UNIVERSITY OF GHANA

COLLEGE OF BASIC AND APPLIED SCIENCES

MACHINE LEARNING ALGORITHMS ON SMALL-SIZED DATASETS IN SOFTWARE EFFORT ESTIMATION: A COMPARATIVE STUDY

BY

SAMUEL ABEDU

(10702420)

THIS THESIS/DISSERTATION IS SUBMITTED TO THE UNIVERSITY OF GHANA, LEGON IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF MPHIL COMPUTER SCIENCE DEGREE

DEPARTMENT OF COMPUTER SCIENCE

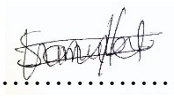
JANUARY, 2021

DECLARATION

I hereby declare that this thesis is my original research work undertaken at the Department of Computer Science, University of Ghana, Legon under the guidance of my thesis supervisors. Other people's work have been duly cited and acknowledged.

STUDENT

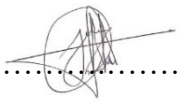
Name: Samuel Abedu

Signature: 

Date: …January 27, 2021……

SUPERVISOR

Name: Dr. Solomon Mensah

Signature: 

Date: …January 29, 2021……

CO-SUPERVISOR

Name: Dr. Isaac Wiafe

Signature: 

Date: …January 29, 2021……

ABSTRACT

Context: Software effort estimation is crucial in the software development process. Overestimating or underestimating the effort for a software project can have consequences on the bidding or development process for a company. Over the years, there has been growing research interest in machine learning approaches in software effort estimation. Though deep learning has been described as the state of the art in the field of machine learning, much has not been done to assess the performance of deep learning approaches in the field.

Objective: This study defines a discretization scheme for setting a threshold for a small-sized dataset in software effort estimation. Also, it investigates the performance of selected machine learning models on the small-sized datasets.

Method: Software effort estimation datasets were identified with their number of project instances and features from existing literature and ranked according to the number of project instances. Eubank's optimal spacing theory was used to discretize the ranking of the project instances into three classes. The performance of selected conventional machine learning models and two deep learning models were assessed on the datasets classified as small-sized. The leave-one-out cross-validation as recommended by Kitchenham was adopted to assess the training and validation needs of the selected model. The performance of each model on the selected datasets was measured using the mean absolute error (MAE). Robust statistical tests were conducted using the Yuen's t-test and Cliff's delta effect size.

Results: Results showed that the conventional machine learning models achieved improved prediction performance as compared to the deep learning models. Nonetheless, after applying early stopping regularisation to the deep learning models, it was found that the deep learning models achieved improved prediction accuracy than the conventional machine learning models but failed to outperform the Automatically Transformed Linear Model (ATLM).

Conclusion: The study concluded that conventional machine learning approaches achieve better performance than deep learning approaches on small-sized datasets. However, applying the early stopping regularisation technique to the deep learning models can improve the performance of the deep learning model. Also, a given software effort estimation dataset can be classified as small-sized if the number of project instances in the dataset is less than 43.

Keyword: Deep learning, Machine learning, Software effort estimation, Small-sized dataset

DEDICATION

To my family especially my mum (Mrs. Georgina Abedu).

ACKNOWLEDGEMENT

I wish to express my profound gratitude to my creator, God Almighty, granting me protection, wisdom, knowledge and skill throughout the writing of this thesis.

Secondly, I acknowledge the effort of my supervisors (Dr. Solomon Mensah and Dr. Isaac Wiafe) for their influence on my academic pursuit. Your constructive criticisms and encouragement have undoubtedly been phenomenal and improved the originality of this thesis.

I would also express my profound gratitude to Dr. Ferdinand A. Katsriku and Dr. Jamal-Deen Abdulai for their advice and encouragement. I am also grateful to the entire staff of the Department of Computer Science for their indispensable help and support in making my thesis a success.

To my colleagues (William, Abigail, Akon, Fred, Jacqueline, and Melody), thank you for being there for me through this period. Finally, to anyone who contributed in one way or the other to the success of my studies, I thank you and God richly bless you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ATLM	Automatically transformed linear model
BN	Bayesian network
CART	Classification and regression tree
COCOMO	Constructive cost model
DNN	Deep neural network
ElasticNet	Elastic Network
ESE	Empirical software engineering
FPA	Function point analysis
KNN	K-nearest neighbour
LOOCV	Leave one out cross validation
LSTM	Long-short term memory
MAE	Mean absolute error
MMRE	Mean magnitude of relative error
OLS	Ordinary least squares
PROMISE	Predictor models in software engineering
SEE	Software effort estimation
SLIM	Software lifecycle management
SVM	Support vector machine
UCP	Use case point

CHAPTER ONE

1 INTRODUCTION

1.1 Background of the study

The accurate estimation of effort for a software project is a challenge for software development researchers and practitioners (Dragicevic et al., 2017). Software effort estimation (SEE) is the process of estimating effort in terms of cost, and allocation of resources to enable timely delivery of projects within budget. Overestimation or underestimation of project efforts may lead to devastating consequences to organizations (Seo & Bae, 2013). For instance, effort overestimation may result in contract loss or resource wastage whilst underestimations lead to poor quality or uncompleted projects (Minku & Yao, 2013a; Zhang et al., 2015).

In most cases, estimations are associated with several complexities. This is as a result of the multidisciplinary nature of software projects and its related uncertain variables (Zare et al., 2016). Likewise, the precise effort of a project is mostly determined when the project is closed (Fuentetaja et al., 2013). Factors such as methods for development, organization type and programming languages make estimations fuzzy (Akhtar, 2013).

Currently, three approaches are widely used for software effort estimation (SEE): expert judgements, parametric models and machine learning methods. These methods aim to increase estimation accuracy and have been demonstrated to support project managers and software engineers in project estimations (Sharma & Singh, 2018).

Expert judgement involves the consultation of single or multiple experts. Experts use their experience, information accessible to them from past projects and their understanding of new projects to estimate efforts of new projects. This approach is a subjective reasoning process and their assessment can be biased (Moløkken-Østvold & Jørgensen, 2004; Rush & Roy,

2001). The biases in expert judgement could also be as a result of political pressure in the company. Notwithstanding the biases in expert judgement, it is the preferred option for new companies and those that lack proper documentation of experience from earlier projects (Host & Wohlin, 1998). Moløkken-Østvold and Jørgensen (2004) recommended using a group of experts to reduce the biases in expert judgement.

Parametric models, on the other hand, uses analytical and statistical methods to estimate effort. An example of such models is the Constructive Cost Model (COCOMO): a regression model built on the number of lines of code in a project (Boehm, 2001). Parametric models do not consider the skills of the development team or the organizational culture when estimating effort for a project. Pospieszny et al. (2018) argued that parametric models may not be feasible for modern software development practices like code reuse, and agile development.

Machine learning has been explored as an approach for estimating effort in recent SEE studies. Techniques like Classification and Regression Tree (CART) (Dejaeger et al., 2012), K-Nearest Neighbour (KNN) (Satapathy & Rath, 2017), Elastic Network Regression (ElasticNet) (Mensah et al., 2018), Support Vector Machine (SVM) (Azzeh & Nassif, 2016) have all been used for effort estimation. Wen et al. (2012) explained that machine learning approaches have achieved accuracies close to acceptable levels (i.e. MMRE $\leq$ 25% or Pred(k) $\geq$ 75%) and have better prediction accuracy than non-machine learning methods such as Software Lifecycle Management (SLIM) and Function Point Analysis (FPA) (Albrecht & Gaffney, 1983).

Building accurate machine learning models for effort estimation depends on large and quality data. However, there have been issues raised over the size and quality of data in empirical software engineering (ESE) including effort estimation datasets (Bosu & Macdonell, 2019). This challenge presents the research community with opportunity in investigating techniques suitable for utilising the small ESE data which includes software effort estimation datasets.

1.2 Problem Statement

Deep learning, known as the state-of-the-art machine learning, has been largely successful in several domains, including computer vision (Krizhevsky et al., 2012; Voulodimos et al., 2018), speech recognition (Deng & Platt, 2014), defect prediction (Yang et al., 2015), financial analysis (Fischer & Krauss, 2018) yet its adoption in SEE is lacking. Deep learning has been demonstrated to thrive on large datasets (LeCun et al., 2015). However, the ability to acquire large dataset for effort estimation is a challenge because owners of effort data are reluctant to share data due to privacy concerns. Thus, researchers and practitioners use small-sized data for estimations (Qi et al., 2017). Also, the computational cost for model training in deep learning serves as a challenge for its successful use (Justus et al., 2019).

This study will investigate the prospects of deep learning on small-size effort estimation datasets. Precisely, the study will compare the mean absolute errors (MAE) of three conventional machine learning models (Elastic Network Regression, Bayesian Network and Classification and Regression Tree) and two deep learning models (Deep Neural Network and Long-Short Term Memory). The results of these models will be benchmarked against the standard Automatically Transformed Linear Model (Whigham et al., 2015). This comparison is to ascertain whether deep learning approaches perform better on small-sized datasets than conventional machine learning algorithms and can be adopted for effort estimation tasks.

1.3 Scope of the Study

The study focuses on building predictive models from historical SEE datasets which are small-sized. Specifically, the study is a comparison of the prediction accuracies of conventional machine learning algorithms. It covers the application of conventional machine learning (ATLM, CART, BN, ElasticNet) and deep learning (DNN and LSTM) methods.

Historical data from single or cross-organizational project development in the PROMISE or Github repositories were utilised for the comparative study. The study also makes use of the mean absolute error (MAE) measure and robust statistical tests.

1.4 Research Aim and Objectives

The aim and specific objectives of this research is discussed in this section.

1.4.1 Research Aim

This study aims to investigate the prospects of deep learning on small-sized SEE datasets and also define a discretization scheme for classifying a given SEE dataset as small-sized. Currently, literature in SEE reports conventional machine learning methods for effort estimation. Studies using deep learning methods for effort estimation is lacking. Thus, this study investigates the prospects of deep learning in SEE by comparing its performance to that of conventional machine learning methods.

1.4.2 Specific Objectives

The study addresses the following specific objectives:

i. To review current datasets used in machine learning based SEE.
ii. To define a discretization scheme for classifying a given SEE dataset as small-sized, medium-sized or large-sized.
iii. To evaluate the prediction performance of deep learning and conventional machine learning algorithms on small-sized datasets.
iv. To assess the effect early stopping has on deep learning models.

1.5 Research Contribution

According to Gregor & Hevner (2013), knowledge contribution of research can be a design theory, model, construct, methods or instantiation. This study contributes to knowledge by

investigating the prospect of deep learning techniques in software effort estimation which is lacking. The study extends the application of a known tool (deep learning) to the SEE field. Thus, this is an exaptation study according to Gregor & Hevner (2013) knowledge contribution framework. The specific theoretical and practical contributions of the study is presented.

1.5.1 Theoretical Contribution

The study ascertains whether conventional machine learning has better performance than deep learning algorithms on small-sized datasets.

1.5.2 Practical Contribution

The study makes the following practical contributions:

i. Defines a discretization scheme for classifying a given SEE dataset as small, medium or large-sized.
ii. Presents an evaluation of deep learning on small-sized SEE datasets.
iii. Recommends early stopping as a way of improving deep learning model's performance on small-sized SEE datasets.

1.6 Organization of The Study

Chapter two of this dissertation presents preliminaries concepts needed to understand this study. Also, a summary of related works on machine learning approaches to effort estimation as well as a summary on deep learning is presented. Chapter two then concludes by presenting the gaps in previous studies that this study will seek to address.

Chapter three presents a systematic literature review on datasets used in SEE studies. The chapter also details the discretization scheme defined based on the results from the systematic literature review.

Chapter four discusses the empirical framework and the methodology employed for the comparative study. To be precise, a description of the datasets and the data pre-processing

techniques used are discussed. The machine learning and deep learning algorithms used in this study are also discussed in this chapter. How the experiments were setup is also discussed.

The results of the experiments are presented in chapter five. The MAE, p-values from the Yuen's t-test and Cliff's delta effect size for each studied model on each of the selected datasets are presented. A discussion of the results from chapter four is presented in chapter six.

Lastly, chapter seven presents a summary of this dissertation as well as the concluding remarks about the results obtained from the comparative analysis. The chapter also discusses threats that could have possibly affected the outcomes of the study. Finally, a discussion of possible future works arising from the dissertation is presented.

CHAPTER TWO

2 LITERATURE REVIEW

2.1 Overview

The aim of the study as stated in the previous chapter is to investigate the prospect of deep learning in software effort estimation by comparing its performance with conventional machine learning approaches. Therefore, this chapter presents the literature review. The literature review focuses on machine learning in software effort estimation to present the knowledge gap and the need for this research.

There are five sections in this chapter. The first section presents a review of software effort estimation. The second section is a review of deep learning. A review of studies adopting deep learning for effort estimation is presented in the third section. The fourth section identifies the knowledge gap and establishes the need for this study. The last section explains the preliminary concepts needed to understand the study.

2.2 Review of Software Effort Estimation

In this section, a review on the machine learning approaches to software effort estimation is presented. Also, a discussion on the sizes of SEE datasets and the feature selection methods in SEE is presented in this section.

2.2.1 Machine Learning Approaches to Software Effort Estimation

Researchers have proposed different machine learning models for SEE, yet these models vary in accuracy and error measurement metrics. Consequently, the need for a baseline model for comparison against the estimation models. Whigham et al. (2015) proposed a baseline model, known as the Automatically Transformed Linear Model (ATLM) to serve a benchmark for estimation. ATLM performs better than other estimation approaches reported to be standard

effort estimators (Whigham et al., 2015). Due to its performance and ease of implementation, it has been widely adopted as a benchmark for model comparison studies (Menzies et al., 2016; Sarro & Petrozziello, 2018; Song et al., 2018, 2019; Xia et al., 2018). However, these studies focused mainly on conventional machine learning models and shallow neural networks with less attention to deep learning models.

Mensah et al. (2018) compared the performance of six models in their duplex output SEE study to solve the issue of conclusion instability. They evaluated the performance of the ordinary least square regression, ridge regression, stepwise regression, Elastic network regression and the LASSO regression models benchmarked against the ATLM. The error metrics MAE, BMMRE, and adjusted R^2 were used in the study. The results showed that the ElasticNet regression model outperformed the other models followed by the ATLM. The models used in the study are conventional machine learning approaches, thus no consideration was given to deep learning approaches in the study. Nevertheless, the authors recommended evaluating deep learning approaches in subsequent studies.

Rahman and Islam (2019) performed a comparative study of machine learning algorithms to software effort estimation (SEE). They investigated the relationship between the project size in terms of the use case point (UCP) effort in person-hours. By comparing the root mean square errors of the functional neural network, extreme learning machine and the decision tree, they demonstrated that the decision tree algorithm provides a minimum of 10% better results for the small-sized project and 6% better results for medium-sized projects. The extreme learning machine achieved a 10% better results for the large-sized projects than the decision tree. The study failed to analyse the practical and statistical significance of their approach. This is a major drawback because without these robust statistical analyses, it difficult to confirm their hypothesis and efficiency of the proposed methods (Kitchenham et al., 2017).

The studies discussed above and more (Banimustafa, 2018; Hosni et al., 2017; Pospieszny et al., 2018; Sarro & Petrozziello, 2018; Song et al., 2019) performed comparative studies of different machine learning algorithms. To the best of our knowledge, none of the studies focused on deep learning. This study, therefore, investigates the prospect of deep learning in SEE by evaluating the performance of conventional machine learning and deep learning algorithms on small-sized SEE datasets.

2.2.2 Sizes of SEE Datasets

The unwillingness of huge software development companies to share project data due to privacy and security concerns (Qi et al., 2017) and the high cost associated with data collection renders effort estimation datasets to small (Song et al., 2018). Due to these challenges, the use of historic effort data in the public domain is on the rise. Jing et al. (2016) also warned that these data may contain missing inputs. Deleting or ignoring such inputs as used in reported in SEE studies could render these datasets relatively small after pre-processing.

Song et al. (2019) and Jing et al. (2016) recommended using synthetic data bootstrapping and data augmentation respectively to increase the size of effort estimation datasets. Adopting this recommendation will increase the size of the data from small to either medium or large. Thus, this study cannot adopt the recommendation as the focus is to evaluate the performance of the models on a small-sized dataset.

Again, the description of effort estimation datasets as small has been done ambiguously. Studies (Jing et al., 2016; Khatibi Bardsiri et al., 2014; Qi et al., 2017; Song et al., 2019; Song et al., 2005) all described SEE datasets as small-sized. However, the size of the dataset used in these studies vary and none of the studies presented the criteria used for describing the dataset as small-sized.

