort values more than Q_h are classified as *High Effort*. Lastly, estimated effort values falling within the thresholds $[Q_l, Q_h]$ are classified as *Moderate Effort*. The algorithm ends with the estimated effort in their respected classes as the second auxiliary response variable $Output(Y_Q)$ in step 24.

5.3. Classification of Software Effort

Table 12 shows the software effort classification thresholds for each of the normalised and un-normalised datasets based on the use of the density-quantile function [51]. The low and high effort class thresholds were discretised at the 25% and 75% quartiles, and the moderate effort class range was between the 25% – 75% quartiles. Given the high rate of inconsistencies within SEE datasets, this study agrees with other researchers [159][24][48] that *to accurately predict software effort, the datasets must be subjected to a normalisation technique*. This is a necessary step for accurate classifications of the estimated effort.

Table 12 Software effort classification based on density-quantile function

Dataset	Normalised data		
	Low	Moderate	High
Kitchenham	0 - 0.1295	0.1296 - 0.2628	> 0.2628
Desharnais	0 - 0.2995	0.2996 - 0.8870	> 0.8871
Maxwell	0 - 0.2120	0.2121 - 0.6384	> 0.6384
ISBSG	0 - 0.1951	0.1952 - 0.5229	> 0.5229
Dataset	Un-normalised data		
	Low	Moderate	High
Kitchenham	0 - 846.2	846.3 - 2.9122	> 2.9122
Desharnais	0 - 2346.8	2346.9 - 6042.8	> 6042.8
Maxwell	0 - 2155.5	2155.6 - 9247.0	> 9247.0
ISBSG	0 - 762.5	762.6 - 3926.3	> 3926.3

To confirm the aforementioned hypothesis, the prediction and classification of the software effort across the three learners (*i.e.*, ATLM, ElasticNet and DNN) were done based on Algorithm 1 and their classification performances were evaluated using mean absolute error (MAE) and logarithmic standard deviation (LSD) [113] as recorded in Table 13. Note that the minimum MAE and LSD represent the best performance evaluation across the three learners in each case of the normalised or un-normalised dataset as recorded in the shaded cells in Table 13. It was observed that the DNN yielded superior prediction accuracy on average across all normalised datasets with the exception of the Kitchenham dataset whereby the ElasticNet resulted in the best accuracy. Similar result was found in the case of the un-normalised dataset whereby the DNN and the ElasticNet yielded superior prediction accuracy on the average. We believe that to be able to assess the classification prediction of the learners, the effort classes should be the same irrespective of the variations between effort estimates of prediction models.

Table 13 Performance evaluation of learners using MAE and LSD

Learner	Normalised data		Un-normalised data		Dataset
	MAE	LSD	MAE	LSD	
ATLM	0.2502	0.8733	2350.9	3107.0	Kitchenham
ElasticNet	0.2070	0.5531	1831.3	7310.6	
DNN	0.2177	0.9562	3096.7	4598.0	
ATLM	0.5575	0.9179	3595.8	5264.0	Desharnais
ElasticNet	0.9922	1.1470	4829.7	4187.8	
DNN	0.5148	0.6908	3136.0	4188.2	
ATLM	0.5101	0.8029	6069.1	2766.8	Maxwell
ElasticNet	0.5328	0.6772	7205.4	9640.9	
DNN	0.1830	0.2267	2541.7	1065.2	
ATLM	0.6771	1.5170	4801.0	9266.2	ISBSG
ElasticNet	0.8969	0.9929	3684.8	6128.0	
DNN	0.6635	0.8496	3798.6	6142.4	

Note: shaded cells denote the best prediction performance across the three learners for each dataset

Here, a *loss* means higher error values between estimated and actual effort with respect to a wrong classification of the effort by a learner. Table 14 presents the recorded *losses* of the estimated effort classifications for each of the four chronological datasets. As the *SSPM* (recommended *Bellwether method* as detailed in chapter 3) was used to sample the BMW, we recorded the average number of *losses* (denoted as ‘0’) for the projects’ estimated effort classes per a dataset. Thus, a *loss* was recorded when the estimated effort class was different from the actual effort class for each project and the average *losses* or *wins* recorded across all projects for each dataset. Note that a *win* (‘1’) denotes a correct classification of the predicted effort in its respective class obtained based on the density-quantile function. Results show that the DNN had no losses on average in estimating the effort classification. This was followed by the ElasticNet with one loss in the case of the normalised ISBSG dataset and 2 losses in cases of the un-normalised Desharnais and Maxwell datasets respectively. The state-of-the-art baseline regression model – ATLM [5] recorded 2 losses and 3 losses in cases of the normalised and un-normalised datasets respectively (Table 14) in estimating the effort classification. Thus, the DNN model can accurately predict the effort classes as compared to the baseline model (ATLM) and the ElasticNet regression used in this study.

Table 14 Classification performance of predicted effort wrt to number of losses

Dataset	Normalised data			Un-normalised data		
	ATLM	ElasticNet	DNN	ATLM	ElasticNet	DNN
Kitchenham	1	1	1	0	1	1
Desharnais	0	1	1	1	0	1
Maxwell	1	1	1	0	0	1
ISBSG	0	0	1	0	1	1

Note: ‘1’ denotes a correct classification of the predicted effort in its respective class and ‘0’ denotes a wrong classification

The respective averages of the predicted effort from each learner together with their respective effort classes benchmarked against the actual effort and class per each dataset are recorded in Table 15. Note that prediction was done using the BMW on the hold-out projects and their averages of predicted effort and classes computed and recorded in Table 15. We investigated the existence of statistical significant differences between the three learners (*i.e.*, ATLM, ElasticNet and DNN) using Yuen's test and Cliff's δ effect size in Table 16. This was done at the 5% asymptotic significance level with a δ value of more than 0.276 yielding significant and practical difference between the prediction results from the learners in each of the studied datasets.

With the exception of the Kitchenham dataset, it was found that there were statistical significant differences between results from the paired learners in the case of data normalisation. In the case of the un-normalised datasets, it was found that there were statistical significant differences between the paired learners under consideration. Note that these un-normalised datasets have high variations among them resulting in the predictive models not adequately learning most of the intrinsic patterns during the training phase. This resulted in the misclassification of the estimated effort for the hold-out project estimation. Thus, in order to obtain accurate classification of the estimated effort, the study recommends data preprocessing techniques such as trimming and normalisation during the training and validation phases of setting up the predictive models.