2.2.3 Feature Selection Techniques In SEE

Feature selection has been used in SEE to reduce dimensionality and eliminate redundant features that may be present in datasets (Hosni et al., 2017). Eliminating the redundant features helps to reduce overfitting problem and better understand the effect of each attribute (effort drivers) in the prediction model (Azzeh et al., 2008; Chen et al., 2005; Hosni et al., 2017).

Mensah et al. (2018) evaluated the performance of different feature selection approaches. Precisely, they evaluated the genetic search, best first, subset forward selection, linear forward and the random search algorithms to feature selection. Based on the findings of the study, Mensah et al. (2018) recommended the genetic search as an optimum feature selection approach in effort estimation. However, the genetic search algorithm to feature selection is unlikely to select optimum features when the size of the dataset is small (Harik et al., 1999; Tong & Mintram, 2010). Due to the scope of this study, (focus on small-sized SEE dataset), the genetic search approach to feature selection cannot be adopted.

Mensah et al. (2018) did not consider the evaluation of the correlation-based feature selection (Hall, 2000) approach. However, Minku and Yao (2013b) explained that the performance of SEE prediction models improve significantly when correlation-based feature selection with greedy stepwise search is used. Accordingly, this study adopts the correlation-based feature selection approach.

2.3 Review on Deep Learning

Machine learning techniques have been applied in studies from a variety of domains like computer vision, and healthcare (Hara et al., 2014; Voulodimos et al., 2018; Zander et al., 2005; Zhang & Tsai, 2005). Machines learning techniques allowed computers to perform various tasks without being explicitly programmed. They learn from data to allow computer or agents make predictions or data-driven decisions (Cavalcante et al., 2019; Ning & You, 2018;

Tsoukalas et al., 2015). Conventional machine learning techniques cannot learn representations from data in their raw format. Building an efficient machine learning model required careful engineering to design a feature extractor capable of selecting relevant information from the raw data for the model training. This challenge inspired the inception of multi-layered neural networks popularly known as deep learning (LeCun et al., 2015).

Deep learning techniques learn high-level abstraction in data by using hierarchical layers and statistical techniques (Marcus, 2018). They are good at discovering intricate structures in high dimensional data and therefore are applicable in many domains of science (LeCun et al., 2015). Appreciable progress has been made in the world of artificial intelligence since its inception. Deep learning models have shown successes in beating records in image recognition and speech recognition (Hinton et al., 2012; Krizhevsky et al., 2012). Also, the field of medicine is one industry benefiting from deep learning emergence. Image processing and analysis are been used to analyse and classify medical images to detect tumours and fractured bones that need skilled professionals to detect (Neill, 2013). Deep learning is been used to provide predictive support to physicians to assist them in disease diagnosis and provide treatment based on data available in the industry (Bini, 2018).

Deep learning techniques require large volumes of data and high computational power to successfully learn intricate structures in the dataset to be able to generalise well on unseen datasets (Xue-Wen Chen & Xiaotong Lin, 2014). Models trained in small datasets suffer from overfitting which is a challenge (Feng et al., 2019; Ng et al., 2015; Perez & Wang, 2017).

2.3.1 Challenges of Small-Sized Dataset in Deep Learning

Deep learning thrives on large datasets to learn representation and patterns in the inputs to build relationships to the outputs (Sawada & Kozuka, 2015; Xue-Wen Chen & Xiaotong Lin, 2014; Zhao et al., 2019). Using a small amount of data in training deep learning models results in the

overfitting problem (Feng et al., 2019; Ng et al., 2015; Perez & Wang, 2017). As stated earlier, during training, the model is required to learn the patterns in the dataset. However, if the amount of training data is small, the model may memorize the dataset instead of mapping it from input to output (Bilbao & Bilbao, 2017). This leads to poor generalization on unseen data i.e. the model performs well on the training data but performs poorly on unseen data (Bilbao & Bilbao, 2017; Hawkins, 2004).

There is active research in the fields of healthcare, agriculture, computer vision and material science to reduce the overfitting problem of deep learning models on a small amount of dataset (Feng et al., 2019; Garcia & Barbedo, 2018; Souza et al., 2018; Wong et al., 2018; Xu et al., 2018). Data augmentation, transfer learning, curriculum learning, regularisation techniques and feature extraction have all been suggested to reduce the overfitting problem (Pasupa & Sunhem, 2016; Perez & Wang, 2017; Wang et al., 2018; Wong et al., 2018).

Transfer learning requires using an existing pre-trained model in the domain and fine-tuning the weights to solve the problem specific to the data (Pan & Yang, 2010). Prior to the study, there were no available pre-trained models in software effort estimation that could be utilized for this study. On the other hand, this study argues that adopting a data augmentation approach would deviate from the set objectives. Adopting a data augmentation approach would increase the size of the dataset from small to either medium or large, thus it will defeat the goal of investigating the prospect of deep learning on small-sized datasets. Hence, The early stopping regularisation technique was adopted for this study over the penalized forms of regularisation because of the lower computational complexity of the early stopping (Raskutti et al., 2014).

2.3.2 Early Stopping

As mentioned earlier, the goal of deep learning model training is to obtain a model with optimum generalisation performance. During model training, the model appears to be getting

better i.e. the error on the training set appears to decrease after each epoch. However, it gets to a point when the model gets worse i.e. the error on the unseen increases (Geman et al., 1992). Overfitting is detected by the behaviour of the loss function which is optimised by the gradient descent used (Poggio et al., 2018).

Gradient descent optimization algorithm enforces implicit regularisation controlled by the number of iterations and asymptotically converges to the minimum norm solution for appropriate initial conditions of gradient descent (Poggio et al., 2018). Thus, there is an optimum early stopping that reduces overfitting of the loss function. This property of the gradient descent is effective for many loss functions and is relevant especially for regression models (Poggio et al., 2018).

Overfitting during neural network model training can be detected with cross-validation and stopped before convergence is achieved. However, care must be taken not to stop the model training when the training error decreases quickly. The reason being that generalisation error has chances of being repaired and overfitting does not start until the error only decreases slowly (Prechelt, 1998b).

2.4 Deep Learning and Software Effort Estimation

Deep learning approaches to effort estimation has not been extensively presented in literature. Nonetheless, studies by (Choetkiertikul et al., 2019; Mensah et al., 2018) have adopted deep learning. Shallow neural networks i.e. neural networks with a single hidden layer has also been utilised in studies (Banimustafa, 2018; Mittas et al., 2015; Nassif et al., 2013; Pai et al., 2013; Rijwani & Jain, 2016; Whigham et al., 2015). However, this study was interested in neural networks with multiple hidden layers. The deep neural network (DNN) and the long-short term memory are the two deep learning approaches presented in SEE literature. The LSTM has adopted mainly by studies on the agile effort estimation approach.

Mensah et al. (2018) adopted the deep neural network as the only estimation model to investigate the significance of bellwether moving window in improving software effort estimation. They argued that the deep neural network as a state-of-the-art machine learning approach has better prediction accuracy than other approaches including the ATLM. The LSTM was adopted by Choetkiertikul et al. (2019) to estimation story points (effort measurement) in agile development. The reason the authors used the LSTM was that the LSTM has few trainable parameters compared to the DNN and a memory cell for information retention.

The LSTM has not been employed in effort estimation studies on traditional software development but agile development. This study will be the first to investigate the use of LSTM on traditional effort estimation data.

2.5 The Knowledge Gap

The studies discussed in this chapter have made contributions in the domain of SEE. However, more work is needed to improve the accuracy of prediction models and also validate the use of state-of-the-art prediction techniques in SEE. Previous studies have reported that the datasets used for building SEE prediction models is small in terms of the number of project instances present in the data (Qi et al., 2017; Song et al., 2018). However, these studies did not give a threshold for describing SEE datasets as small. This leaves an open question for researchers to address as to what SEE datasets can be classified as small-sized.

Again, studies investigating the use of deep learning models is SEE is lacking. Considering the claim that SEE datasets are small in size and accurate deep learning models require large size data for training, there is the need to investigate the prospect of this state-of-the-art machine learning technique on the small-sized SEE datasets. This study acknowledges that conventional machine learning models have better prediction accuracy than deep learning models on small-

sized datasets. So, this study investigates the prospects of deep learning in SEE by comparing its performance on smaller datasets to the performance of conventional machine learning models.

2.6 Preliminary Concepts

2.6.1 Eubank's Optimal Spacing Selection

The Eubank's (1981) optimal spacing selection is an efficient approach for defining thresholds in discretizing values in a continuous space. Mensah et al. (2018) argued that the Eubank's optimal spacing selection approach produces optimal spacing selection threshold for discretising effort into classes. They used the Eubank's optimal spacing selection approach to classify effort predictions into three class, that is low effort, moderate effort and high effort in the study to solve the conclusion instability problem in software effort estimation. Accordingly, it was adopted to define a threshold for the various class of sizes (small, medium or large)

Eubank, (1981) introduced a density quantile function approach to optimal spacing selection for linear estimation of location. The spacing selection approach introduced by Eubank, (1981) uses the sample quantile function over an interval, say, $[p,q] \subset [0,1]$ to estimate the location and scale parameters for a censored set of order statistics. The sample quantile function produces optimal asymptotic spacings. The asymptotically optimal spacings agree with the optimal spacings, thus, it can be used as a threshold value for the spacing selection. The density quantile function is defined in equation (*2.1*).

$$Q(u) = F^{-1}(u), 0 \leq u \leq 1 \quad \textbf{(2.1)}$$

Where F is the censored set distribution function and u is the location parameter for defining the asymptotically optimal spacing.

2.6.2 Kernel Density Plot

Kitchenham et al. (2017) recommended the kernel density plot be used for visualising distribution in datasets. They argued that the kernel density plot provides sufficient information on distributions of datasets when compared to boxplots.

Histograms are also appropriate ways of visualising the distribution in datasets. However, there can be variations in the histogram plot for the same dataset when the number of bins is varied. Tukey (1977) resolved this variation by using the kernel density estimation. The kernel density estimation solves the probability density function of a variable by smoothening its histogram. The total area under the kernel density curve is one.

2.6.3 Statistical Test

The statistical significance of a prediction is affected by the size of the dataset used to train the prediction model. Obtaining a high statistical significance power for models trained with small-sized data is challenging (Bosu & Macdonell, 2019). Empirical software engineering studies test for the statistical significance power of models using Yuen's t-test, Scott-Knott test and Kruskal-Wallis.

Kitchenham et al. (2017) recommended a non-parametric test (t-test) for measuring the statistical significance when the size of a dataset is small but normally distributed. Kitchenham et al. (2017) argued that t-tests are appropriate based on evidence from the Central Limit Theorem. The t-test adopted for this study is Yuen's t-test.

Yuen's test (Yuen, 1974) is robust pairwise testing on two vectors of data samples based on their trimmed means. The Yuen's test is reduced to Welch's t-test (Welch, 1947) if the means of the data samples are not trimmed. The trimmed means is used as a measure of the central location of the data samples to test for the significant difference between them.

2.6.4 Practical Significance of Prediction Models

Measuring the effect size of observations is important irrespective of the statistical significance. Statistical significance tests focus on rejecting the null hypothesis and do not provide information on the magnitude of the impact of the observation (Macbeth et al., 2010). There are several effect size measurements like the Cohen d (Cohen, 1988), Glass delta (GLASS, 1976) and Hedge's g (Hedges, 1981). The above effect size measurements are appropriate for observations that obey the normality and homoscedasticity assumptions. For non-normal distribution of observation, Cliff's delta method is recommended measure (Macbeth et al., 2010).

Kitchenham et al. (2017) also recommended Cliff's delta as a robust non-parametric measure to find the practical significance of proposed interventions in the field of software engineering. Cliff's delta measures the overlap between the two data samples, in the case of this study, the actual effort for the project and the estimated effort for the project. Cliff's delta is not affected by outliers because it does not assume the data sample to follow a distribution. Nonetheless, outliers in the data sample for this study will be identified and removed to ensure models are not affected.

2.7 Chapter Summary

The chapter reviewed the application of machine learning for software effort estimation. The review acknowledged that there has been progress in research on machine learning techniques for effort estimation. However, the review identified that two studies adopted deep learning for effort estimation with one study (Choetkiertikul et al., 2019) focusing on agile development effort estimation. The number and publication year of these studies showed that much work has not been done to investigate deep learning approaches in effort estimation compared to studies on conventional machine learning approaches. Also, it was identified that SEE datasets have been described as small-sized. Yet, the studies did not state the criteria for describing

these datasets as small-sized. These identified knowledge gaps formed the basis and justification for this study.

The aim of this study is investigating prospect deep learning on smaller-sized effort estimation datasets. To this effect, this chapter included a review on the application of deep learning on small-sized datasets in other domains. The reason being that the two software effort estimation studies which used deep learning did not focus on small-sized SEE datasets. From the review, Early Stopping regularisation was identified as a suitable technique for improving the performance of the deep learning models in this study. Also, the preliminary concepts adopted in this study were discussed in this chapter. The discussion of these concepts included the theoretical justification for their use in this study.

CHAPTER THREE

3 PRELIMINARY STUDY

3.1 Overview

Software effort estimation datasets have been described as small-sized ambiguously in literature. There is no defined threshold for describing these datasets as small-size. Studies (Jing et al., 2016; Khatibi Bardsiri et al., 2014; Qi et al., 2017; Song et al., 2019; Song et al., 2005) have described SEE datasets as small-sized. However, the datasets these studies used vary. To solve this problem, this study introduces a discretization scheme for defining relatively small-sized effort estimation dataset. To build the discretisation, there was a need to survey all current SEE datasets.

This study extracted current SEE datasets by conducting a systematic literature review on SEE datasets used for machine learning models. The extracted datasets are then used to develop the discretization scheme.

3.2 Chapter Objective

The objective of this chapter is to develop a discretization scheme for describing relatively small-size SEE datasets. To achieve this, a systematic literature review in the area field of software effort estimation was conducted. The review focused only on studies using machine learning approaches for estimating effort. The review answered one research question, i.e. what are the datasets used in the study?

3.3 Search and Selection Process

3.3.1 Selection Criteria

To extract datasets used to train effort estimation models, the following inclusion and exclusion criteria were defined.

Inclusion Criteria

i. The study was published between 2010 and 2019.
ii. The study was published in a journal.
iii. The study uses a machine learning approach for the estimation.
iv. The dataset used is in the public domain or repository.

Exclusion criteria

i. Studies that were not published in English were excluded.
ii. Review articles were excluded as they do not perform empirical analysis.
iii. Conference papers were excluded with the assumption publicly available datasets that are presented in conference papers would have been already published in journals.

3.3.2 Search Process

To select relevant studies for the review, an automated search process was performed on three electronic databases (ACM digital library, IEEE Xplore and ScienceDirect). The selected electronic databases are amongst the popular databases in the field of Computer Science and Engineering, so a projection was made that it would contain relevant studies on the subject of research. Also, as part of the selection of the electronic databases, the rate of content update in these databases was considered.

3.4 Results of Review

The search for primary studies from the three electronic databases identified thirty-seven papers. Twenty-five (25) of the primary studies were retrieved from the Sciencedirect digital library. Majority of the papers were retrieved from the Sciencedirect database because contains more journals dedicated to empirical software engineering than the other two databases (ACM digital library and IEEE Explore). Seven (7) primary studies were retrieved from IEEE Explore and five (5) from the ACM digital library.

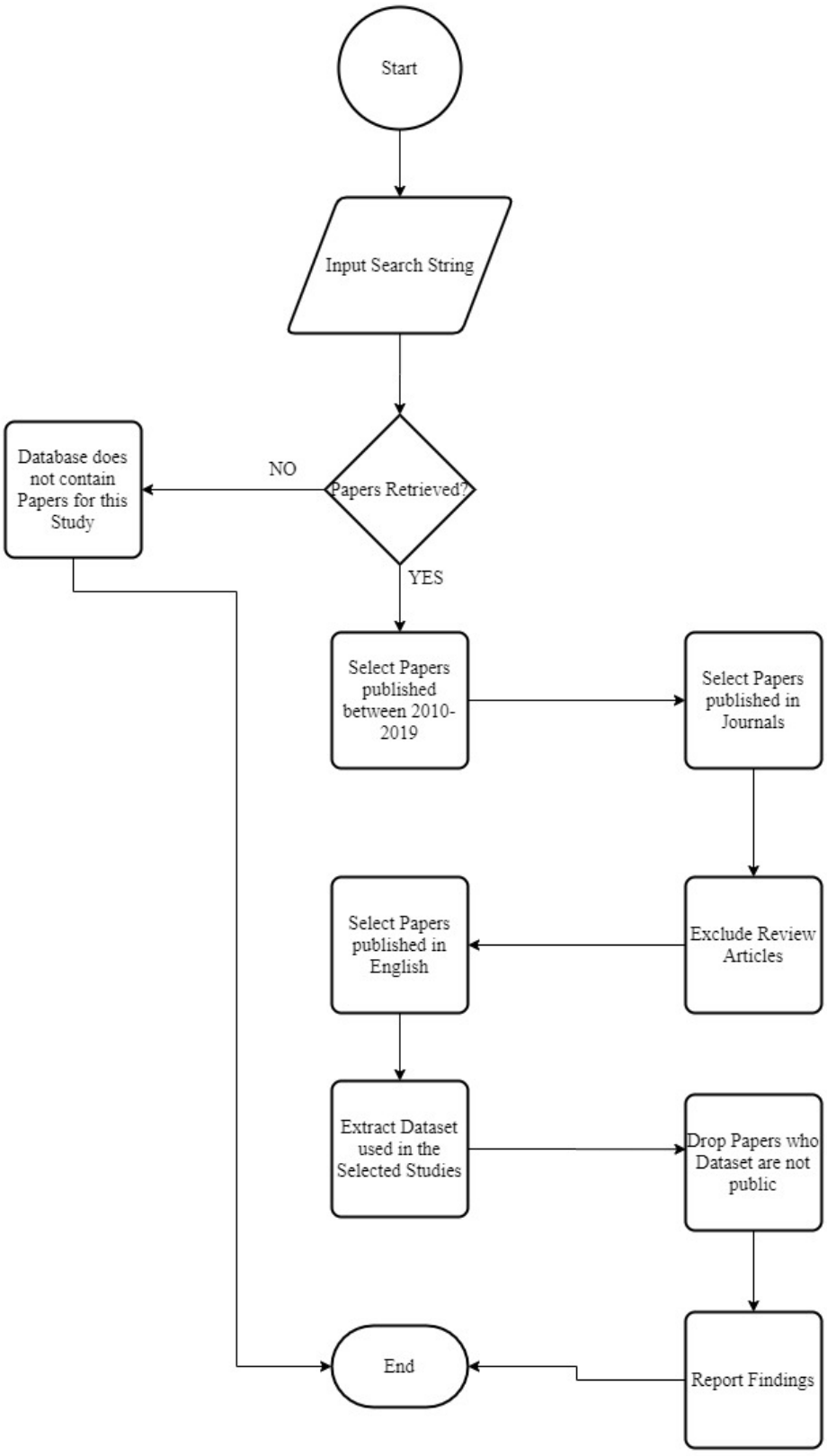


Figure 3.1: Flowchart of the Systematic Review Process

During the dataset extraction phase of the review, it was observed that some studies used datasets that are in the public repositories and ones that are not in any public repository. In such instances, the datasets not in the public domain were ignored and datasets in the public domain recorded. Dataset was identified as public when it is available in a public repository like the PROMISE repository or Github. Also, datasets that were presented by authors as an appendix in the study were identified as public datasets. This was included as public because they are available to researchers.

A total of twenty-three distinct datasets were extracted from these studies. The twenty-three datasets did not include datasets that are not in the public domain or have levels of restrictions to its usage. The dataset with the studies that used it is presented in table 3.1. COCOMO81 and Desharnais are the most datasets in the selected primary study. The two datasets, COCOMO81 and Desharnais are part of the oldest SEE datasets available and thus has the advantage of usage in SEE studies. Although these are the frequently used datasets, caution must be taken in using these as they may be obsolete and not suitable for modern software engineering practices (Mair et al., 2005). It was also noted that the Albrecht, Kemerer and Maxwell datasets have had a fair usage in SEE studies.

3.5 Discretization Scheme

This study agrees with Hua et al. (2005) that the number of features in a dataset should be strictly less than the number of instances to prevent overfitting. Assuming a dataset has N instances, then per the definition of Hua et al. (2005), the upper threshold for the number of features F should be $N - 1$. Following this, the Sdr dataset which has 23 features to 12 project instances was dropped for this study.

Table 3.1: Datasets and corresponding studies that used it

Dataset	Reporting Studies	Total Papers
Albrecht	P1, P2, P3, P5, P7, P8, P9, P11, P12, P17, P19, P23, P24, P32, P35	15
Atkinson	P21	1
Bielak	P31	1
China	P2, P3, P7, P11, P12, P17, P21, P24, P35, P36	10
Cocomo81	P1, P2, P3, P5, P7, P9, P10, P11, P14, P15, P16, P17, P18, P20, P22, P23, P24, P28, P29, P30, P35, P36	22
Cosmic	P17, P36	2
Desharnais	P1, P2, P3, P4, P5, P6, P7, P8, P9, P11, P12, P15, P17, P19, P22, P23, P24, P30, P31, P35	20
Finnish	P1, P3, P7, P12, P27, P35	6
Kemerer	P1, P2, P3, P7, P8, P9, P11, P12, P17, P23, P24, P32, P35, P36	14
Kitchenham	P3, P6, P10, P12, P17, P21, P36	7
Lopez-Martin, (2011) Data 1	P37	1
Lopez-Martin, (2011) Data 2	P37	1
Maxwell	P1, P2, P3, P4, P7, P10, P11, P12, P15, P17, P21, P22, P29, P35, P36	15
Miyazaki	P8, P17, P24	3

Miyazaki94	P1, P3, P7, P19, P35	5
Nasa	P2, P4, P11, P12, P13, P15, P17, P20, P23, P30, P34	11
Nasa93	P1, P3, P5, P7, P10, P12, P16, P20, P25, P26, P30, P35, P36	13
Qi et al. (2017) Java Data	P33	1
Qi et al. (2017)Web Data	P33	1
Sdr	P1, P3, P5, P7, P20, P30, P35	7
Telecom1	P1, P2, P3, P7, P8, P11, P12, P17, P21, P35	10
Tukutuku	P28	1
USP05	P4, P17	2

The remaining twenty-two (22) datasets from the systematic literature review were used to develop the discretization scheme. Table 3.2 shows these datasets with their respective number of features, project instances and ranking. The rankings of the size of the datasets were based on the number of project instances. The features were not considered in determining the size of the dataset since the number of reported features varied from each dataset examined.

In this study, a given dataset is classified as small, medium or large based on the number of project instances. The classification scheme was based on Eubank's optimal spacing theorem (Eubank, 1981). The theorem, making use of the quantile function was used to discretize the rankings of the number instances into three classes i.e. first quartile class (Q1), second quartile class (Q2), and the third quartile class (Q3). All studied datasets having their ranks to be less than or equal to the first quartile class were classified as the small-sized dataset. Alternatively, all datasets with their ranks greater than or equal to the third-class quartile were classified as

large size dataset. Then all datasets with their ranks between the first and third quartiles were classified as medium.

3.6 The Outcome of Discretization Scheme

From the discretization scheme defined, six datasets (Kemerer, Atkinson, Telecom1, Albrecht, Finnish and Cosmic) felled in the first quartile range. These datasets are classified as small-sized. The discretisation scheme defined a maximum threshold of 43 project instances for describing a given SEE dataset as small-sized. Thus, a given SEE dataset is small-sized if the number of project instances in the dataset is less than 44.

The discretisation scheme defined a minimum threshold of 147 project instances for classifying a dataset as large-sized. Thus, a given SEE dataset is large-sized if the number of project instances in the dataset is greater than 147. These datasets were the dataset presented by Qi et al. (2017) for estimating effort for a java software project, Tukutuku, Bielak, Dataset 1 from Lopez-Martin (2011), USP05 and China datasets.

Datasets with project instances within the ranges 44 and 146 inclusive are classified as medium-sized. Thus, ten datasets (Miyazaki94, Miyazaki, Qi et al. (2017) Web data, Nasa, Cocomo81, Maxwell, Lopez-Martin (2011) Data 2, Desharnais, Nasa93 and Kitchenham) were classified as the medium-sized dataset.

The threshold values for the classification were obtained from the use of the Eubank's optimal spacing theorem. A threshold of 43 project instances for small-sized. Dataset with instances between 44 and 146 inclusive as medium-sized and datasets with project instances greater than 146 as large-sized.

3.7 Chapter Summary

This chapter defined a discretisation scheme for defining relatively small, medium or large-sized software effort estimation dataset. The discretisation scheme was based on Eubank's

optimal spacing theory using the density quantile function. To build the discretisation scheme, there was a need to identify all current software effort estimation datasets. A systematic literature review on machine learning approaches for software effort estimation was conducted to identify all current SEE datasets.

The review identified 23 distinct SEE datasets that are in the public domain. Twenty-two (22) which satisfied the recommendation by Hua et al. (2005) were used to develop the discretisation scheme. Six datasets, namely the Albrecht, Atkinson, Cosmic, Finnish, Kemerer and Telecom1 were classified as small-sized and a threshold of 43 project instances was defined to classify a dataset as small-sized. Ten datasets were classified as medium-sized with the number of project instances within 44 and 146 inclusive. Six datasets were classified as large-sized with a lower threshold of 147 project instances.

Table 3.2: The discretisation of datasets in quartiles

Dataset	Source	Features	Instances	Rank	Quartile
Kemerer	PROMISE	7	15	1	Q1
Atkinson	PROMISE	14	16	2	Q1
Telecom1	PROMISE	3	18	3	Q1
Albrecht	PROMISE	8	24	4	Q1
Finnish	PROMISE	8	38	5	Q1
Cosmic	PROMISE	11	42	6	Q1
Miyazaki	PROMISE	8	48	7	Q2
Miyazaki94	PROMISE	8	48	8	Q2
Qi et al. (2017) Web data	Github	8	49	9	Q2

Tukutuku	(Mendes & Kitchenham, 2004)	9	53	10	Q2
Nasa	PROMISE	17	60	11	Q2
Cocomo81	PROMISE	17	63	12	Q3
Maxwell	PROMISE	24	63	13	Q3
Lopez-Martin, (2011) Data 2	(Lopez-Martin, 2011)	9	68	14	Q3
Desharnais	PROMISE	9	81	15	Q3
Nasa93	PROMISE	24	93	16	Q3
Kitchenham	PROMISE	9	145	17	Q4
Qi et al. (2017) Java data	Github	8	147	18	Q4
Bielak	(Bielak, 2000)	4	152	19	Q4
Lopez-Martin, (2011) Data 1	(Lopez-Martin, 2011)	9	163	20	Q4
USP05	PROMISE	17	203	21	Q4
China	PROMISE	17	499	22	Q4

Q1 denotes the first quartile
Q2 denotes the second quartile
Q3 denotes the third quartile
Q4 denotes the fourth quartile
PROMISE – PRedictOr Models In Software Engineering

CHAPTER FOUR

4 METHODOLOGY

4.1 Overview

This chapter presents the empirical framework and the experimental set up of the study. The empirical framework discusses the data pre-processing techniques, the feature selection techniques and the model evaluation adopted for the study. It also includes a description of the datasets and models used. The model design procedure is presented in the experimental set section.

4.2 Empirical Framework

These section presents the empirical framework of the study. Specifically, it discusses the datasets, the data pre-processing techniques, feature selection and performance measures used in the study.

4.2.1 Data Description

The study identified twenty-two (22) datasets from the PROMISE and Github repositories based on the systematic literature review in chapter 3. Six of the 22 datasets were classified as small based on the classification in section **3.6.** Thus, these six datasets were adopted for comparative analysis. All six datasets (Albrecht, Atkinson, Cosmic, Kemerer, Finnish, and Telecom1) satisfied Hua et al. (2005) proposition. The number of project instances, number of features, description of project type, the mean and standard deviation of all six datasets are presented in table 4.1.

Albrecht Dataset

The Albrecht dataset contains 24 project cases developed with third generational language (3GL) from IBM. Eighteen (18) of the projects were written in COBOL, four in PL1 and two in the DMS language. The dataset contains seven independent features for estimating effort. The effort is recorded in 1000-person hour.

Atkinson Dataset

The Atkinson contains 16 project cases undertaken to build a telecommunication product in the U.K. The dataset has 12 input features and the effort is recorded in man-month.

Cosmic Dataset

The Cosmic dataset contains 42 project case. The dataset has ten input features and the effort is recorded in man-month. The dataset is in the PROMISE repository but further information concerning the organization, programming language about the data is not available.

Table 4.1: Description of selected datasets for building prediction models

Dataset	Instances	Features	Description	Mean (Effort)	Std (Effort)
Albrecht	24	7	IBM DP Services Project	21.875	28.4179
Atkinson	16	12	Builds to a large telecommunications product at U.K. company X	456.062	241.07
Cosmic	42	10	N/A	4965.5	8457.1
Finnish	38	6	Data collected by the TIEKE organization from IS projects from nine different Finnish companies	7678.29	7135.28
Kemerer	15	6	Large business application	219.247	263.055
Telecom1	18	2	Enhancement to a U.K. telecommunication product	284.338	264.715

Finnish Dataset

The Finnish dataset is cross-company projects dataset. it was collected from nine different Finnish companies by the TIEKE organization. The dataset initially contained 40 project case, however, Kitchenham and Kansala (1993) reported that two of the project cases were removed due to missing values. This resulted in the dataset having 38 project instances and six input features. The effort is recorded in person-hours.

Kemerer Dataset

The Kemerer dataset is company-specific. It contains 15 completed business data processing projects of the same company. The dataset has six independent features that can be used to estimate project effort. The effort is measured in person-month.

Telecom1 Dataset

The Telecom1 dataset contains 18 software enhancement projects undertaken on a U.K. telecommunications product. The version of the dataset available for this study has two (2) input features for estimating effort.

4.2.2 Data Pre-Processing

High-quality data is crucial in training machine learning models (Ilg et al., 2017). To ensure data quality (i.e., address the issues of missing data, outliers, and influential points) all the datasets were pre-processed. The flowchart of the data pre-processing phase is shown in figure 4.1.

Missing Data Treatment

An analysis of all six datasets was performed to address missing data values. Columns of each project instance were checked for null or NaN values to remove such project instances. None of the datasets contained missing data value.

Resolving Outliers and Influential Data Points

Outliers and influential data points render data not fit to be used in building estimation models (Bosu & Macdonell, 2019). To make such data worthy to be used in building estimation models, the issue of outliers and influential data points must be resolved. Thus, this study adopted Mendes & Kitchenham's (2004) approach to resolve it. Investigations were conducted to identify the effect of outliers and influential points. The untransformed data was first fitted on an ordinary least square (OLS) model to observe the effect of outliers and influential points. After that, it was transformed and fitted on the OLS model again to observe the changes in the effect of the outliers and influential data points. Subsequently, the outliers and influential data points were identified and removed.

The outliers were identified based on the Cook's distance measure approach and kernel density plots suggested by Kitchenham et al. (2017). A data point is considered an outlier if its Cook's distance measure is three times more than the mean Cook's distance as suggested by Seo & Bae (2013).

On the issue of influential data points, DFFITS was used to determine such data points. Data points whose DFFITS was greater than one (1) were considered influential (Belsley et al., 2005; Mensah et al., 2018).