Table 15 Actual vs. Predicted effort and their respective effort classes

Dataset	Normalised data				Un-normalised data			
	Actual (class)	Predicted effort (class)			Actual (class)	Predicted effort (class)		
		ATLM	ElasticNet	DNN		ATLM	ElasticNet	DNN
Kitchenham	0.2918 (H)	0.3460 (H)	0.5048 (H)	0.3403 (H)	3113.1 (M)	5419.8 (H)	2214.4 (M)	3128.0 (M)
Desharnais	0.7183 (M)	0.9305 (H)	0.6216 (M)	0.8099 (M)	4833.9 (M)	4575.7 (M)	2203.6 (L)	4494.8 (M)
Maxwell	0.5912 (M)	0.4206 (M)	0.3818 (M)	0.5743 (M)	8229.8 (M)	1024.4 (L)	6113.3 (M)	7592.2 (M)
ISBSG	0.5252 (H)	0.4059 (M)	0.2923 (M)	0.6015 (H)	3703.3 (M)	4.4418 (H)	3327.4 (M)	3422.5 (M)

Table 16 Statistical significant differences of predicted effort across learners

Learners	Normalised data			Un-normalised data		
	Yuen's test		Cliff's δ	Yuen's test		Cliff's δ
	<i>t</i> -value	<i>p</i> -value	effect size	<i>t</i> -value	<i>p</i> -value	effect size
Kitchenham Dataset						
ATLM vs. ElasticNet	-6.1913	4.5656e-08*	0.2494	7.2803	3.4968e-10*	0.3063**
ATLM vs. DNN	0.5862	0.5589	0.0242	-1.3678	0.1740	0.2026
DNN vs. ElasticNet	6.3680	2.2315e-08*	0.2550	-9.1855	9.9289e-14*	0.7628**
Desharnais Dataset						
ATLM vs. ElasticNet	-6.9372	1.3616e-09*	0.5345**	13.1448	3.5165e-19*	0.5725**
ATLM vs. DNN	-7.2173	2.4216e-10*	0.3579**	-0.8192	0.4147	0.0431
DNN vs. ElasticNet	4.5409	2.6452e-05*	0.2628	-23.9502	4.3824e-32*	0.1412
Maxwell Dataset						
ATLM vs. ElasticNet	-2.8675	0.0051*	0.8255**	6.1593	6.2886e-08*	0.574**
ATLM vs. DNN	0.1719	0.8638	0.2076	0.7637	0.4481	0.0328
DNN vs. ElasticNet	3.0690	0.0029*	0.8362**	-41.5009	2.3172e-45*	0.3511**
ISBSG Dataset						
ATLM vs. ElasticNet	-37.8801	1.2266e-51*	0.4920**	15.3447	2.8993e-22*	0.5589**
ATLM vs. DNN	2.4159	0.0173*	0.1767	-4.7134	7.0889e-06*	0.7113**
DNN vs. ElasticNet	36.6313	9.6384e-49*	0.4818**	-27.2875	3.6436e-35*	0.6965**

Statistical Significance: $p < 0.05^*$ Practical Significance: $\delta \geq 0.276^{**}$

Chapter 6

Final Remarks

6.1. Summary and Conclusion

Selection of relevant training data from a pool of historical and chronological datasets to be considered as the *moving window* has been shown in previous studies [34][37][40] to improve software effort estimation (SEE) accuracy. A *moving window* refers to a set of recently completed projects to be used as the training set when developing SEE models [38]. However, the different sizing, aging and weighting function parameters of the moving window affect prediction accuracy. Again, the selection of irrelevant project data to be considered as the moving window also affects prediction accuracy. This study attempted to tackle the four aforementioned challenges faced when using a moving window by conducting empirical and theoretical investigations based on six proven postulations. Moving windows work by an assumption that *recently completed projects are more relevant to new project estimation as compared to using the entire collection of available projects* (regarded as the *growing portfolio*). This study investigates the moving window assumption based on the theory that *the prediction outcome of a future event depends on the outcomes of previous events*. To attempt to tackle the relevancy constraints of moving windows, the study investigated the existence of exemplary projects (referred to as *Bellwethers* [28]) that yields improved prediction accuracy over *new* projects to be

estimated [31][26]. Thus this study uses three *Bellwether methods* (*SSPM*, *SysSam* and *RandSam*) to sample *Bellwethers* from a history of chronological projects to be considered as the moving window. The ergodic Markov chain is used to determine the stationarity of the *Bellwethers* (which are known to be recently completed projects with defined window size and age parameters). The sampled *Bellwethers* which are exemplary and recently completed projects are referred to as the *Bellwether moving window*. It should be noted that projects forming the *Bellwether moving window* have their completion dates prior to the starting date of the *new* or hold-out project(s) to be estimated. The sampled *Bellwether moving window* was benchmarked against the *growing portfolio* (*i.e.*, the entire collection of historical projects) and used to predict the targets of p_i hold-out projects from each studied chronological dataset. The study also introduced two weighting functions (namely Biweight and Triweight functions) in addition to the existing four weighting functions [37] to further investigate their effects on prediction accuracy. The sizing and aging parameters of the *Bellwether moving window* were investigated using the Markov chains methodology [30]. Investigations on the four studied chronological datasets (the Desharnais dataset, Kitchenham dataset, Maxwell dataset and ISBSG dataset) reinforce the existence of exemplary projects (referred to as the *Bellwether effect*) to be used for the training needs of prediction models. Validation of the study's results was done using robust statistical tests, namely Brunner's ANOVA-like test, Yuen's test and Cliff's δ effect size (as recommended by Kitchenham *et al.* [53]), and the mean absolute error (as recommended by Shepperd and MacDonell [47]). Statistical inferences were made at the 5% asymptotic significance level. The study also contributed with a guideline (based on

the studied chronological datasets) to assist researchers and practitioners in sampling *Bellwether moving windows* to be used for the training needs of prediction models. Empirical results from this study have shown that the *Bellwether moving window* should have an approximate window size of 40 to 80 projects and should not be older than 3 years old relative to the *new* projects (or hold-out projects). It was observed that weighting the *Bellwether moving window* with the Triweight function was more advantageous with respect to relative effective prediction accuracy. The characteristic of the exemplary projects that constituted the *Bellwether moving window* was further investigated and it was found that *trimming + log transforming* the data yielded improved prediction accuracy as compared to using the untransformed data.

Furthermore, this study seeks to address the *conclusion instability* problem [8] using the *Bellwether moving window* and a classification benchmark for the predicted software effort across different learners. *Conclusion instability* refers to variations in predicted results obtained when applying different SEE models on a given dataset [43][8]. Thus the study was motivated by the *conclusion instability* issue identified by Keung *et al.* [8] whereby the inconsistency in predicted results may be reduced based on the relevant selection of training and validation sets, data preprocessing and multiple prediction models. The study investigated whether the *Bellwether moving window* sampled from the studied chronological datasets can yield accurate classifications of the estimated effort values given a set of learners. The study adopted Eubank's optimal spacing theory [51] to classify the *software effort* into three classes – *low*, *moderate* and *high*. These classification thresholds will provide a benchmark to assist researchers and

practitioners to assess their prediction results and interpret the likely level of effort to be expended during project development [3]. The study therefore constructed *duplex output* SEE models using the sampled *Bellwether moving window* weighted with the Rectangular, Triangular, Epanechnikov, Gaussian, Biweight and Triweight functions. The *duplex output* SEE models predict the software effort (as *output 1*) and classifies the estimated effort into the respective effort class (as *output 2*). Three learners, namely automatically transformed linear model (ATLM), ElasticNet and Deep neural networks (DNN) are used and prediction performance assessed using the aforementioned robust statistical measures.

The software effort class estimated as auxiliary output was found to relatively reduce the *conclusion instability* of SEE results by adding an interpretation guide to know the prediction performance across the three learners. Thus given that the learners, l_i and/or l_j yield accurate classifications of their respective estimated efforts, then their prediction accuracies can relatively be trusted when used for *new* project estimation. Results show that the DNN learner accurately classifies the estimated *effort* of the hold-out projects into its respective *class* as compared to the ATLM and ElasticNet regression. The study therefore recommends the use of DNN trained with the *Bellwether moving window* for setting up SEE models since it yielded improved prediction accuracy across the studied datasets. Note that the *Bellwether moving window* will continue to be relevant for prediction purposes provided the phenomena being measured share similar characteristics with the *Bellwether* projects.

6.2. Threats to Validity

6.2.1. External Validity

This dissertation considered historical projects from four chronological datasets. These datasets are convenience samples but cannot be a general representative of all chronological datasets. Therefore, results from this study might not be confidently generalised beyond these datasets. However, the datasets used in this study are relatively large and consist of projects from well-known data repositories (*i.e.*, ISBSG and PROMISE [49]). Hence, they can be considered as a fair and generalisable representation of projects for an organisation that develops projects bearing similar characteristics.

As a result of using chronological projects for this study, it is important to note that numerous projects without start and completion dates were eliminated during the preprocessing phase. Thus, the heterogeneous nature of these datasets especially the ISBSG dataset resulted in a vast reduction of sampled projects. Hence, this cannot lead to a true reflection of results provided all projects had known start and completion dates.

6.2.2. Internal Validity

Threats to internal validity concern factors with regard to the selection of the independent variables used for setting up the predictive models that could influence the results of this study [124][163][14]. Thus, this study selected *prior* features (or independent variables which are known to be available before the commencement of a project [101]) from each of the four chronological datasets. This study aimed at obtaining *Bellwethers* to be used for prediction of the targets of *new* software projects, hence *posterior* features such as

lines of code, number of source files, etc were not considered as independent variables in building the SEE models. Even though most previous studies considered these *posterior* features in setting up the SEE model, it should be noted that in the practical sense, these *posterior* features are unknown before the development of *new* projects.

6.2.3. Construct Validity

This study considered the deep neural networks (DNN) [101][52] for building the prediction model when sampling *Bellwether moving windows* for *new* project estimation. We chose this learner because it has been shown to improve prediction accuracy in previous studies [31][33], and hence can be used as a reliable and effective SEE model. Even though results in chapter 4 are limited to a single learner for the training and validation needs, the DNN model might yield similar prediction results to other SEE models used in previous studies [34][36][37]. In attempting to address the *conclusion instability* problem [3] across learners in chapter 5, the study made use of three learners – Automatically transformed linear model (ATLM), ElasticNet regression and DNN. These three learners were chosen because they have shown to improve relative prediction accuracy in previous studies [5][98][3][32][101][33].

In this study, we considered the mean absolute error (MAE) as the performance evaluation measure for assessing the prediction results. Even though other evaluation measures could have been used for evaluation assessment, the MAE has been proven reliable and effective in previous studies [47][113][5][3]. Note that the logarithmic standard deviation (LSD) was used in addition to MAE when investigating the *conclusion instability* problem across learners in chapter 5. MAE and LSD are unbiased performance

measures towards model overestimation and underestimation [113][36][47]. Other evaluation measures such as the mean magnitude of relative error (MMRE), median magnitude of relative error (MdMRE) and PRED(k) are misleading evaluation measures which result in biasness in model assessment [113][106].

6.2.4. Conclusion Validity

The models employed in this study were automated and hence can result in automation biasness. For example, the sampling of *Bellwethers* by *SSPM*, *SysSam* and *RandSam* was done automatically. Automating a process involves series of assumptions made which can result in biasness. Nevertheless, the assumptions made in this study are based on prior knowledge from previous studies [37][34][30][36][58][164] in building data mining and prediction models. Hence, the validity of results can be trusted.

In summary, robust statistical tests (*e.g.* Yuen's test and Brunner's test) suitable for the datasets and problem at hand were carefully applied to verify the conclusion validity of results. Lastly, a robust effect size method, namely Cliff's δ effect size was used to provide additional evidence on the conclusion and relevance of results.

6.3. Future Work

In a future study, further empirical studies will be conducted in the context of other software engineering domains (such as software defect prediction, bug localization, software vulnerabilities) to investigate if the *Bellwether moving window* will yield improved prediction accuracy against the *growing portfolio* benchmark. A further study on this research is to compare results with other models such as the Bayesian estimation

model and other deep learning models (*e.g.* Convolutional neural networks (CNN), Recurrent neural networks (RNN), Deep belief networks (DBN) and Restricted Boltzmann Machines (RBM)). A future work can be done to extend the *Bellwether* concept to other chronological datasets such as the projects from the Finnish dataset, and project datasets from the industries. This will add more weight in generalising the relative effectiveness of using *Bellwethers* for improved prediction accuracy in software engineering and other related fields.

The following measures can be taken into consideration in order to improve on the proposed Bellwether framework:

- Prior to using the proposed framework, there is the need to preprocess the *growing portfolio* in order to sample the *Bellwether moving window* with the best size and age parameters. As stated in section 4.4, there is the need to apply *trimming + log transformation* in order to eliminate the tails of a given dataset.
- More chronological datasets are required for a better generalization of the proposed framework. This will result in a promising and more trustworthy window size and age thresholds for practitioners and researchers seeking to consider *Bellwethers* for predictive modelling.
- It was found that the Deep neural networks yielded improved prediction accuracy against the benchmark predictive models, namely automatically transformed linear model (ATLM) and ElasticNet regression. In going forward, there is the need to benchmark the Deep neural networks against other predictive models for

sound generalization and recommendation in choosing an effective predictive model.

- The chronological datasets considered for this study are very old. Although the emphasis is not on how old the datasets are, there is the need to consider more recent datasets by collecting recent projects from industries to further investigate the prediction accuracy of the proposed framework. This will give insights on how the framework will yield when used on real project cases from the industries.
- The investigative study found that the datasets have to be normally distributed in order to obtain the best window size and age parameters for improved prediction accuracy. In going forward, further studies can be conducted on using the framework on different chronological datasets following different probability distributions. This will further investigate the plausible effects of other distributions on prediction accuracy with respect to new project estimation.

Bibliography

- [1] E. Mendes and N. Mosley, “Bayesian Network Models for Web Effort Prediction: A Comparative Study,” *IEEE Transactions on Software Engineering*, vol. 34, no. 6, pp. 723–737, 2008.
- [2] M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, and S. Biffl, “Optimal project feature weights in analogy-based cost estimation: Improvement and limitations,” *IEEE Transactions on Software Engineering*, vol. 32, no. 2, pp. 83–92, 2006.
- [3] S. Mensah, J. Keung, M. F. Bosu, and K. E. Bennin, “Duplex output software effort estimation model with self-guided interpretation,” *Information and Software Technology*, vol. 94, pp. 1–13, 2017.
- [4] L. L. Minku and X. Yao, “Which models of the past are relevant to the present? A software effort estimation approach to exploiting useful past models,” *Automated Software Engineering*, vol. 24, no. 3, pp. 499–542, 2017.
- [5] P. A. Whigham, C. A. Owen, and S. G. Macdonell, “A Baseline Model for Software Effort Estimation,” *ACM Transactions on Software Engineering and Methodology*, vol. 24, no. 3, pp. 1–11, 2015.
- [6] S. K. Sehra, Y. S. Brar, N. Kaur, and S. S. Sehra, “Research patterns and trends in software effort estimation,” *Information and Software Technology*, vol. 91, pp. 1–21, 2017.
- [7] F. Sarro, A. Petrozziello, and M. Harman, “Multi-objective software effort estimation,” in *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*, 2016, pp. 619–630.
- [8] J. Keung, E. Kocaguneli, and T. Menzies, “Finding conclusion stability for selecting the best effort predictor in software effort estimation,” *Automated Software Engineering*, vol. 20, no. 4, pp. 543–567, 2013.
- [9] F. Qi, X.-Y. Jing, X. Zhu, X. Xie, B. Xu, and S. Ying, “Software effort estimation based on open source projects: Case study of Github,” *Information and Software Technology*, vol. 92, pp. 145–157, 2017.
- [10] T. Lee, T. Gu, and J. Baik, “MND-SCEMP: An empirical study of a software cost estimation modeling process in the defense domain,” *Empirical Software Engineering*, vol. 19, no. 1, pp. 213–240, 2014.
- [11] S. Mensah, J. Keung, J. Svajlenko, K. E. Bennin, and Q. Mi, “On the Value of a Prioritization Scheme for Resolving Self-admitted Technical Debt,” *Journal of Systems and Software*, vol. 135, pp. 37–54, 2017.
- [12] S. Mensah, J. Keung, M. F. Bosu, and K. E. Bennin, “Rework Effort Estimation of Self-admitted Technical Debt,” in *Joint Proceedings of the 4th International Workshop on*

Quantitative Approaches to Software Quality (QuASoQ) and 1st International Workshop on Technical Debt Analytics (TDA) co-located with the 23rd Asia-Pacific Software Engineering Conference (APSEC), 2016, pp. 72–75.