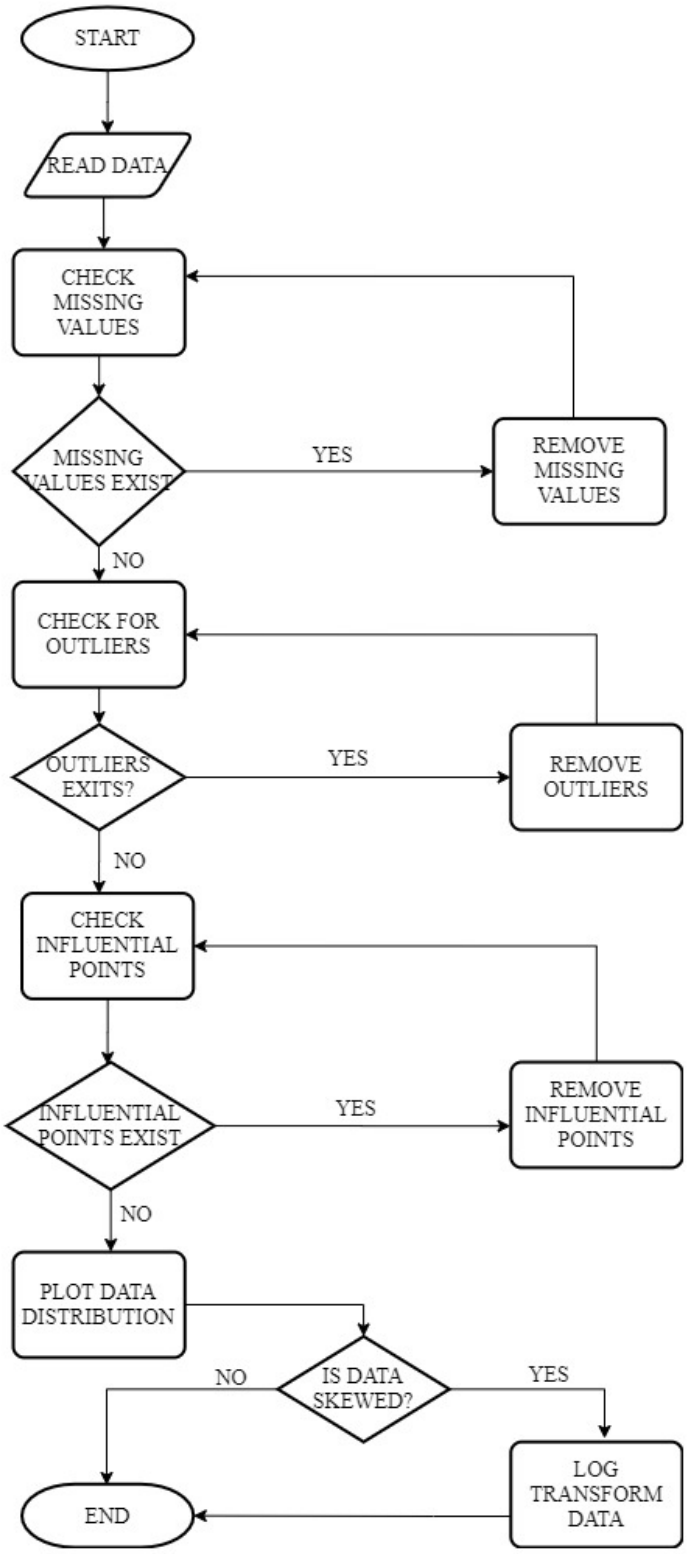


Figure 4.1: Flowchart of the Data Pre-processing Stage

4.2.3 Kernel Density Plots

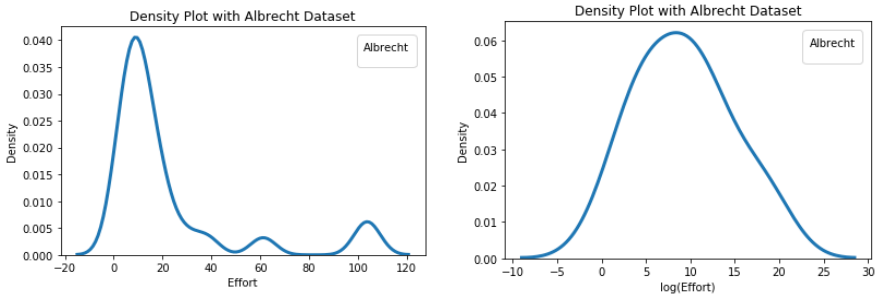
The study adopted the kernel density plot to visualise the distribution of the datasets. The kernel density plots showed that the distribution of the datasets improved after pre-processing.

The graph in figure 4.2a showed that the Albrecht dataset contained outliers and is skewed to the right. The distribution of Albrecht dataset was close to normal after the data pre-processing as shown in figure 4.2b.

The Atkinson dataset looked normally distributed before pre-processing as shown in figure 4.3a. However, it contained influential data points. After eliminating the influential points, the Atkinson dataset skewed to the left and thus was log transformed. The kernel density plot after the pre-processing is shown in figure 4.3b.

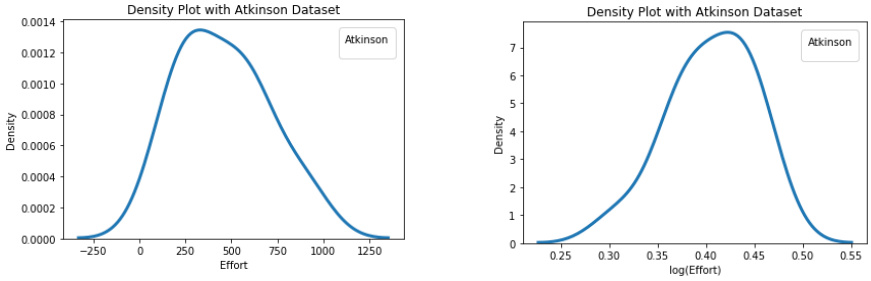
The plot for the Cosmic dataset was skewed to the right as shown in figure 4.4a. The Cosmic dataset contained outliers and influential data points. After pre-processing the Cosmic dataset to remove the outliers and influential data points and log transforming the Effort variable to reduce the skewness, the kernel density plot showed that the distribution was close to normal as shown in figure 4.4b.

Likewise, the Finnish, Kemerer and Telecom1 datasets also had their distributions skewed to the right due to the presence of outliers and influential data points. The plots for these datasets are shown in figures 4.5a, 4.6a and 4.7a respectively. After pre-processing these datasets, the kernel density plots showed that their distributions were close to that of a plot of a normal distribution. However, the plot of the Telecom1 dataset had two points were the log(Effort) values were concentrated. The plots after pre-processing are shown in figures 4.5b, 4.6b and 4.7b respectively.



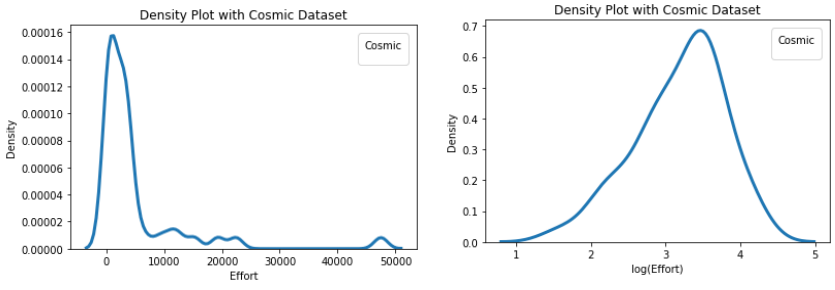
(a) (b)

Figure 4.2: (a) Kernel density plot of Albrecht dataset. (b) Kernel density plot of pre-processed Albrecht dataset



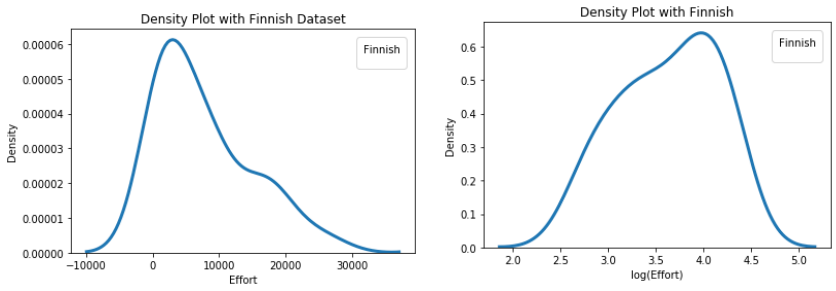
(a) (b)

Figure 4.3: (a) Kernel density plot of Atkinson dataset. (b) Kernel density plot of pre-processed Atkinson dataset



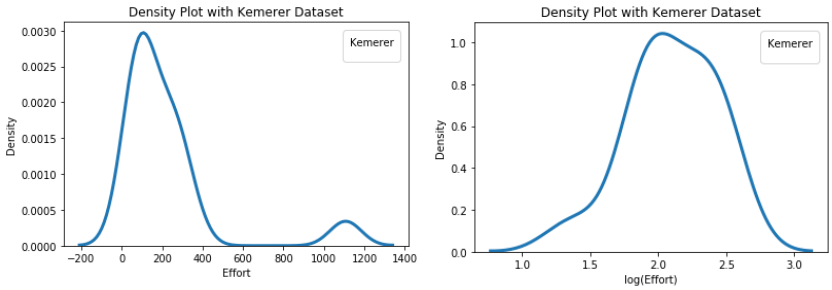
(a) (b)

Figure 4.4: (a) Kernel density plot of Cosmic dataset. (b) Kernel density plot of pre-processed Cosmic dataset



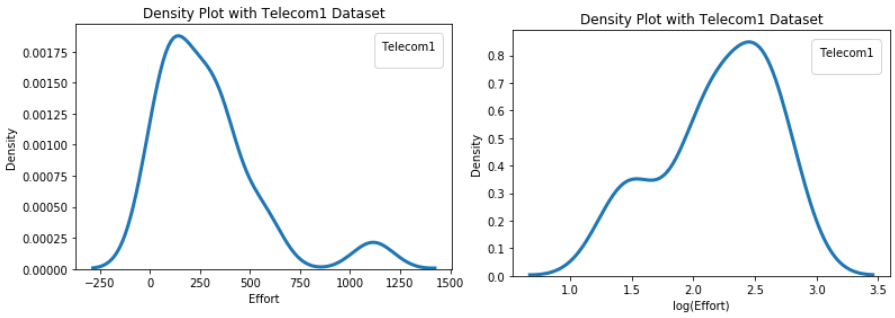
(a) (b)

Figure 4.5: (a) Kernel density plot of Finnish dataset. (b) Kernel density plot of pre-processed Finnish dataset



(a) (b)

Figure 4.6: (a) Kernel density plot of Kemerer dataset. (b) Kernel density plot of pre-processed Kemerer dataset



(a) (b)

Figure 4.7: (a) Kernel density plot of Telecom1 dataset. (b) Kernel density plot of pre-processed Telecom1 dataset

4.2.4 Feature Selection

To avoid multicollinearity, correlation-based feature selection (CFS) (Hall, 2000) was used. Minku and Yao (Minku & Yao, 2013b) explained that the performance of SEE prediction models improve significantly when correlation-based feature selection with greedy stepwise search is used to pre-process. Feature selection facilitates the selection of informative and relevant features to enhance model training. CFS is a multivariate filter approach to feature selection. Its output is dependent on the search strategy used (Hosni et al., 2017). In this study, the CFS with the best-first search approach was used. This according to Hall (2000) selects optimal features from a data compared to CFS with other search strategies like the forward selection. CFS evaluates and ranks feature subsets using Pearson's correlation. This is based on the hypothesis that, a good feature subset contains features highly correlated with the dependent variable, yet uncorrelated with each other.

Ghiselli (1964) formalised this heuristic as:

$$Merit_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \quad \textbf{(4.1)}$$

Where, $Merit_s$ is the heuristic "merit" of a feature subset set S containing k features, $\overline{r_{cf}}$ the average feature class correlation and $\overline{r_{ff}}$ the average feature-feature intercorrelation. The correlation coefficient between the features was set to 0.9. Each dataset with the total number of features and the number of features selected after the feature selection is shown in table **4.2**.

Table 4.2: The outcome of feature selection

Dataset	Number of Features	Features Selected
Albrecht	7	6
Atkinson	12	5
Cosmic	10	10
Finnish	6	5
Kemerer	6	5

Telecom1	2	2

4.2.5 Model Selection

Four conventional machine learning models and two deep learning models were adopted for this study. All the selected conventional machine learning models have been used in recent effort estimation studies and have been proved as effective effort estimation models. The two deep learning models are models suitable for regression problems and been used for effort estimation either in the traditional or agile development environment. This study is limited to these six estimation models.

Automatically Transformed Linear Model

The Automatically Transformed Linear Model (ATLM) was proposed by Whigham et al. (2015) to serve as a baseline model for comparative studies in SEE. The model is based on a multiple linear regression (MLR). Equation (*4.2*) in the form of MLR implemented in ATLM. The ATLM performs an automatic data pre-processing. It performs log and square-root transformations on the data and compares the skewness in the dependent and independent variables. The transformation that results in less skewness is used for model construction and effort prediction. The ATLM performs an inverse transform on the predicted effort values to make the predictions meaningful compared to the untransformed data.

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_n \beta_{ni} + \varepsilon_i \quad \textbf{(4.2)}$$

Elastic Net Regression

Elastic Net regression is recommended as a baseline SEE approach by Mensah et al. (2018). The elastic net (Zou & Hastie, 2005) performs regularisation and feature selection to eliminate highly correlated estimator variables before constructing the model. Given input data with N

observation pairs (x_i, y_i) and an approximated regression function $E(Y|X = x) = \beta_0 + x^T\beta$, the elastic net solves defined by Friedman et al. (2010)

$$\hat{\beta} = \min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left[\frac{1}{2N}\sum_{i=1}^{N}(y_i - \beta_0 - x_i^T)^2 + \lambda P_\alpha(\beta)\right] \quad \textbf{(4.3)}$$

Where P_α is the elastic net penalty given by

$$P_\alpha(\beta) = \left[\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j|\right] \quad \textbf{(4.4)}$$

For $j = 1, \ldots, p$. The P_α is used in finding highly correlated input variables (Zou & Hastie, 2005).

Mensah et al. (2018) was the first study to implement the elastic net model in effort estimation.

Bayesian Network

Bayesian network (BN) is a graphical model that encodes the probabilistic relationship between related variables. BN is determined by a pair as:

$$BN = (G, P) \quad \textbf{(4.5)}$$

Where G is a directed acyclic graph with nodes X_i and P is the local probability of all variables in the network. For instance, given node X_1 connected to node X_2 and node X_2 connected to node Y, then the conditional probability of finding Y as defined by Fuentetaja et al. (2013) is given as

$$P(Y/X_2, X_1) = P(Y/X_2) \quad \textbf{(4.6)}$$

The joint probability of each variable satisfies the Markov's condition that each variable X_i is conditionally independent of the set of all its non-descendants. The distribution is factorised as

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\pi(X_i)) \quad \textbf{(4.7)}$$

Where $\pi(X_i)$ is the set of nodes directly connected to X_i. The Bayesian network takes into account the probability effect of each input variable on the dependent variable.

Classification and Regression Tree

The classification and regression tree model recursively partitions data and fits a prediction model within each partition. It is represented graphically by a decision tree where dependent variables take continuous values for regression tree and a finite set of values for a classification tree.

Deep Neural Network

Deep neural network (DNN) is an artificial neural network with multiple hidden layers. The DNN finds a relationship between input variables and the output variable by assigning weights to the inputs. Mensah et al. (2018) recommended that DNN with two hidden layers is appropriate for effort estimation. Accordingly, the DNN model was set up with two hidden layers and the rectified linear activation (Relu) function. The Relu preserves the properties of linear models that make it easier to be optimised (Goodfellow et al., 2016). Also, the decision to choose the Relu function was informed by the logic that effort prediction should not be negative. The Relu is defined in equation (*4.8*) as

$$Relu(x) = \max(0, x) \tag{4.8}$$

Taking the weights assigned to the neuron as w_{ij} at the start of training. The weights are updated after each training epoch (t) to $w_{ij}^{(t+1)}$ where the change in weights $(\Delta w_{ij}^{(t+1)})$ is defined in equation *(4.9)* as

$$\Delta w_{ij}^{(t+1)} = \gamma \Delta w_{ij}^{t} - \alpha \frac{\partial L(.)}{\partial w_{ij}} \tag{4.9}$$

Where γ is the learning momentum, α is the learning rate and $L(.)$ is the loss function.

Long-Short Term Memory

The long-short term memory (LSTM) is a variant of the recurrent neural network that solves the vanishing gradient problem (Hochreiter & Schmidhuber, 1997). The LSTM uses a loop in the network that allows information to persist in the network. Given a data (X, Y) with $(x_1, \dots, x_n) \in X$ and outputs $(y_1, \dots, y_n) \in Y$. At step t the LSTM reads input x_t, the hidden state h_{t-1} and the previous memory c_{t-1} to compute hidden state h_t. The hidden state h_t is used to predict the output y_t. The memory cell is the element that stores the accumulated information over time. The structure of the LSTM network is shown in Figure 4.8.

The LSTM model was also set up with two LSTM layers and the glorot uniform kernel initializer was used. The Relu activation function was used for all layers in the network.

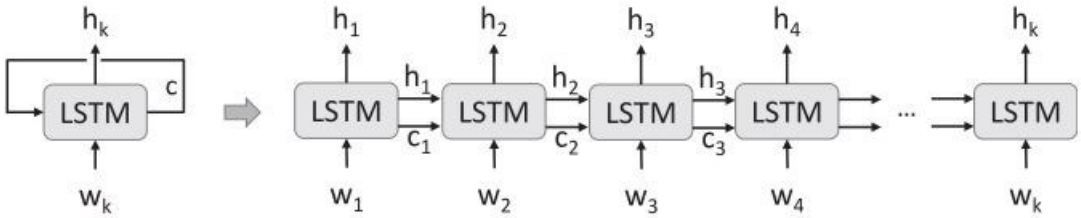


Figure 4.8: The LSTM network (Choetkiertikul et al., 2019)

4.2.6 Performance Measure

Mean absolute error (MAE) is a reliable and robust performance measure for assessing software effort estimation models (Foss et al., 2003; Shepperd & MacDonell, 2012). It has been successfully employed to evaluate the prediction accuracy of SEE models (Choetkiertikul et al., 2019; Hosni et al., 2017; Mensah et al., 2018; Song et al., 2019) and thus it was adopted for this study. The MAE is a risk function and it measures the average error magnitude. The MAE is defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|E_{A_i} - E_{E_i}\right| \quad \textbf{(4.10)}$$

where E_A is the actual effort of a project case or instance and E_E is the estimated effort of a project instance with n number of instances.