- [13] A. Potdar and E. Shihab, “An exploratory study on self-admitted technical debt,” *Proceedings - 30th International Conference on Software Maintenance and Evolution, ICSME 2014*, pp. 91–100, 2014.
- [14] G. Bavota and B. Russo, “A large-scale empirical study on self-admitted technical debt,” in *Proceedings of the 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories - MSR '16*, 2016, pp. 315–326.
- [15] F. Zhao, Y. Tang, Y. Yang, H. Lu, Y. Zhou, and B. Xu, “Is Learning-to-Rank Cost-Effective in Recommending Relevant Files for Bug Localization?,” in *Proceedings of the 2015 IEEE International Conference on Software Quality, Reliability and Security, QRS 2015*, 2015, pp. 298–303.
- [16] X. Ye, R. Bunescu, and C. Liu, “Learning to Rank Relevant Files for Bug Reports Using Domain Knowledge,” *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, no. April, pp. 689–699, 2014.
- [17] F. Zhang, A. E. Hassan, S. McIntosh, and Y. Zou, “The use of summation to aggregate software metrics hinders the performance of defect prediction models,” in *IEEE Transactions on Software Engineering*, 2017, vol. 43, no. 5, pp. 476–491.
- [18] F. Zhang, Q. Zheng, Y. Zou, and A. E. Hassan, “Cross-project defect prediction using a connectivity-based unsupervised classifier,” in *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*, 2016, pp. 309–320.
- [19] K. E. Bennin, K. Toda, Y. Kamei, J. Keung, A. Monden, and N. Ubayashi, “Empirical Evaluation of Cross-Release Effort-Aware Defect Prediction Models,” in *Proceedings of the 2016 IEEE International Conference on Software Quality, Reliability and Security, QRS 2016*, 2016, pp. 214–221.
- [20] Q. Huang, X. Xia, and D. Lo, “Supervised vs Unsupervised Models: A Holistic Look at Effort-Aware Just-in-Time Defect Prediction,” in *Proceedings of the 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2017, pp. 159–170.
- [21] X. Chen, Y. Zhao, Q. Wang, and Z. Yuan, “MULTI: Multi-objective effort-aware just-in-time software defect prediction,” *Information and Software Technology*, vol. 93, pp. 1–13, 2018.
- [22] T. Menzies, Y. Yang, G. Mathew, B. Boehm, and J. Hihn, “Negative results for software effort estimation,” *Empirical Software Engineering*, vol. 22, no. 5, pp. 2658–2683, 2016.
- [23] L. L. Minku and X. Yao, “Software effort estimation as a multiobjective learning problem,” *ACM Transactions on Software Engineering and Methodology*, vol. 22, no. 4, pp. 1–32, 2013.
- [24] E. Kocaguneli and T. Menzies, “How to Find Relevant Data for Effort Estimation?,” in *2011 International Symposium on Empirical Software Engineering and Measurement (ESEM)*, IEEE, 2011, pp. 255–264.

- [25] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, “An Empirical Comparison of Model Validation Techniques for Defect Prediction Models,” *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 1–18, 2017.
- [26] R. Krishna, T. Menzies, and W. Fu, “Too much automation? the bellwether effect and its implications for transfer learning,” in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, ACM, 2016, pp. 122–131.
- [27] S. J. Pan and Q. Yang, “a Survey on Transfer Learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.
- [28] B.-C. Chen, R. Ramakrishnan, J. W. Shavlik, and P. Tamma, “Bellwether analysis: Searching for cost-effective query-defined predictors in large databases,” *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 1, pp. 1–49, 2009.
- [29] B.-C. Chen, R. Ramakrishnan, J. W. Shavlik, and P. Tamma, “Bellwether analysis: predicting global aggregates from local regions,” in *Proceedings of the 32nd International Conference on Very Large Data Bases*, 2006, pp. 655–666.
- [30] R. P. Dobrow, “Introduction to Stochastic Processes With R,” *Wiley Online Library*, pp. 181–222, 2016.
- [31] S. Mensah, J. Keung, S. G. Macdonell, M. F. Bosu, and K. E. Bennin, “Investigating the Significance of Bellwether Effect to improve Software Effort Estimation,” in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 2017, pp. 340–351.
- [32] S. Mensah, J. Keung, M. Bosu, K. Bennin, and P. K. Kudjo, “A Stratification and Sampling Model for Bellwether Moving Window,” in *Proceedings of the 29th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 2017, pp. 481–486.
- [33] S. Mensah, J. Keung, S. G. MacDonell, M. F. Bosu, and K. E. Bennin, “Investigating the significance of the Bellwether effect to improve software effort prediction: Further empirical study,” *IEEE Transactions on Reliability*, vol. In press, 2018.
- [34] S. Amasaki and C. Lokan, “A replication study on the effects of weighted moving windows for software effort estimation,” in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, EASE*, 2016.
- [35] C. Lokan and E. Mendes, “Investigating the use of duration-based moving windows to improve software effort prediction: a replicated study,” *Information and Software Technology*, vol. 56, pp. 1063–1075, 2014.
- [36] C. Lokan and E. Mendes, “Investigating the use of moving windows to improve software effort prediction: a replicated study,” *Empirical Software Engineering*, vol. 22, no. 2, pp. 716–767, 2016.
- [37] S. Amasaki and C. Lokan, “On the effectiveness of weighted moving windows: Experiment on linear regression based software effort estimation,” *Journal of Software: Evolution and Process*, vol. 27, no. 7, pp. 488–507, 2015.
- [38] 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 (ESEM). IEEE Computer Society, 2009, pp. 111–122.

- [39] C. Lokan and E. Mendes, “Investigating the Use of Duration-Based Moving Windows to Improve Software Effort Prediction,” in *Proceedings of the 2012 19th Asia-Pacific Software Engineering Conference*, 2012, pp. 818–827.
- [40] S. Amasaki and C. Lokan, “Towards Better Selection Between Moving Windows and Growing Portfolio,” in *Proceedings of the 17th International Conference Product-Focused Software Process Improvement: PROFES*, 2016, vol. 17, pp. 627–630.
- [41] S. Amasaki and C. Lokan, “The evaluation of weighted moving windows for software effort estimation,” in *Proceedings of the International Conference on Product Focused Software Process Improvement*, 2013, vol. 7983 LNCS, pp. 214–228.
- [42] B. Kitchenham, S. L. Pfleeger, B. McColl, and S. Eagan, “An empirical study of maintenance and development estimation accuracy,” *Journal of Systems and Software*, vol. 64, no. 1, pp. 57–77, 2002.
- [43] E. Kocaguneli and T. Menzies, “Software effort models should be assessed via leave-one-out validation,” *Journal of Systems and Software*, vol. 86, no. 7, pp. 1879–1890, 2013.
- [44] P. Phannachitta, J. Keung, A. Monden, and K. Matsumoto, “A stability assessment of solution adaptation techniques for analogy-based software effort estimation,” *Empirical Software Engineering*, vol. 22, no. 1, pp. 474–504, 2016.
- [45] S. Dragicevic, S. Celar, and M. Turic, “Bayesian network model for task effort estimation in agile software development,” *Journal of Systems and Software*, vol. 127, pp. 109–119, 2017.
- [46] T. Menzies, O. Jalali, J. Hihn, D. Baker, and K. Lum, “Stable rankings for different effort models,” *Automated Software Engineering*, vol. 17, no. 4, pp. 409–437, 2010.
- [47] M. Shepperd and S. MacDonell, “Evaluating prediction systems in software project estimation,” *Information and Software Technology*, vol. 54, no. 8, pp. 820–827, 2012.
- [48] M. F. Bosu and S. G. Macdonell, “Data Quality in Empirical Software Engineering: A Targeted Review,” in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering - EASE '13*, 2013, pp. 171–176.
- [49] T. Menzies, R. Krishna, and D. Pryor, “The PROMISE Repository of Empirical Software Engineering Data,” *North Carolina State University, Department of Computer Science*, 2016. [Online]. Available: <http://openscience.us/repo>.
- [50] A. Bakir, B. Turhan, and A. Bener, “A comparative study for estimating software development effort intervals,” *Software Quality Journal*, vol. 19, no. 3, pp. 537–552, 2011.
- [51] R. L. Eubank, “A density-quantile function approach to optimal spacing selection,” *The Annals of Statistics*, vol. 9, no. 3, pp. 494–500, 1981.
- [52] Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [53] B. Kitchenham *et al.*, “Robust Statistical Methods for Empirical Software Engineering,”

Empirical Software Engineering, vol. 21, no. 1, pp. 212–259, 2016.

- [54] E. Kocaguneli, T. Menzies, and E. Mendes, “Transfer learning in effort estimation,” *Empirical Software Engineering*, vol. 20, no. 3, pp. 813–843, 2015.
- [55] A. Z. Zambom and R. Dias, “A Review of Kernel Density Estimation with Applications to Econometrics,” *International Econometric Review*, vol. 5, no. 1, pp. 20–42, 2012.
- [56] P. N. Tattar, S. Ramaiah, and B. G. Manjunath, *A Course in Statistics with R*. John Wiley & Sons, 2016.
- [57] J. E. Cook and A. L. Wolf, “Discovering models of software processes from event-based data,” *ACM Transactions on Software Engineering and Methodology*, vol. 7, no. 3, pp. 215–249, 1998.
- [58] B. Zhou, H. Okamura, and T. Dohi, “Enhancing performance of random testing through Markov chain Monte Carlo methods,” *IEEE Transactions on Computers*, vol. 62, no. 1, pp. 186–192, 2013.
- [59] S. Bandyopadhyay, “Simulated Annealing Using a Reversible Jump Markov Chain Monte Carlo,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 479–490, 2005.
- [60] G. Gianini, A. Rizzi, and E. Damiani, “A Retinex model based on Absorbing Markov Chains,” *Information Sciences*, vol. 327, pp. 149–174, 2016.
- [61] R. Patel, P. Agharkar, and F. Bullo, “Robotic Surveillance and Markov Chains with Minimal Weighted Kemeny Constant,” *IEEE Transactions on Automatic Control*, vol. 60, no. 12, pp. 3156–3167, 2015.
- [62] E. N. N. Nortey, T. Ansah-narh, R. Asah-asante, and R. Minkah, “A Markov chain Monte Carlo (MCMC) methodology with bootstrap percentile estimates for predicting presidential election results in Ghana,” *SpringerPlus*, no. April, 2015.
- [63] S. Walk, P. Singer, M. Strohmaier, T. Tudorache, M. A. Musen, and N. F. Noy, “Discovering Beaten Paths in Collaborative Ontology-Engineering Projects using Markov Chains,” *Journal of Biomedical Informatics*, vol. 51, pp. 254–271, 2014.
- [64] S. Wu *et al.*, “Survey on Prediction Algorithms in Smart Homes,” *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 636–644, 2017.
- [65] X. R. Cao, “Optimization of Average Rewards of Time Nonhomogeneous Markov Chains,” *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 1841–1856, 2015.
- [66] H. Fischer, *A history of the central limit theorem: From classical to modern probability theory*. Springer Science & Business Media, 2010.
- [67] K. Yao and J. Gao, “Law of Large Numbers for Uncertain Random Variables,” *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 3, pp. 615–621, 2016.
- [68] J. M. Lachin, “Introduction to sample size determination and power analysis for clinical trials,” *Controlled Clinical Trials*, vol. 2, no. 2, pp. 93–113, 1981.
- [69] E. Brunner, U. Munzela, and M. L. Purib, “The multivariate nonparametric Behrens–

- Fisher problem Edgar problem,” *Statistical Planning and Inference*, vol. 108, pp. 37–53, 2002.
- [70] T. Tian and R. R. Wilcox, “A Comparison of Two Rank Tests for Repeated Measures Designs,” *Journal of Modern Applied Statistical Methods*, vol. 6, no. 1, pp. 331–335, 2007.
 - [71] P. Mair and R. Wilcox, “Robust Statistical Methods in R Using the WRS2 Package,” *Technical report, Harvard University*, 2016.
 - [72] K. K. Yuen, “The Two-Sample Trimmed t for Unequal Population Variances,” *Biometrika*, vol. 61, no. 1, pp. 165–170, 1974.
 - [73] B. L. Welch, “The Significance of the Difference Between Two Means when the Population Variances are Unequal,” *Biometrika*, vol. 29, no. 3, pp. 350–362, 1938.
 - [74] N. Cliff, “Dominance statistics: Ordinal analyses to answer ordinal questions,” *Psychological Bulletin*, vol. 114, no. 3, pp. 494–509, 1993.
 - [75] H. C. Kraemer and D. J. Kupfer, “Size of treatment effects and their importance to clinical research and practice,” *Biological psychiatry*, vol. 59, no. 11, pp. 990–996, 2006.
 - [76] A. Arcuri and L. Briand, “A Hitchhiker’s guide to statistical tests for assessing randomized algorithms in software engineering,” *Software Testing Verification and Reliability*, vol. 24, no. 3, pp. 219–250, 2014.
 - [77] S. Vegas, C. Apa, and N. Juristo, “Crossover Designs in Software Engineering Experiments: Benefits and Perils,” *IEEE Transactions on Software Engineering*, vol. 42, no. 2, pp. 120–135, 2016.
 - [78] L. Madeyski and B. Kitchenham, “Effect sizes and their variance for AB/BA crossover design studies,” *Empirical Software Engineering*, pp. 1–36, 2017.
 - [79] V. B. Kampenes, T. Dyba, J. E. Hannay, and D. I. K. Sjøberg, “A systematic review of effect size in software engineering experiments,” *Information and Software Technology*, vol. 49, no. 11–12, pp. 1073–1086, 2007.
 - [80] J. W. Tukey, *Exploratory data Analysis*, 1st ed., vol. 1. Pearson, 1977.
 - [81] E. Kocaguneli, T. Menzies, and J. W. Keung, “Kernel methods for software effort estimation: Effects of different kernel functions and bandwidths on estimation accuracy,” *Empirical Software Engineering*, vol. 18, no. 1, pp. 1–24, 2013.
 - [82] K. Beyer and R. Ramakrishnan, “Bottom-up computation of sparse and Iceberg CUBE,” in *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, 1999, vol. 28, no. 2, pp. 359–370.
 - [83] J. Han, J. Pei, G. Dong, and K. Wang, “Efficient computation of iceberg cubes with complex measures,” in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001, vol. 30, no. 2, pp. 1–12.
 - [84] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” in *Communications of the ACM*, 2008, vol. 51, no. 1, pp. 107–113.
 - [85] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, “On the relative value of cross-