4.2.7 Statistical Significance of Models

As recommended by Kitchenham et al. (2017) Yuen's test (Yuen, 1974) was used to calculate the statistical significance of the models. This because, it is a robust statistical test that combines trimmed means and Welch's test (Welch, 1947) to compare the central location of two samples (Yuen, 1974). It is not sensitive to changes at lower tails because of the trimmed means, thus, the number of degrees of freedom is reduced (Kitchenham et al., 2017). Yuen's method gives better results in terms of type I error, probabilities and probability coverage (Wilcox, 2012). The confidence interval was set to 0.95 in this study.

4.2.8 Practical Significance of Models

Effect size is useful regardless of the statistical significance of a test statistic because it provides an objective measure on the practical importance of the experimental effect (Arcuri & Briand, 2014). Kitchenham et al. (2017) recommended using Cliff's method (Cliff, 1993) for finding the effect size of non-parametric problems. This study adopted Cliff's δ effect size to evaluate the practical significance of the models used. Cliff's δ effect size is defined in equation *(4.11)*. where $\boldsymbol{y_i}$ is the actual effort and the $\boldsymbol{\hat{y}_j}$ is the predicted effort of each dataset. The number of project cases in the actual and predicted classes are denoted by $\boldsymbol{n}$ and $\boldsymbol{m}$ respectively.

To interpret the practical significance, Kampenes et al. (2007) defined effect size as negligible ($\delta < 0.112$), small ($0.112 \leq \delta < 0.276$), medium ($0.276 \leq \delta < 0.427$) and large ($\delta \geq 0.427$). Mensah et al. (2018) agreed that results are misleading when the effect size is small or negligible and recommended using the effect size threshold of medium to large. However, this

study adopts an effect size threshold of negligible ($\delta < 0.112$), with the argument that there should not be a huge practical significance difference between the actual effort and the predicted effort.

$$\delta = \frac{SUM(y_i > \hat{y}_j) - SUM(y_i < \hat{y}_j)}{nm} \quad \textbf{(4.11)}$$

4.3 Experimental Set-Up

Six prediction models were each trained on the selected datasets. The models were set up using the leave-one-out cross-validation (LOOCV) (Kocaguneli & Menzies, 2013) and the model's predictions were evaluated using the mean absolute error (MAE). The flowchart of the experimental phase is shown in figure 4.9.

All the experiments were executed on Nvidia Graphics Processing Unit (GPU) laptop running on Ubuntu 18.0 Linux operating system. The Anaconda software with the Python3 and CUDA toolkit was installed. Python has efficient libraries for machine learning. The Sci-kit learn and Keras with Tensorflow libraries were installed. These are popular python libraries for conventional machine learning and deep learning. The CUDA toolkit provides the development environment for Nvidia GPU computing. The Keras and Sci-kit learn libraries contains the implementation of deep learning and conventional machine learning algorithms as well as the loss functions and tools for data pre-processing like the Pandas and NumPy.

4.3.1 Performance of Deep Learning Models Against Conventional Machine Learning Models

As mentioned earlier, two deep learning techniques, the Deep Neural Network (DNN) and the Long-short term memory (LSTM) were modelled against three conventional machine learning approaches, ElasticNet regression, Bayesian Network (BN) and Classification and regression tree (CART). These models were benchmarked against the Automatic transform linear model

(ATLM). Before setting up these models, all datasets were pre-processed using the data pre-processing procedure described in the data pre-processing section 4.1. Additionally, log transformation was applied as described by Mendes & Kitchenham, (2004) to the datasets to reduce the effect of skewness on the prediction model.

Chen et. al. (2005) argued that a model's performance can be improved through feature selection. Hence, correlation-based feature selection was applied as described in section 4.1 to select features that are relevant for model construction; features that are highly correlated with the predictive variable but uncorrelated with each other.

Similar to partitions adopted by (Dejaeger et al., 2012; Vinay Kumar et al., 2008), each of the datasets was partitioned in training and test sets in a holdout approach, thus 80% as a training set and 20% as the test set. This approach was adopted so models can be evaluated on unseen

data. Models were trained on the 80% set using the LOOCV. The final model is evaluated on the 20% test dataset. The MAE values are obtained from evaluating the performance on the test set and not values outputted during the model training.

4.3.2 Improving the Performance of Deep Learning Models

Based on the suggestion of Louati & Ktata, (2020), the effectiveness of early stopping as a way of improving the prediction performance of a model was investigated. Geman, et al. (1992) stated that all deep neural networks are prone to overfitting. It was observed that while the error measure of deep neural networks appeared to improve after each training epoch. However, it got to a point the error started increasing steadily. According to Prechelt (1998a), that is the point of overfitting. Goodfellow et al. (2016, p. 247) recommended early stopping regularisation technique as an approach to stopping the training. Accordingly, the study used early stopping to halt the model training. The models were again evaluated on the 20% test set to check if there were improvement in performance after early stopping.

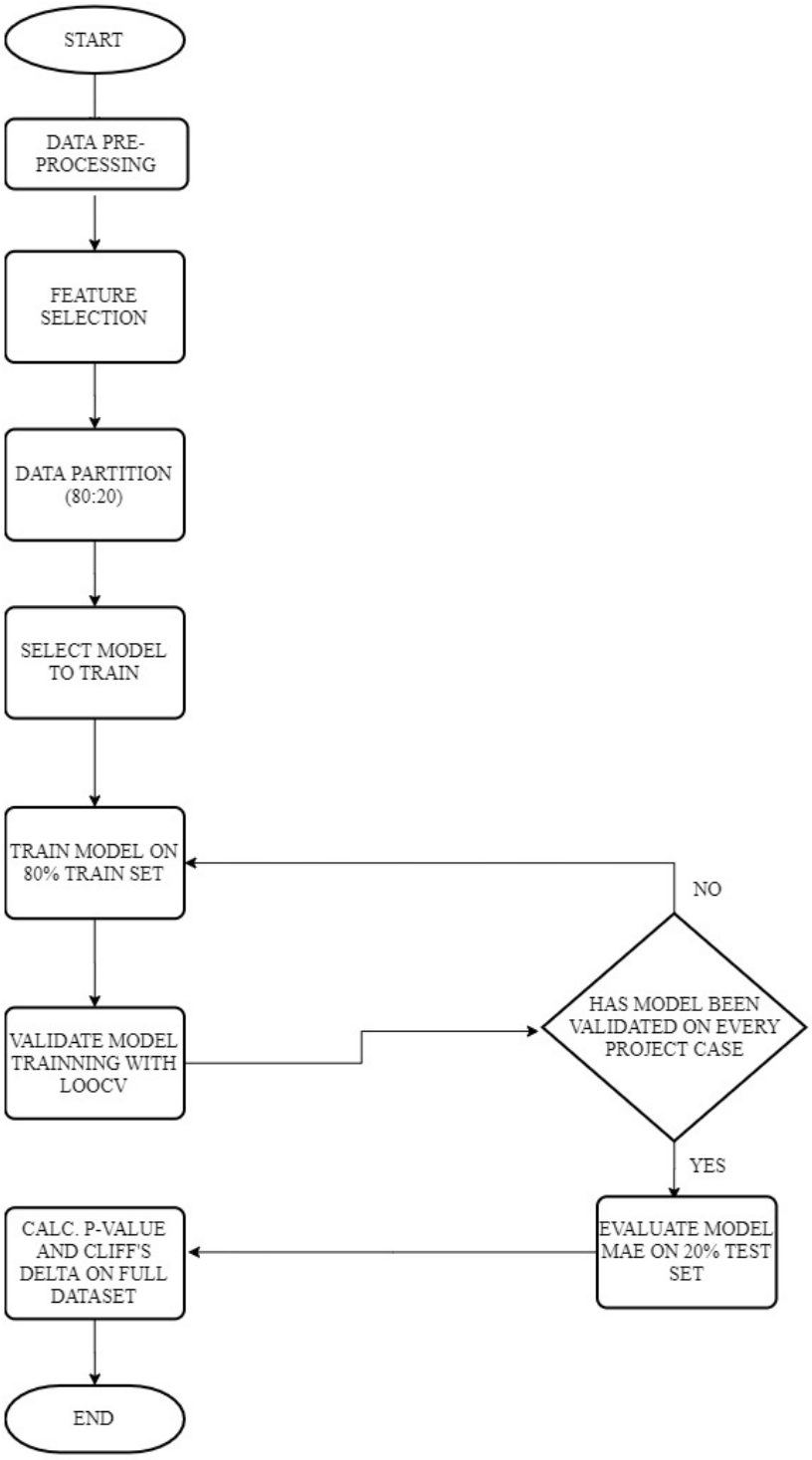


Figure 4.9: Flowchart of the Empirical Study

4.4 Chapter Summary

The chapter presented the empirical framework of the study. It included the description of the datasets used for the study, the data pre-processing techniques used, the feature selection method, model selection and evaluation measures.

It also described the experiment design which was in two phases. The first experiment designed to solve objective 3 of the study. A second experiment was designed to improve the prediction performance of the deep learning models using the Early Stopping regularisation. All the models were trained using the leave-one-out cross-validation. With the experiments completed, the next chapter presents the results.

CHAPTER FIVE

5 RESULTS

5.1 Overview

The preceding chapter discussed the empirical framework and the experimental set for the comparative study. This chapter presents the results obtained by the models. It evaluates the mean absolute error (MAE), p-values from the Yuen's t-test and Cliff's delta effect size for each studied model on each of the selected datasets. The focus is to compare the MAE of the deep learning models to the conventional machine learning models. The p-values and Cliff's delta effect size is to show the statistical significance and practical significance of each of the models.

5.2 Performance of Deep Learning Models Against Conventional Machine Learning Models

The graph of the recorded mean absolute errors (MAE) of the six selected models is shown in Figure 5.1. The results showed that conventional machine learning approaches produced better results on small-sized datasets compared to the deep learning approaches. The results corroborate findings in other fields like material science (Feng et al., 2019) that conventional machine learning approaches perform better than deep learning approaches on small-sized datasets.

The results showed that the ElasticNet regression model was outperformed by the deep learning approaches. The benchmark model, ATLM, had the best average MAE value of 0.1585 across all the six datasets. The Bayesian network (BN) and the classification and regression tree (CART) models achieved an average MAE of 0.1721 and 0.1804 respectively. The results from the deep learning models (DNN and LSTM) outperformed the ElasticNet model with an average MAE of 0.2026 and 0.2146 respectively.

The statistical significance of the results was assessed with the p-values. The p-values from the Yuen's test on the model's prediction were within the 0.05 asymptotic significance level. This shows that predictions from the models are statistically significant. Nonetheless, the deep neural network model trained on the Atkinson dataset had a p-value of 0.0629 which is above the acceptable 0.05 asymptotic level. Although the p-value was above 0.05, the study failed to reject the predictions from the DNN model because its prediction on the other datasets were statistically significant.

Evaluating the practical importance of the models showed that Cliff's δ effect size for all the models were negligible ($\delta < 0.112$) following Kampenes et al. (2007) definitions. Meaning there is a small difference between the actual effort values and the predicted effort estimations. This shows that the magnitude of effect in the prediction power of the models is not high. Thus, the predictions from the models are meaningful in the real world. The performance of the models for each dataset is discussed in the subsections below.

The Bayesian Network is the highly practical significant model on the Albrecht dataset. The ATLM, DNN and CART models had a Cliff's delta effective size of 0.0051 on the Atkinson dataset making them the models with the highest practical significance of the Atkinson datasets. The CART model was the most practical significant model for the Cosmic and Finnish datasets with a Cliff's delta effect size of 0.0007 and 0.0008 respectively. The ATLM, ELasticNet, Bayesian network and CART models had a Cliff's delta effect size of 0.0102 on the Kemerer dataset. The ATLM and CART model were the most practically significant models with Cliff's delta effect size of -0.0044 and 0.0044 respectively. A negative Cliff's delta effect size shows the number of predicted effort greater than the actual effort was higher than the number of actual effort greater than the predicted effort that is $SUM(y_i < \hat{y}_j) >$

$SUM(y_i > \hat{y}_j)$ where y_i is the actual effort and $\hat{y}_j$ is the predicted effort. The Cliff's delta effect size of all the models on each dataset is presented in Table 5.2.

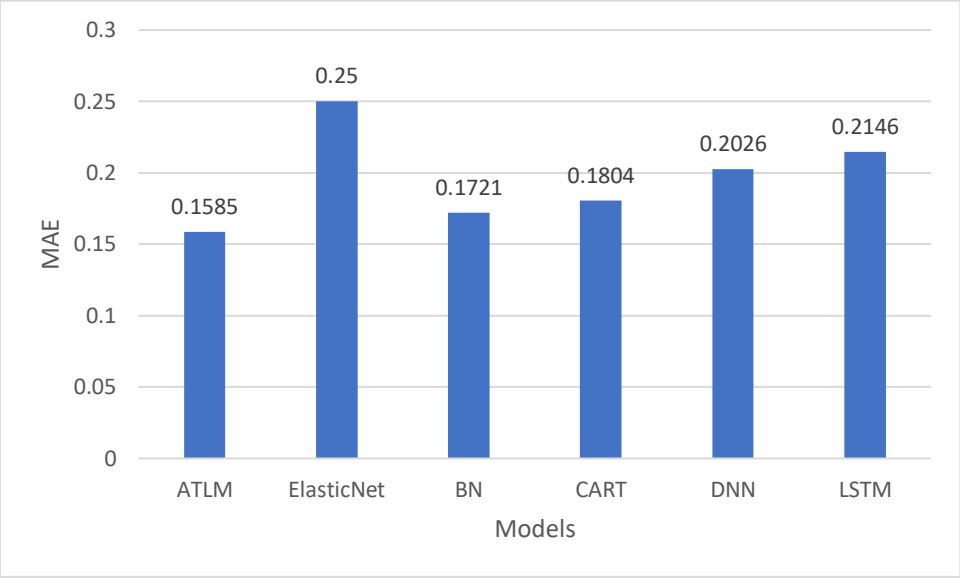


Figure 5.1: Average MAE of models

Table 5.1: Comparison of p-values of the six models

Dataset	ATLM	ElasticNet	BN	CART	DNN	LSTM
Albrecht	0.02	0.0039	0.004	0.0151	0.0045	0.0045
Atkinson	0.0194	0.0033	0.0211	0.0301	0.0629	0.0029
Cosmic	0.0292	0.0405	0.037	0.0483	0.0051	0.0035
Finnish	0.0023	0.0282	0.0047	0.0057	0.0117	0.0074
Kemerer	0.0113	0.0032	0.002	0.0152	0.0016	0.0016
Telecom1	0.0094	0.0034	0.0046	0.0022	0.0481	0.0014

Table 5.2: Comparison of Cliffs δ of the Six Models

Dataset	ATLM	ElasticNet	BN	CART	DNN	LSTM
Albrecht	0.0104	-0.0173	-0.0035	-0.0104	0.0588	0.0588
Atkinson	0.0051	0.0102	0.0102	0.0051	0.0051	-0.0102
Cosmic	0.002	0.002	0.0033	0.0007	0.0256	0.0256
Finnish	0.0017	0.0035	0.0035	0.0008	0.0294	0.0294
Kemerer	0.0102	0.0102	0.0102	0.0102	-0.0204	0.0714
Telecom1	-0.0044	0.0133	0.0222	0.0044	0.0667	0.0667

5.2.1 Analysis of Albrecht Dataset

The DNN model recorded the lowest MAE of 0.387 on the Albrecht dataset. All the conventional machine learning model performed better than the deep learning models on the Albrecht dataset except CART model which failed to outperform the LSTM model. The ATLM is the best performer on the Albrecht data. This is shown in Figure 5.2.