- company and within-company data for defect prediction,” *Empirical Software Engineering*, vol. 14, no. 5, pp. 540–578, 2009.
- [86] M. D’Ambros, M. Lanza, and R. Robbes, “Evaluating defect prediction approaches: a benchmark and an extensive comparison,” *Empirical Software Engineering*, vol. 17, no. 4–5, pp. 531–577, 2012.
 - [87] R. Wu, H. Zhang, S. Kim, and S. C. Cheung, “ReLink: Recovering links between bugs and changes,” in *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 2011, pp. 15–25.
 - [88] M. Jureczko and L. Madeyski, “Towards identifying software project clusters with regard to defect prediction,” in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering - PROMISE '10*, 2010, pp. 1–10.
 - [89] M. Shepperd, Q. Song, Z. Sun, and C. Mair, “Data Quality : Some Comments on the NASA Software Defect Datasets,” *IEEE Transactions on Software Engineering*, vol. 39, no. 9, pp. 1208–1215, 2013.
 - [90] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
 - [91] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, “Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings,” *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 485–496, 2008.
 - [92] B. Turhan, A. Tosun, and A. Bener, “Empirical evaluation of mixed-project defect prediction models,” in *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, 2011, pp. 396–403.
 - [93] S. A. B and C. Lokan, “A Virtual Study of Moving Windows for Software Effort Estimation Using Finnish Datasets,” in *International Conference on Product-Focused Software Process Improvement*, 2017, vol. 10611, pp. 71–79.
 - [94] B. W. Boehm *et al.*, *Software Cost Estimation with Cocomo II*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.
 - [95] J. W. Keung, B. A. Kitchenham, and D. R. Jeffery, “Analogy-X: Providing statistical inference to analogy-based software cost estimation,” *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 471–484, 2008.
 - [96] N.-H. Chiu and S.-J. Huang, “The adjusted analogy-based software effort estimation based on similarity distances,” *Journal of Systems and Software*, vol. 80, no. 4, pp. 628–640, 2007.
 - [97] C. Mair *et al.*, “An investigation of machine learning based prediction systems,” *The Journal of Systems and Software*, vol. 53, pp. 23–29, 2000.
 - [98] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens, “Data mining techniques for software effort estimation: A comparative study,” *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 375–397, 2012.
 - [99] A. Heiat, “Comparison of artificial neural network and regression models for estimating software development effort,” *Information and Software Technology*, vol. 44, no. 15, pp. 911–922, 2002.

- [100] V. Khatibi Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons," *Empirical Software Engineering*, vol. 19, no. 4, pp. 857–884, 2014.
- [101] S. Mensah, J. Keung, K. E. Bennin, and M. F. Bosu, "Multi-Objective Optimization for Software Testing Effort Estimation," in *Proceedings of the 28th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 2016, pp. 527–530.
- [102] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, and R. Madachy, "Active Learning and effort estimation: Finding the essential content of software effort estimation data," *IEEE Transactions on Software Engineering*, vol. 39, no. 8, pp. 1040–1053, 2013.
- [103] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd Editio. Erlbaum Associates, Hillsdale, 1988.
- [104] F. Ferrucci, M. Harman, J. Ren, and F. Sarro, "Not going to take this anymore: Multi-objective overtime planning for Software Engineering projects," *Proceedings - International Conference on Software Engineering*, pp. 462–471, 2013.
- [105] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. Wadsworth, Belmont, CA: CRC Press, 1984.
- [106] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Transactions on Software Engineering*, vol. 23, no. 11, pp. 736–743, 1997.
- [107] M. Azzeh, A. B. Nassif, and S. Banitaan, "A better case adaptation method for case-based effort estimation using multi-objective optimization," *Proceedings - 2014 13th International Conference on Machine Learning and Applications, ICMLA 2014*, no. 3, pp. 409–414, 2014.
- [108] J. Kennedy, "Particle swarm optimization," *Encyclopedia of machine learning*. Springer, Boston, MA, pp. 760–766, 2011.
- [109] J. Carlos, F. Cabrera, and C. A. C. Coello, "Micro-MOPSO : A Multi-Objective Particle Swarm Optimizer That Uses a Very Small Population Size," in *Multi-Objective Swarm Intelligent Systems*, Springer, Berlin, Heidelberg, 2010, pp. 83–104.
- [110] B. Turhan and E. Mendes, "A comparison of cross-versus single-company effort prediction models for web projects," in *2014 40th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE*, 2014, pp. 285–292.
- [111] A. N. Lam, A. T. Nguyen, H. A. Nguyen, and T. N. Nguyen, "Combining deep learning with information retrieval to localize buggy files for bug reports," in *Proceedings - 2015 30th IEEE/ACM International Conference on Automated Software Engineering, ASE*, 2015, pp. 476–481.
- [112] S. Kim, M. Lee, and J. Shen, "A novel deep learning by combining discriminative model with generative model," in *Proceedings of the International Joint Conference on Neural Networks*, 2015, pp. 3371–3377.
- [113] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion MMRE," *IEEE Transactions on Software Engineering*, vol. 29, no.

11, pp. 985–995, 2003.

- [114] R. R. Wilcox and H. K. Keselman, “Modern Robust Data Analysis Methods: Measures of Central Tendency,” *Psychological methods*, vol. 8, no. 3, pp. 254–274, 2003.
- [115] M. F. Bosu, S. Mensah, K. E. Bennin, and D. Abuaiadah, “Revisiting the Conclusion Instability Issue in Software Effort Estimation,” *Proceedings of the 30th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, vol. In press, 2018.
- [116] J. M. Desharnais, “Analyse statistique de la productivité des projets informatiques à partir de la technique des points de fonction,” University of Montreal, 1989.
- [117] Y. F. Li, M. Xie, and T. N. Goh, “A study of the non-linear adjustment for analogy based software cost estimation,” *Empirical Software Engineering*, vol. 14, no. 6, pp. 603–643, 2009.
- [118] 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*, vol. 36, no. 3, pp. 5921–5931, 2009.
- [119] M. Tsunoda, T. Kakimoto, A. Monden, and K. Matsumoto, “An Empirical Evaluation of Outlier Deletion Methods for Analogy-Based Cost Estimation,” in *Proceedings of the 7th International Conference on Predictive Models in Software Engineering. ACM*, 2011, p. 17.
- [120] F. González-Ladrón-de-Guevara, M. Fernández-Diego, and C. Lokan, “The usage of ISBSG data fields in software effort estimation: A systematic mapping study,” *Journal of Systems and Software*, vol. 113, pp. 188–215, 2016.
- [121] C. Lokan, T. Wright, P. R. Hill, and M. Stringer, “Organizational benchmarking using the ISBSG data repository,” *IEEE Software*, vol. 18, no. 5, pp. 26–32, 2001.
- [122] K. Maxwell, *Applied statistics for software managers*. Englewood Cliffs, NJ: Prentice Hall, 2002.
- [123] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris, “Software productivity and effort prediction with ordinal regression,” *Information and Software Technology*, vol. 47, pp. 17–29, 2005.
- [124] E. Mendes, I. Watson, and C. Triggs, “A Comparative Study of Cost Estimation Models for Web Hypermedia Applications,” *Empirical Software Engineering*, vol. 8, no. 2, pp. 163–196, 2003.
- [125] ISBSG, “The ISBSG Development & Enhancement Project Data in: ISBSG (Ed.),” 2013. [Online]. Available: <http://www.isbsg.org>.
- [126] J. Huang, Y.-F. Li, and M. Xie, “An empirical analysis of data preprocessing for machine learning-based software cost estimation,” *Information and Software Technology*, vol. 67, pp. 108–127, 2015.
- [127] L. L. Minku and X. Yao, “Ensembles and locality : Insight on improving software effort estimation,” *Information and Software Technology*, vol. 55, no. 8, pp. 1512–1528, 2013.

- [128] D. Rodríguez, M. A. Sicilia, E. García, and R. Harrison, “Empirical findings on team size and productivity in software development,” *The Journal of Systems & Software*, vol. 85, no. 3, pp. 562–570, 2012.
- [129] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, F. Sarro, and E. Mendes, “Using tabu search to configure support vector regression for effort estimation,” *Empirical Software Engineering*, vol. 18, no. 3, pp. 506–546, 2013.
- [130] Y. S. Seo and D. H. Bae, “On the value of outlier elimination on software effort estimation research,” *Empirical Software Engineering*, vol. 18, no. 4, pp. 659–698, 2013.
- [131] E. Mendes and B. Kitchenham, “Further comparison of cross-company and within-company effort estimation models for Web applications,” in *Proceedings of the 10th International Symposium on Software Metrics (METRICS’04)*, 2004, pp. 348–357.
- [132] R. D. Cook, “Detection of Influential Observation in Linear Regression,” *Technometrics*, vol. 19, no. 1, p. 15, 1977.
- [133] D. A. Belsley, E. Kuh, and R. E. Welsch, *Regression diagnostics: Identifying influential data and sources of collinearity*. John Wiley & Sons, 2005.
- [134] E. A. Montgomery, Douglas C. Peck and G. G. Vining, *Introduction to Linear Regression Analysis*, 5th ed. John Wiley & Sons, 2012.
- [135] R. R. Wilcox, *Applying Contemporary Statistical Techniques*, 1st ed. Academic Press, 2003.
- [136] F. Zhang, I. Keivanloo, and Y. Zou, “Data Transformation in Cross-project Defect Prediction,” *Empirical Software Engineering*, vol. 22, no. 6, pp. 3186–3218, 2017.
- [137] J. Nam and S. Kim, “Heterogeneous defect prediction,” in *Proceedings of the 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), Bergamo, Italy*, 2015, pp. 508–519.
- [138] B. Kitchenham, “Keynote: Robust Statistical Methods – Why, What and How,” in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE), ACM*, 2015, pp. 1–6.
- [139] R. Shanmugam and R. Chattamvelli, “Skewness and Kurtosis,” in *Statistics for Scientists and Engineers*, Wiley Online Library, 2016, pp. 89–110.
- [140] D. Pelleg and A. W. Moore, “X-means: Extending K-means with efficient estimation of the number of clusters,” in *Proceedings of the 17th International Conference on Machine Learning table of contents*, 2000, pp. 727–734.
- [141] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.
- [142] A. E. Hoerl and R. W. Kennard, “Ridge regression: Biased estimation for nonorthogonal problems,” *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.
- [143] R. Tibshirani, “Regression shrinkage and selection via the lasso: a retrospective,” *Journal*

of the Royal Statistical Society: Series B (Statistical Methodology), vol. 73, no. 3, pp. 273–282, 2011.

- [144] J. Friedman, T. Hastie, and R. Tibshirani, “Regularization paths for generalized linear models via coordinate descent,” *Journal of statistical software*, vol. 33, no. 1, pp. 1–22, 2010.
- [145] P. T. Trish, O. Edison, P. H. Andrew, P. Giovanni, and P. Carlo, “Prediction of kinase inhibitor response using activity profiling, in vitro screening, and elastic net regression,” *BMC systems biology*, vol. 8, no. 1, p. 74, 2014.
- [146] Q. Li, B. Xie, J. You, W. Bian, and D. Tao, “Correlated Logistic Model With Elastic Net Regularization for Multilabel Image Classification,” *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3801–3813, 2016.
- [147] J. Schmidhuber, “Deep Learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.
- [148] S. Wang, T. Liu, and L. Tan, “Automatically Learning Semantic Features for Defect Prediction,” in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 297–308.
- [149] L. Li *et al.*, “Hybrid Deep Neural Network--Hidden Markov Model (DNN-HMM) Based Speech Emotion Recognition,” in *Proceedings of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, 2013, pp. 312–317.
- [150] M. Abadi *et al.*, “Deep Learning with Differential Privacy,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM*, 2016, pp. 308–318.
- [151] B. Zamanlooy and M. Mirhassani, “Efficient VLSI implementation of neural networks with hyperbolic tangent activation function,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 1, pp. 39–48, 2014.
- [152] N. Zhang and D. Shetty, “An effective LS-SVM-based approach for surface roughness prediction in machined surfaces,” *Neurocomputing*, vol. 198, pp. 35–39, 2016.
- [153] D. W. Marquardt, “An Algorithm for Least-Squares Estimation of Nonlinear Parameters,” *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.
- [154] M. T. Hagan and M. B. Menhaj, “Training Feedforward Networks with the Marquardt Algorithm,” *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.
- [155] M. F. Bosu and S. G. Macdonell, “A taxonomy of data quality challenges in empirical software engineering,” in *Proceedings of the Australian Software Engineering Conference (ASWEC), IEEE*, 2013, pp. 97–106.
- [156] C. Mair, M. Shepperd, and M. Jørgensen, “An analysis of data sets used to train and validate cost prediction systems,” *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, p. 1, 2005.
- [157] B. Kitchenham and E. Mendes, “Why comparative effort prediction studies may be invalid,” in *Proceedings of the 5th International Conference on Predictor Models in*

Software Engineering (PROMISE), ACM, 2009, no. 5, pp. 1–5.

- [158] I. Myrtveit and E. Stensrud, “Validity and reliability of evaluation procedures in comparative studies of effort prediction models,” *Empirical Software Engineering*, vol. 17, no. 1–2, pp. 23–33, 2012.
- [159] L. Song, L. L. Minku, and X. Yao, “The impact of parameter tuning on software effort estimation using learning machines,” in *Proceedings of the 9th International Conference on Predictive Models in Software Engineering (PROMISE)*, ACM, 2013, pp. 1–10.
- [160] R. L. Eubank, P. L. Smith, and P. W. Smith, “Uniqueness and Eventual Uniqueness of Optimal Designs in Some Time Series Models,” *The Annals of Statistics*, vol. 9, no. 3, pp. 486–493, 1981.
- [161] E. Parzen, “Nonparametric Statistical Data Modeling,” *Journal of the American Statistical Association*, vol. 74, no. 365, pp. 105–121, 1979.
- [162] R. M. O’Brien, “A caution regarding rules of thumb for variance inflation factors,” *Quality and Quantity*, vol. 41, no. 5, pp. 673–690, 2007.
- [163] R. Malhotra and M. Khanna, “An empirical study for software change prediction using imbalanced data,” *Empirical Software Engineering*, vol. 22, no. 6, pp. 2806–2851, 2017.
- [164] Z. Tu and Z. Song-Chun, “Image segmentation by data-driven markov chain monte carlo,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 657–673, 2002.