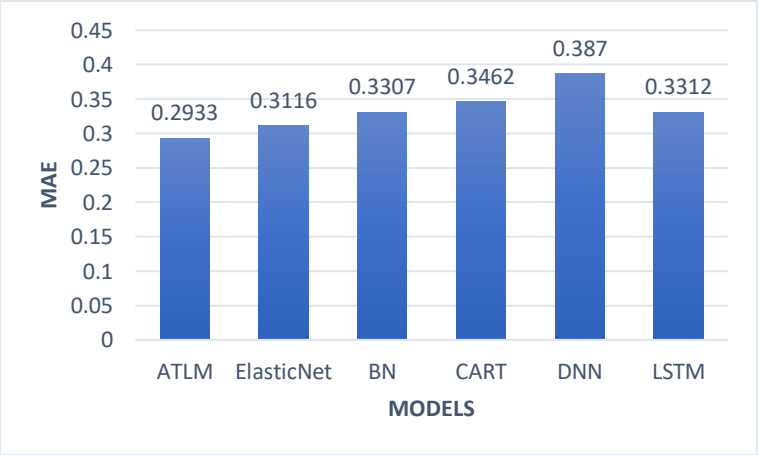


Figure 5.2: MAE measure of Models on Albrecht Dataset

5.2.2 Analysis of Atkinson Dataset

Results of the predictive models on the Atkinson dataset showed that the LSTM model outperformed the other models with an MAE of 0.0297. This is in contrast with the hypothesis that conventional machine learning performance on small-sized data is better than deep learning. However, all the conventional machine learning models outperformed the DNN model. The MAE measure of all the models is shown in the graph in figure **5.3.**

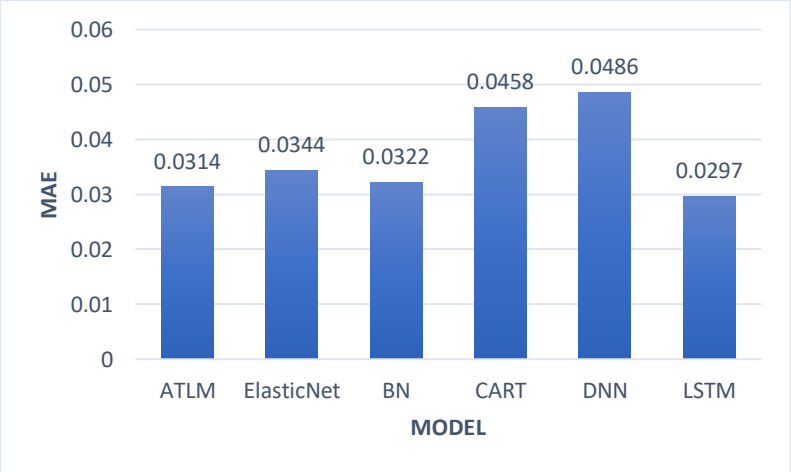


Figure 5.3: MAE measure of Models on Atkinson Dataset

5.2.3 Analysis of Cosmic Dataset

The CART model recorded an MAE of 0.3843 to be the best predictive model on the Cosmic dataset. All the conventional machine learning models outperformed the deep learning models on the Cosmic dataset. An observation on the error measures on all the selected datasets showed that the MAE measure on the Cosmic dataset for all models was high compared to the other datasets. This result is shown in Figure 5.4.

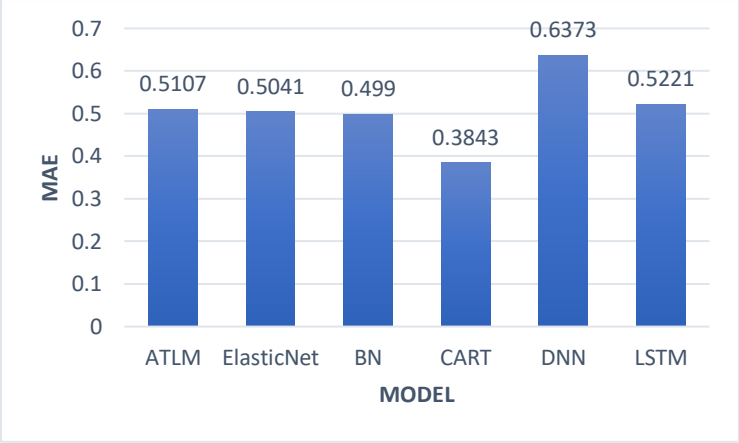


Figure 5.4: MAE measure of Models on Cosmic Dataset

5.2.4 Analysis of Finnish Dataset

There was a high imbalance in the recorded MAE for the models on the Finnish dataset. The ElasticNet and CART models recorded relatively high MAE (0.4857 and 0.0929 respectively) values compared to the ATLM, BN, DNN and LSTM models. This imbalance in the MAE's affected the average MAE of the models across all datasets. This is the reason the deep learning models (DNN and LSTM) were able to outperform the ElasticNet model on the average MAE in section **5.1**. The graph showing the result of models on the Finnish dataset is shown in Figure 5.5.

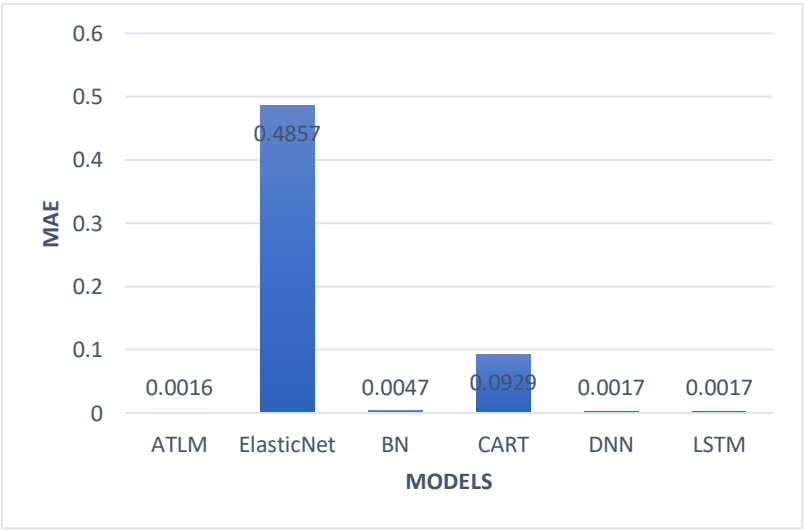


Figure 5.5: MAE measure of Models on Finnish Dataset

5.2.5 Analysis of Kemerer Dataset

The Bayesian Network (BN) recorded the lowest MAE of 0.0536 on the Kemerer dataset. The DNN model showed accurate predictive performance by producing an MAE of 0.0615 which lower than that of the ATLM, CART and LSTM models (0.08, 0.1596 and 0.2487 respectively) as shown in Figure 5.6. The ElasticNet model also achieved accurate prediction performance with an MAE score of 0.055.

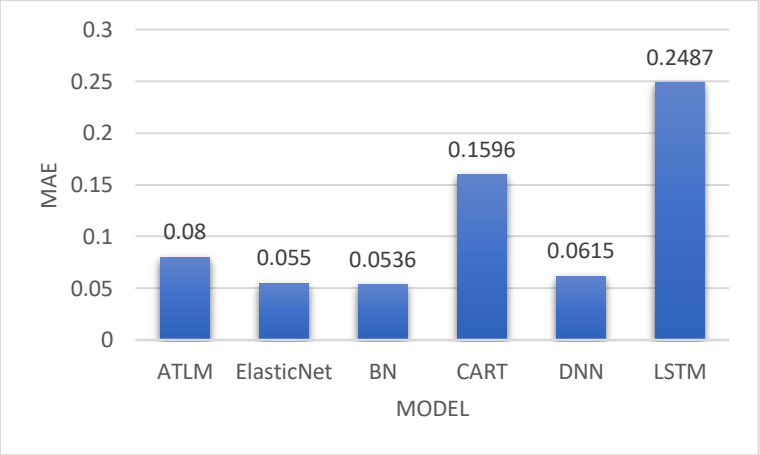


Figure 5.6: MAE measure of Models on Kemerer Dataset

5.2.6 Analysis of Telecom1 Dataset

Two independent variables (CHANGES, FILES) of the Telecom1 were used in setting up the prediction model. The baseline ATLM produced an MAE score of 0.0339 on Telecom1 dataset. The CART and DNN models also showed impressive prediction accuracy with MAE of 0.0538 and 0.0797 respectively as shown in Figure 5.7. The LSTM model performed poorly compared to the other models with an MAE as 0.1542. The ElasticNet and BN models recorded MAE score of 0.1091 and 0.1122 respectively.

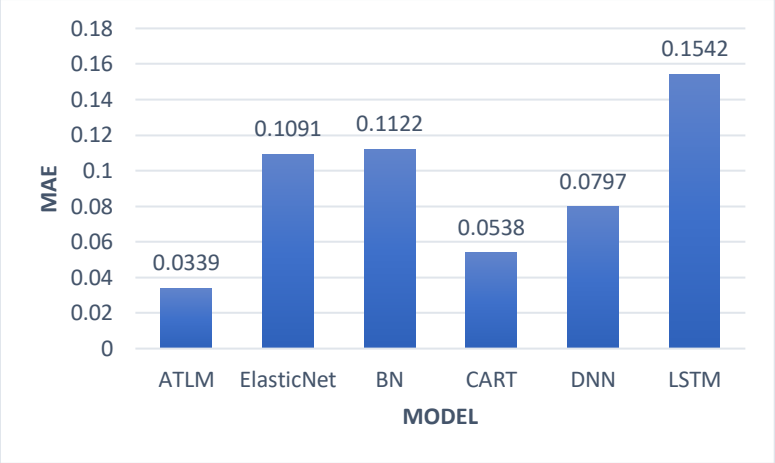


Figure 5.7: MAE measure of Models on Telecom1 Dataset

5.3 Improving the Performance of Deep Learning Models

The results from the initial experiment underscored the need to improve the performance of the deep learning models. Early stopping regularisation technique was added to the remodelling of the deep learning models to improve results. Results after remodelling showed an improvement in the average performance of the deep learning models. The average measure of the mean absolute errors (MAE) of the deep neural network reduced from 0.2026 to 0.1604 and that of LSTM reduced from 0.2146 to 0.1771. Thus, after early stopping, the average MAE measure of the DNN was better than the other three conventional machine learning models but failed to outperform the benchmarked ATLM. The average MAE measure of the LSTM model after early stopping was better than ElasticNet and CART models. However, it failed to outperform the average MAEs of the BN and ATLM. These results are presented in Figure 5.8.

The goal of applying early stopping is to improve the performance of the deep learning models. However, after early stopping, the performance of the LSTM model on the Atkinson and Finnish reduced from 0.0297 to 0.0375 and 0.0017 to 0.0135 respectively. The performance of the DNN model improved on all the datasets after applying early stopping as expected.

The statistical and practical significance of the estimation models were assessed with the p-values and Cliff's δ effect size respectively. The p-values of both the DNN and LSTM models on each dataset were within the acceptable 0.05 asymptotic significance level. On assessing the practical significance of the model, Cliff's δ effect size of each model on each of the datasets was negligible as was defined by Kampenes et al. (2007). The p-values and Cliff's δ effect size of each model on the selected datasets are presented in tables 5.3 and 5.4 respectively.

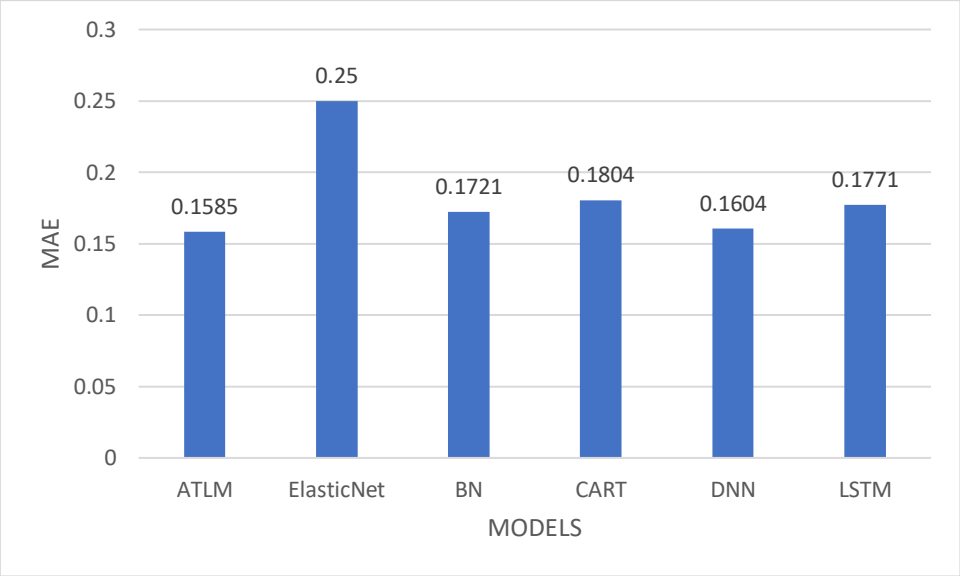


Figure 5.8: Average MAE of models after early stopping

Table 5.3: Comparison of p-values of the six models after early stopping was applied to the deep learning models

Dataset	ATLM	ElasticNet	BN	CART	DNN	LSTM
Albrecht	0.02	0.0039	0.004	0.0151	0.0308	0.0045
Atkinson	0.0194	0.0033	0.0211	0.0301	0.0029	0.0029
Cosmic	0.0292	0.0405	0.037	0.0483	0.0148	0.0389
Finnish	0.0023	0.0282	0.0047	0.0057	0.0082	0.0041
Kemerer	0.0113	0.0032	0.002	0.0152	0.0215	0.0016
Telecom1	0.0094	0.0034	0.0046	0.0022	0.0014	0.0115

Table 5.4: Comparison of Cliffs δ of the six models after early stopping is applied to the deep learning models

Dataset	ATLM	ElasticNet	BN	CART	DNN	LSTM
Albrecht	0.0104	-0.0173	-0.0035	-0.0104	0.0588	0.0588
Atkinson	0.0051	0.0102	0.0102	0.0051	0.0051	0.0051
Cosmic	0.002	0.002	0.0033	0.0007	-0.0007	0.0256
Finnish	0.0017	0.0035	0.0035	0.0008	0.0052	0.0035
Kemerer	0.0102	0.0102	0.0102	0.0102	0.0714	0.0714
Telecom1	-0.0044	0.0133	0.0222	0.0044	0.0667	0.0044

5.3.1 Analysis of Albrecht Dataset

The use of early stopping improved the prediction accuracy of the deep learning models on the Albrecht dataset. The performance of the LSTM model increased from an MAE score of 0.3312 to 0.2895. This made the LSTM the best predictive model on the Albrecht dataset after using early stopping. The prediction performance of the DNN also improved from an MAE score of 0.387 to 0.344 as shown in Figure 5.9. However, it only outperformed the CART model after using early stopping.

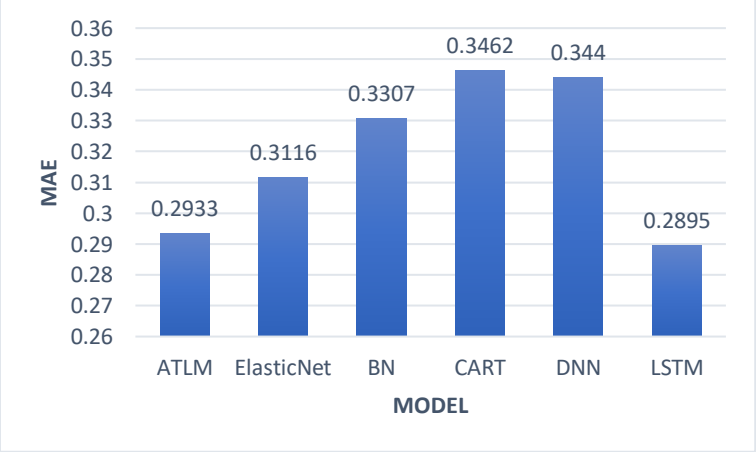


Figure 5.9: MAE measure of models on Albrecht dataset after early stopping

5.3.2 Analysis of Atkinson Dataset

Contrary to expectation, the predictive accuracy of the LSTM model reduced with an MAE score of 0.0297 to 0.0375 after using early stopping. This resulted in the ATLM, ElasticNet and BN models producing better prediction accuracy than the LSTM. Nonetheless, that of the DNN improved slightly with an MAE score of 0.0486 to 0.0447. Thus, the DNN outperformed the CART but failed to outperform the other models. The graph showing this performance is shown in Figure 5.10.