Appendix

Appendix 1: *Feature/Variable Definition in Kitchenham Dataset*

Variable	Full name	Type	Description
project	Project identifier	Numerical	Assignment of numbers to projects 1 to 145
client	Client code	Nominal	Assignment of client codes {1, 2, 3, 4, 5}
type	Project type	Nominal	Categorization of projects {A, C, D, P, Pr, U}
start_date	Actual start date	Date	Range of actual start dates of projects {14/01/1994 to 13/11/1998}
duration	Actual duration	Numerical	Duration of project from specification until delivery, measured in days
effort	Actual effort	Numerical	Work carried out from specification until delivery, measured in hours
AFP	Actual function points	Numerical	Function points measured according to the IFPUG standard 4.0: inputs, outputs, inquiries, logical internal files and external interface files
end_date	Estimated completion date	Date	Range of estimated completion dates of projects {27/01/1995 to 17/02/1999}
first_estimate	First estimate	Numerical	First estimate of projects measured in hours
method	First estimate method	Numerical	Categorization of first estimation method {A, C, CAE, D, EO, W}

Appendix 2: *Feature/Variable Definition in Desharnais Dataset*

Variable	Full name	Type	Description
TeamExp	Team's experience	Numerical	Measured in years
ManagerExp	Project manager's experience	Numerical	Measured in years
YearEnd	Year of end	Numerical	Year of completion
Length	Length of project	Numerical	Measured in months
Effort	Development effort	Numerical	Measured in person-hours
Transactions	Transactions	Numerical	Number of transactions
Entities	Entities	Numerical	Number of entities
PointsAdjust	Adjusted function points	Numerical	Number of Adjusted function points
Envergure	Development environment	Numerical	Development environment
PointsNonAdjusted	Unadjusted function points	Numerical	Number of Unadjusted function points
Language	Programming language	Nominal	Traditional language {1}, high level language {2}, 4 th generation language {3}

Appendix 3: *Feature/Variable Definition in ISBSG Dataset*

Variable	Full name	Type	Description
DevType	Development type	Nominal	1 = Enhancement 2 = New development 3 = Re-development
OrgType	Organization type	Nominal	1 = Banking 2 = Communication 3 = Community services 4 = Computer, Software, ISP 5 = Electricity, Gas, Water; 6 = Financial, Property & Business Services; 7 = Insurance; 8 = Manufacturing; 9 = Government, Public Administration 10 = Transport & Storage; 11 = Wholesale & Retail Trade; 12 = Others.
BusType	Business Area	Nominal	1 = Accounting; 2 = Banking; 3 = Engineering; 4 = Financial; 5 = Insurance, Actuarial; 6 = Inventory; 7 = Legal; 8 = Logistics; 9 = Manufacturing 10 = Personnel; 11 = Research & Development; 12 = Sales & Marketing; 13 = Telecommunications; 14 = Others.
AppType	Application Type	Nominal	1 = Billing; 2 = Office information system, Executive information system, Decision support system; 3 = Electronic Data Interchange; 4 = Financial; 5 = Management Information System; 6 = Network Management, Communications; 7 = Process control, sensor control, real time; 8 = Transaction/Production System; 9 = Others.
DevPlat	Development Platform	Nominal	1 = Mainframe 2 = Mid-range 3 = Multi; 4 = Personal Computer.
PriProLan	Primary Programming Language	Nominal	1 = ABAP; 2 = Access;

			3 = ASP; 4 = C; 5 = C++; 6 = COBOL; 7 = JAVA; 8 = Lotus Notes; 9 = NATURAL; 10 = ORACLE; 11 = PL/I; 12 = PL/SQL; 13 = PowerBuilder; 14 = SQL; 15 = Visual Basic; 16 = Others.
DevTech	Development Techniques	Nominal	1 = Business area modeling; 2 = Data Modelling; 3 = Event Modelling 4 = Joint Application Development; 5 = Multifunction teams 6 = Object Oriented Analysis; 7 = Object Oriented Design; 8 = Process Modelling; 9 = Prototyping; 10 = Rapid Application Development; 11 = WaterFall; 12 = Others.
InpCont	Input count	Numerical	The count of inputs
OutCont	Output count	Numerical	The count of outputs
EnqCont	Enquiry count	Numerical	The count of enquiries
FileCont	File count	Numerical	The count of files
IntCont	Interface count	Numerical	The count of interfaces
AFP	Adjusted function points	Numerical	The adjusted function point-count number
NorEffort	Normalized work effort	Numerical	For project covering less than a full development life cycle, this value is an estimate of the full development effort in hours.

Appendix 4: *Feature/Variable Definition in Maxwell Dataset*

Variable	Full name	Type	Description
syear	Start year of project	Numerical	1985 to 1993
Time	Time	Numerical	Time = syear – 1985 + 1, with levels: 1,2,3,4,5,6,7,8,9
App	Application	Nominal	1 = Information/on-line service (infServ) 2 = Transaction control, logistics, order processing (TransPro) 3 = Customer service (CustServ) 4 = Production control, logistics, order processing (ProdCont) 5 = Management information system (MIS)
Har	Hardware platform	Nominal	1 = Personal computer (PC) 2 = Mainframe (Mainfrm) 3 = Multi-platform (Multi) 4 = Minicomputer (Mini) 5 = Networked (Network)
Db	Database	Nominal	1 = Relatnl (Relational) 2 = Sequentl (Sequential) 3 = Other (Other) 4 = None (None)
Ifc	User interface	Nominal	1 = Graphical user interface (GUI) 2 = Text user interface (TextUI)
Source	Where developed	Nominal	1 = In-house (Inhouse) 2 = Outsourced (Outsrccd)
Telonus	Telonus use	Nominal	0 = No 1 = Yes
Nlan	Number of different development languages used	Ordinal	1 = one language used 2 = two languages used 3 = three languages used 4 = four languages used
T01	Customer participation	Ordinal	1 = Very low 2 = Low 3 = Nominal 4 = High 5 = Very high
T02	Development environment adequacy	Ordinal	✓
T03	Staff availability	Ordinal	✓
T04	Standards use	Ordinal	✓
T05	Methods use	Ordinal	✓
T06	Tools use	Ordinal	✓
T07	Software's logical complexity	Ordinal	✓
T08	Requirements volatility	Ordinal	✓
T09	Quality requirements	Ordinal	✓
T10	Efficiency requirements	Ordinal	✓

T11	Installation requirements	Ordinal	✓
T12	Staff analysis skills	Ordinal	✓
T13	Staff application knowledge	Ordinal	✓
T14	Staff tool skills	Ordinal	✓
T15	Staff team skills	Ordinal	✓
Duration	Duration	Numerical	Duration of project from specification until delivery, measured in months
Size	Application size	Numerical	Function points measured using the experience method
Effort	Effort	Numerical	Work carried out by the software supplier from specification until delivery, measured in hours