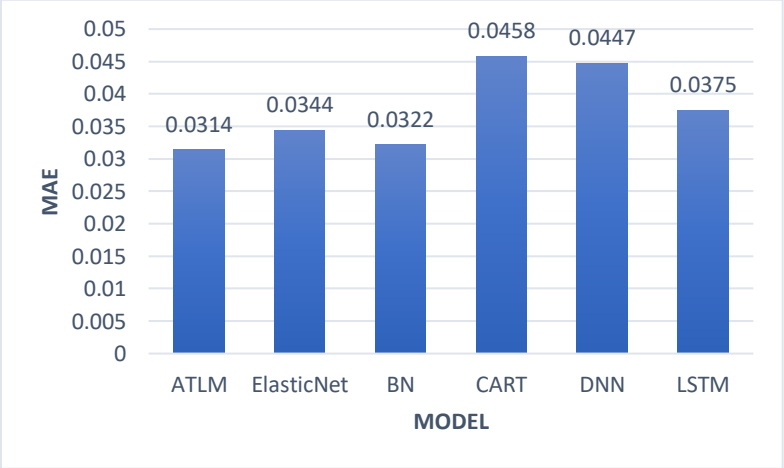


Figure 5.10: MAE measure of models on Atkinson dataset after early stopping

5.3.3 Analysis of Cosmic Dataset

Results on the Cosmic dataset after early stopping showed that the performance of the deep learning models (DNN and LSTM) improved significantly. The MAE score of the DNN reduced from 0.6373 to 0.4373 and that of the LSTM reduced from 0.5221 to 0.4074. Thus, the deep learning models produced better performance than the conventional machine learning models except for the CART model on the Cosmic dataset. This result is shown in the graph in Figure 5.11.

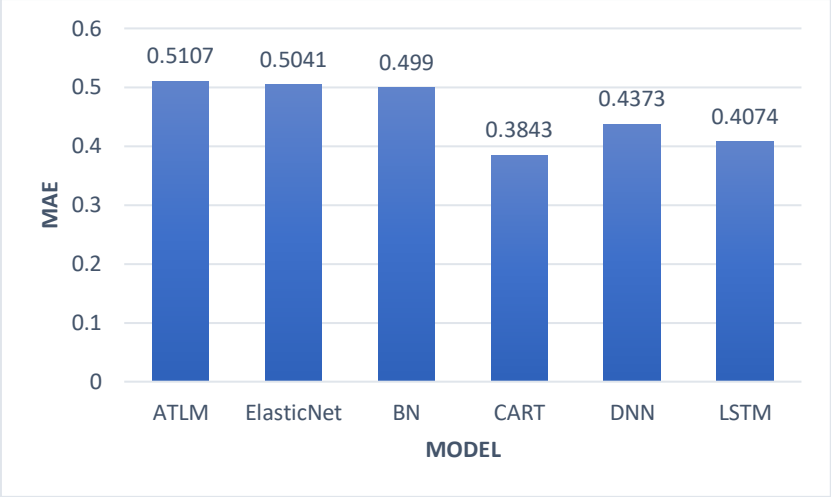


Figure 5.11: MAE measure of models on Cosmic dataset after early stopping

5.3.4 Analysis of Finnish Dataset

The performance of the DNN model improved after early stopping with the MAE score of 0.0014. Consequently, the DNN outperforming all the models. The performance of the LSTM model reduced after using early stopping with MAE score from 0.0017 to 0.0135. Therefore, the BN model outperformed the LSTM model after early stopping as shown in Figure 5.12.

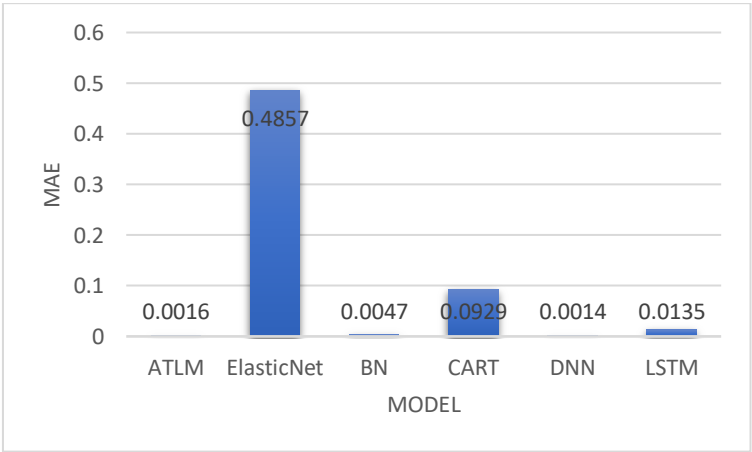


Figure 5.12: MAE measure of models on Finnish dataset after early stopping

5.3.5 Analysis of Kemerer Dataset

A slight improvement was recorded in the performance of the deep learning models after early stopping. The MAE score of the DNN model improved from 0.0615 to 0.0566 and that of the

LSTM from 0.2487 to 0.2316. Although the performance of the LSTM model improved, it failed to outperform any of the models. The DNN model also failed to outperform the ElasticNet and the BN models. This is shown in Figure 5.13.

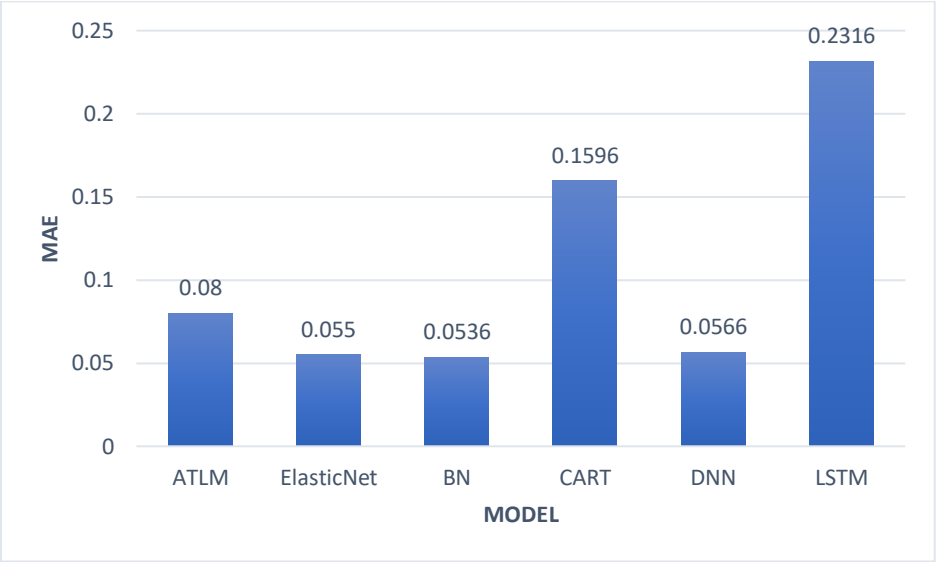


Figure 5.13: MAE measure of models on Kemerer dataset after early stopping

5.3.6 Analysis of Telecom1 Dataset

The prediction accuracy of the DNN model increased after using early stopping. The MAE score improved from 0.0797 to 0.0782. However, the DNN model did not outperform the ATLM and CART models. The MAE score of the LSTM model improved from 0.1542 to 0.083. As a result, the prediction accuracy of the LSTM model beat that of the ElasticNet and BN models but could not beat the prediction accuracy of the ATLM and CART model. This result is shown in the graph in Figure 5.14.

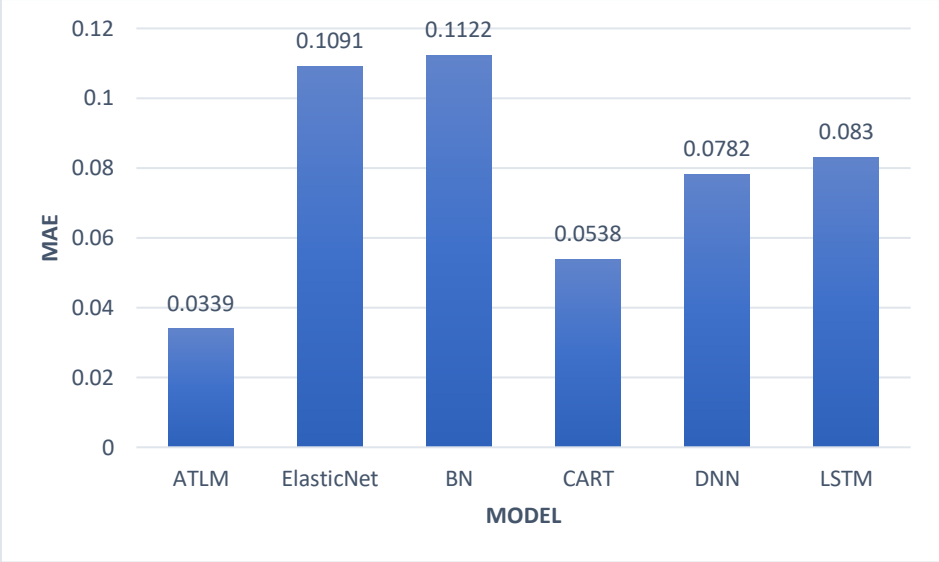


Figure 5.14: MAE measure of models on Telecom1 dataset after early stopping

5.4 Chapter Summary

The chapter presented the results of the experiments. It included an analysis of the average prediction accuracy of the models on all datasets and analysis of prediction accuracy of the models on each dataset. There were instances where the results corroborate findings of similar studies in other domains and instances where it contradicts findings of similar studies in other domains. The next chapter presents a discussion of the results outlining the implications to researcher and practitioners.

CHAPTER SIX

6 DISCUSSION

6.1 Overview

The preceding chapter presented the results of the study. This chapter is a discussion of the results. The results of the discretization scheme defined for a small-sized dataset in section 3.6 is first discussed. This is followed by a discussion of the results in chapter 5.

6.2 Definition of Small-Sized SEE Dataset

A discretisation scheme for classifying the software effort estimation datasets was defined in chapter 3 of the study. The discretisation scheme defined a maximum threshold of 43 project instances for describing a given SEE dataset as small-sized. Thus, a given SEE dataset is small-sized if the number of project instances in the dataset is less than 44.

Based on the evidence from the central limit theory, a sample size is statistically described as small-sized when the number of samples in less than 30. To the best of our knowledge, this definition for a small-sized has not been adopted in software effort estimation studies. Studies by (Jing et al., 2016; Khatibi Bardsiri et al., 2014; Qi et al., 2017; Song et al., 2019; Song et al., 2005) described SEE datasets as small-sized. However, the datasets used in these studies had project instances greater 30. Thus, any assumption that SEE studies adopt this statistical definition of a small-size is false. Also, the defined threshold of small-size in this study is higher than the statistical definition of a small-size sample. The difference will allow more SEE datasets to be classified as small-size. Currently, there are five SEE datasets (Albrecht – 24 instances, Atkinson – 16 instances, Sdr – 12 instances, Kemerer – 15 instances and Telecom1 – 18 instances) in the public domain that has project instances less than 30. The difference in the definition allowed two datasets (Finnish – 38 instances and Cosmic – 42 instances) to be

classified as small-sized. This increase presents SEE researchers a range of datasets to use when considering empirical studies on small-sized datasets.

Notwithstanding the lack of agreement between the definition from this study and the evidence from the central limit theorem, the definition from the study compares well with the claim by Wason (2018). Wason (2018) claimed the optimum number of instance for building an accurate machine learning model should be 10 times the number of features. Thus, given a dataset with N number of features, then the optimum number of project instances should be $10N$. This implies that any number of instances less than $10N$ can be described as small-size. All the datasets classified as small-sized in this study had their number of project instances less than 10x the number of features. This is presented in table 4.1.

6.3 Performance of Models

The results from the comparative study showed that either ATLM, LSTM, CART or BN produced the best prediction accuracy on at least one dataset. This implies that there was no single model that produced the best prediction accuracy on all the selected datasets. This, a confirmation of the no free lunch theorem (Wolpert, 1996) that "*there is no one model that works best for every problem*". Also, the performance of the models across the datasets was affected by a number of factors. This included the features selected for building the model, the unit of effort measurement of the dataset (hours/months) and the number of project instances in the dataset. From this observation, this study suggests using multiple learners when estimating project effort and selecting the best performing learning for your dataset.

Again, closely examining the results indicated that there were few instances where the deep learning models outperform some of the conventional machine learning models. That is on the Atkinson and Finnish datasets where the LSTM and DNN models outperformed the conventional machine learning models. This contradicts the claim that conventional machine

learning models have better prediction accuracy than deep learning models on small-sized datasets. Nonetheless, the average performance showed that conventional machine learning models performed better than deep learning models on the small-sized datasets. Thus, the average performances confirm this claim as reported by Feng et al. (2019). Accordingly, this study suggests that researchers should use multiple datasets when performing a comparative study of predictive models. This is because there could be discrepancies in the results on models on a single dataset and not reflect the prediction accuracy of the models. Using multiple datasets will reduce this error and observed results across the datasets will be a reflection of the prediction accuracy of the models.

Evidence from this study shows that deep learning approaches can perform relatively better than conventional machine learning approaches when early stopping regularisation techniques are used. Nonetheless, the two deep learning approaches used in the study (DNN and LSTM) both failed to outperform the baseline ATLM. As a result, this study suggests both conventional machine learning and deep learning methods should be used for effort estimation. Also, Cliff's delta effect size of the models was within the negligible threshold (Kampenes et al., 2007). The small effect size implies that there was not a huge difference in the predicted effort and the actual effort of the deep learning models. This shows that both conventional machine learning and deep learning approaches make meaningful predictions when used for effort estimation.

The long-short term memory (LSTM) proved to be a good predictive model for effort estimation. However, literature using LSTM for effort estimation is mainly in an agile setting (Choetkiertikul et al., 2019). The results showed that the LSTM can be a good predictive model for estimating effort in the traditional project setting. Therefore, this study also recommends using the LSTM for traditional software effort estimation and not on in agile development setting. Despite the suggestion encouraging the use of deep learning for effort estimation, this

study cautions that there are complexities associated with the implementation of deep learning models.

6.4 Impact of Early Stopping

The impact of early stopping on the error measure, statistical and practical significance of the deep learning models was assessed. The prediction accuracy of the DNN model improved for all six datasets. However, the p-values of the DNN model on the Albrecht, Cosmic and Kemerer datasets increased. The effect size of the Albrecht, Atkinson and Telecom1 datasets remained the same but that of Kemerer increased. For the LSTM model, the MAE of the Atkinson and Finnish datasets increased showing poor performance whilst the MAE of the Albrecht, Cosmic, Kemerer and Telecom1 showed improvement in performance. The p-values of the Cosmic and Telecom1 data also increased denoting a decline in the p-value. The Cliff's δ effect size remained the same for Albrecht, Cosmic and Kemerer, improved for the Finnish and Telecom1 datasets but declined for the Atkinson datasets. This is summarised in tables 6.1 and 6.2 where one (1) depicts an improvement in the measure after early stopping, -1 depicts a reduction in the measure after early stopping and zero (0) depicts no change in the measure before and after early stopping. The summary in tables 6.1 and 6.2 showed that the early stopping regularisation technique had a positive impact on models performance. This study recommends early stopping to researchers and practitioners working with smaller datasets for effort estimation and using deep learning methods. There are other techniques like data augmentation and transfer learning that are used to improve performance when modelling deep learning algorithms with smaller datasets. These techniques were not investigated in this study; however, researchers and practitioners can consider them during modelling as they have been proven to be successful techniques in other fields.

Table 6.1: Impact of early stopping on the performance of the Deep Neural Network Model

Dataset	MAE	*p*-value	Cliff's delta
Albrecht	1	-1	0
Atkinson	1	1	0
Cosmic	1	-1	1
Finnish	1	1	1
Kemerer	1	-1	-1
Telecom1	1	1	0

1 denotes an improvement in measure after early stopping
0 denotes no change in the measure after early stopping
-1 denotes a reduction in the measure after early stopping

Table 6.2: Impact of early stopping on the performance of the Long-Short Term Memory Model

Dataset	MAE	*p*-value	Cliff's delta
Albrecht	1	0	0
Atkinson	-1	0	-1
Cosmic	1	-1	0
Finnish	-1	1	1
Kemerer	1	0	0
Telecom1	1	-1	1

6.5 Computational Complexity

One remarkable observation during the model training was the time taken to complete all epochs. This study did not focus on assessing the computational cost associated with running the deep learning models and the conventional machine learning models. Notwithstanding, the

observations on the running time of all models during the study is in line with the claim by Fu & Menzies, (2017) that running deep learning models is computationally expensive. Analysing the difference in the error rates recorded, this study is in agreement with Fu & Menzies, (2017) that it is not worth sacrificing the computational resource for a small change in the accuracy. This claim has also been made by Jain & Kaushal (2018). However, this study makes exceptions to safety-critical systems since a small change in the accuracy can have high practical importance (Kampenes et al., 2007).

6.6 Chapter Summary

In this chapter, the results of the study were discussed. The defined threshold of 43 project instances for describing a given software effort estimation dataset as small-sized was first discussed. The subsequent sections in this chapter discussed the findings of the comparative study and the impact of early stopping on the models.

CHAPTER SEVEN

7 CONCLUSION

7.1 Summary of the Study

Accurately estimating the effort for a project remains a challenge and has attracted research interest over the past decades (Idri et al., 2016; Sharma & Singh, 2018; Wen et al., 2012). Over the years, researchers and practitioners have adopted expert judgement, parametric models and machine learning techniques to estimate the effort of projects (Huang et al., 2015). The machine learning approach to effort estimation has been reported to achieve high accuracies than the non-machine learning approaches. However, building accurate machine learning models rely on large historical project data. (Bardsiri et al., 2014; Bielak, 2000). The size of Software Effort Estimation (SEE) datasets have been described as small (Jing et al., 2016; Khatibi Bardsiri et al., 2014; Qi et al., 2017; Song et al., 2019; Song et al., 2005). Nonetheless, to the best of our knowledge, the criteria for describing these datasets as small-sized has not been defined in previous SEE studies.

The study defined a discretization scheme to set thresholds for describing the size of a given SEE dataset as small, medium or large. The definition of the thresholds was based on Eubank's optimal spacing theory (Eubank, 1981) which uses the density quantile function to generate an optimal asymptotic spacing between the classes. Six datasets, namely Albrecht, Atkinson, Kemerer, Cosmic, Finnish and Telecom1 were classified as small-sized and were considered for the empirical analysis. The six were used for the empirical study.

The empirical study investigated the prospects of deep learning in SEE by comparing the performance of selected conventional machine learning techniques and selected deep learning models to the baseline ATLM on small-sized datasets. The leave-one-out cross-validation

(LOOCV) was applied in this study for the training and validation needs of the selected models. The performance of the models was evaluated using the mean absolute error (MAE).

The initial results showed two conventional machine learning models (BN, CART) with the baseline ATLM outperforming the deep learning models. After applying the early stopping regularization to the deep learning models, the deep learning models showed improved performance and outperformed the conventional machine learning models. Despite the improved performance of the deep learning models, it failed to outperform the baseline ATLM. The results of the study were validated using Yuen's t-test and Cliff's delta effect size. Statistical inference was made at the 5% asymptotic level.

The evidence from this study suggests that deep learning approaches can be used for effort estimation in addition to the conventional machine learning approaches. However, using deep learning methods comes with an additional computational cost. Also, adopting early stopping regularisation in designing deep learning models improved its prediction accuracy. The study recommends the use of different models for different effort datasets. The results showed that no single model has the best prediction accuracy on all the datasets thus informing this recommendation.

7.2 Limitations of the Study

In this dissertation, a systematic literature review was conducted to identify plausible datasets to build the discretization scheme. The results of the review may have been affected by the incompleteness of the search strategy. Studies were selected from three electronic databases: ACM digital library, ScienceDirect and IEEE Explore which are popular databases in the field of computer science. These three databases are not an exhaustive representation of databases containing articles on empirical software effort estimation. Again, the search and selection criteria used for selecting the primary studies was made clear to ensure that the results of the

review were reproducible. However, different researchers may have varying understanding of selection criteria. The difference in understanding can produce variations in the results.

The datasets used in this study have all been used in previous SEE studies. These datasets are made available in its raw format, thus there is a risk that the pre-processing techniques used could affect final model construction. Nonetheless, these data pre-processing techniques have been recommended in previous empirical software engineering studies.

Using a biased validation method to assess the performance of estimation models in any empirical study can be a threat to the validity of the study. Many studies divide datasets into training and validation sets, this means the models are always validated on the same set in each training round. This can result in a biased performance of the model. To avoid this limitation, each dataset was divided into training and testing/validation sets. LOOCV was used on the training set, thus the validation set for each training round changes. The final output estimation model was evaluated on the unseen testing set for the performance of the estimation model.

7.3 Future work

In a future study, further empirical study will be conducted on the large datasets. The study will test the hypothesis that deep learning approaches perform better than conventional machine learning approaches on medium to large-sized datasets. The goal is to measure the significant effect of deep learning on relatively medium to large-sized datasets.

Future studies will also include sampling small-sized data from the large-sized datasets using the Bellwether sampling techniques recommended by Mensah et al. (2018). Deep learning models will then be set up using the sampled small-sized and large-sized data sets and evaluating the prediction performance in each case.

REFERENCES

Akhtar, N. (2013). Perceptual Evolution for Software Project Cost Estimation using Ant Colony System. *International Journal of Computer Applications*, *81*(14), 23–30. https://doi.org/10.5120/14185-2385

Albrecht, A. J., & Gaffney, J. E. (1983). Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Transactions on Software Engineering*, *SE-9*(6), 639–648. https://doi.org/10.1109/TSE.1983.235271

Arcuri, A., & Briand, L. (2014). A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing Verification and Reliability*, *24*(3), 219–250. https://doi.org/10.1002/stvr.1486

Azzeh, M., & Nassif, A. B. (2016). A hybrid model for estimating software project effort from Use Case Points. *Applied Soft Computing Journal*, *49*, 981–989. https://doi.org/10.1016/j.asoc.2016.05.008

Azzeh, M., Neagu, D., & Cowling, P. (2008). Improving analogy software effort estimation using fuzzy feature subset selection algorithm. *Proceedings - International Conference on Software Engineering*, 71–78. https://doi.org/10.1145/1370788.1370805

Banimustafa, A. (2018). Predicting Software Effort Estimation Using Machine Learning Techniques. *2018 8th International Conference on Computer Science and Information Technology, CSIT 2018*, 249–256. https://doi.org/10.1109/CSIT.2018.8486222

Bardsiri, V. K., Jawawi, D. N. A., Hashim, S. Z. M., & Khatibi, E. (2014). A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons. *Empirical Software Engineering*, *19*(4), 857–884. https://doi.org/10.1007/s10664-013-9241-4

Belsley, D. A., Kuh, E., & Welsch, R. E. (2005). Regression Diagnostics: Identifying Influential Data and Sources of Collinearity. In *John Wiley & Sons* (Vol. 571).

Bielak, J. (2000). Improving size estimates using historical data. *IEEE Software*, *17*(6), 27–35. https://doi.org/10.1109/52.895165

Bilbao, I., & Bilbao, J. (2017). Overfitting problem and the over-training in the era of data: Particularly for Artificial Neural Networks. *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, 173–177. https://doi.org/10.1109/INTELCIS.2017.8260032

Bini, S. A. (2018). Artificial Intelligence, Machine Learning, Deep Learning, and Cognitive Computing: What Do These Terms Mean and How Will They Impact Health Care? *Journal of Arthroplasty*, *33*(8), 2358–2361. https://doi.org/10.1016/j.arth.2018.02.067

Boehm, B. W. (2001). Software Engineering Economics. In *Pioneers and Their Contributions to Software Engineering* (pp. 99–150). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-48354-7_5

Bosu, M. F., & Macdonell, S. G. (2019). Experience: Quality benchmarking of datasets used in software effort estimation. *Journal of Data and Information Quality*, *11*(4). https://doi.org/10.1145/3328746

Cavalcante, I. M., Frazzon, E. M., Forcellini, F. A., & Ivanov, D. (2019). A supervised machine learning approach to data-driven simulation of resilient supplier selection in digital manufacturing. *International Journal of Information Management*, *49*, 86–97. https://doi.org/10.1016/j.ijinfomgt.2019.03.004

Chen, Z., Menzies, T., Port, D., & Boehm, B. (2005). Feature subset selection can improve software cost estimation accuracy. *Proceedings of the 2005 Workshop on Predictor*

Models in Software Engineering, PROMISE 2005. https://doi.org/10.1145/1083165.1083171

Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., & Menzies, T. (2019). A Deep Learning Model for Estimating Story Points. *IEEE Transactions on Software Engineering*, *45*(7), 637–656. https://doi.org/10.1109/TSE.2018.2792473

Cliff, N. (1993). Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological Bulletin*, *114*(3), 494–509. https://doi.org/10.1037/0033-2909.114.3.494

Cohen, J. (1988). Statistical Power Analysis for the Behavioral Sciences. In *Statistical Power Analysis for the Behavioral Sciences*. Routledge. https://doi.org/10.4324/9780203771587

Dejaeger, K., Verbeke, W., Martens, D., & Baesens, B. (2012). Data mining techniques for software effort estimation: A comparative study. *IEEE Transactions on Software Engineering*, *38*(2), 375–397. https://doi.org/10.1109/TSE.2011.55

Deng, L., & Platt, J. C. (2014). Ensemble Deep Learning for Speech Recognition. *INTERSPEECH-2014*.

Dragicevic, S., Celar, S., & Turic, M. (2017). Bayesian network model for task effort estimation in agile software development. *Journal of Systems and Software*, *127*, 109–119. https://doi.org/10.1016/j.jss.2017.01.027

Eubank, R. L. (1981). A Density-Quantile Function Approach to Optimal Spacing Selection. *The Annals of Statistics*, *9*(3), 494–500. https://doi.org/10.1214/AOS/1176345454

Feng, S., Zhou, H., & Dong, H. (2019). Using deep neural network with small dataset to predict material defects. *Materials & Design*, *162*, 300–310. https://doi.org/10.1016/J.MATDES.2018.11.060

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for

financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669. https://doi.org/https://doi.org/10.1016/j.ejor.2017.11.054

Foss, T., Stensrud, E., Kitchenham, B., & Myrtveit, I. (2003). A Simulation Study of the Model Evaluation Criterion MMRE. *IEEE Transactions on Software Engineering*, *29*(11), 985–995. https://doi.org/10.1109/TSE.2003.1245300

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1), 1–22. https://doi.org/10.18637/jss.v033.i01

Fu, W., & Menzies, T. (2017). Easy over hard: A case study on deep learning. *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, *Part F1301*, 49–60. https://doi.org/10.1145/3106237.3106256

Fuentetaja, R., Borrajo, D., López, C. L., & Ocón, J. (2013). Multi-step Generation of Bayesian Networks Models for Software Projects Estimations. *International Journal of Computational Intelligence Systems*, *6*(5), 796–821. https://doi.org/10.1080/18756891.2013.805583

Garcia, J., & Barbedo, A. (2018). Impact of dataset size and variety on the e ff ectiveness of deep learning and transfer learning for plant disease classi fi cation. *Computers and Electronics in Agriculture*, *153*(March), 46–53. https://doi.org/10.1016/j.compag.2018.08.013

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, *4*(1), 1–58. https://doi.org/10.1162/neco.1992.4.1.1

Ghiselli, E. E. (1964). Maturity of Self-Perception in Relation to Managerial Success. *Personnel Psychology*, *17*(1), 41–48. https://doi.org/10.1111/j.1744-

6570.1964.tb00049.x

GLASS, G. V. (1976). Primary, Secondary, and Meta-Analysis of Research. *Educational Researcher*, *5*(10), 3–8. https://doi.org/10.3102/0013189x005010003

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. https://www.deeplearningbook.org/

Gregor, S., & Hevner, A. R. (2013). Positioning and Presenting Design Science Research for Maximum Impact. In *Source: MIS Quarterly* (Vol. 37, Issue 2).

Hall, M. (2000). *Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning*. https://researchcommons.waikato.ac.nz/handle/10289/1024

Hara, K., Sun, J., Moore, R., Jacobs, D., & Froehlich, J. E. (2014). Tohme: Detecting curb ramps in Google Street View using crowdsourcing, computer vision, and machine learning. *UIST 2014 - Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, 189–204. https://doi.org/10.1145/2642918.2647403

Harik, G., Cantu-Paz, E., Goldberg, D. E., & Miller, B. L. (1999). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, *7*(3), 231–253. https://doi.org/10.1162/evco.1999.7.3.231

Hawkins, D. M. (2004). The Problem of Overfitting. *Journal of Chemical Information and Computer Sciences*, *44*(1), 1–12. https://doi.org/10.1021/ci0342472

Hedges, L. V. (1981). Distribution Theory for Glass's Estimator of Effect size and Related Estimators. *Journal of Educational Statistics*, *6*(2), 107–128. https://doi.org/10.3102/10769986006002107

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A. R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., & Kingsbury, B. (2012). Deep neural networks for acoustic

modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97. https://doi.org/10.1109/MSP.2012.2205597

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hosni, M., Idri, A., & Abran, A. (2017). Investigating heterogeneous ensembles with filter feature selection for software effort estimation. *ACM International Conference Proceeding Series*, *Part F1319*, 207–220. https://doi.org/10.1145/3143434.3143456

Host, M., & Wohlin, C. (1998). Experimental study of individual subjective effort estimations and combinations of the estimates. *Proceedings - International Conference on Software Engineering*, 332–339. https://doi.org/10.1109/ICSE.1998.671386

Hua, J., Xiong, Z., Lowey, J., Suh, E., & Dougherty, E. R. (2005). Optimal number of features as a function of sample size for various classification rules. *Bioinformatics*, *21*(8), 1509–1515. https://doi.org/10.1093/bioinformatics/bti171

Huang, J., Li, Y. F., & Xie, M. (2015). An empirical analysis of data preprocessing for machine learning-based software cost estimation. *Information and Software Technology*, *67*, 108–127. https://doi.org/10.1016/j.infsof.2015.07.004

Idri, A., Hosni, M., & Abran, A. (2016). Systematic literature review of ensemble effort estimation. *Journal of Systems and Software*, *118*, 151–175. https://doi.org/10.1016/j.jss.2016.05.016

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., & Brox, T. (2017). FlowNet 2.0: Evolution of optical flow estimation with deep networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 1647–1655. https://doi.org/10.1109/CVPR.2017.179

Jain, K., & Kaushal, S. (2018). A Comparative Study of Machine Learning and Deep Learning Techniques for Sentiment Analysis. *2018 7th International Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions, ICRITO 2018*, 483–487. https://doi.org/10.1109/ICRITO.2018.8748793

Jing, X. Y., Qi, F., Wu, F., & Xu, B. (2016). Missing data imputation based on low-rank recovery and semi-supervised regression for software effort estimation. *Proceedings - International Conference on Software Engineering*, *14-22-May-2016*, 607–618. https://doi.org/10.1145/2884781.2884827

Justus, D., Brennan, J., Bonner, S., & Mcgough, A. S. (2019). Predicting the Computational Cost of Deep Learning Models. *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, 3873–3882. https://doi.org/10.1109/BigData.2018.8622396

Kampenes, V. B., Dybå, T., Hannay, J. E., & Sjøberg, D. I. K. (2007). A systematic review of effect size in software engineering experiments. In *Information and Software Technology* (Vol. 49, Issues 11–12, pp. 1073–1086). Elsevier. https://doi.org/10.1016/j.infsof.2007.02.015

Kitchenham, B., & Kansala, K. (1993). Inter-item correlations among function points. *Proceedings - 1st International Software Metrics Symposium, METRIC 1993*, 11–14. https://doi.org/10.1109/METRIC.1993.263805

Kitchenham, B., Madeyski, L., Budgen, D., Keung, J., Brereton, P., Charters, S., Gibbs, S., & Pohthong, A. (2017). Robust Statistical Methods for Empirical Software Engineering. *Empirical Software Engineering*, *22*(2), 579–630. https://doi.org/10.1007/s10664-016-9437-5

Kocaguneli, E., & Menzies, T. (2013). Software effort models should be assessed via leave-one-out validation. *Journal of Systems and Software*, *86*(7), 1879–1890.

https://doi.org/10.1016/j.jss.2013.02.053

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks* (pp. 1097–1105). http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lopez-Martin, C. (2011). A fuzzy logic model for predicting the development effort of short scale programs based upon two independent variables. *Applied Soft Computing Journal*, *11*(1), 724–732. https://doi.org/10.1016/j.asoc.2009.12.034

Louati, F., & Ktata, F. B. (2020). A deep learning-based multi-agent system for intrusion detection. *SN Applied Sciences*, *2*(4), 675. https://doi.org/10.1007/s42452-020-2414-z

Macbeth, G., Razumiejczyk, E., & Ledesma, R. D. (2010). Cliff´s Delta Calculator: A non-parametric effect size program for two groups of observations. *Universitas Psychologica*, *10*(2), 545–555. https://doi.org/10.11144/javeriana.upsy10-2.cdcp

Mair, C., Shepperd, M., & Jørgensen, M. (2005). An analysis of data sets used to train and validate cost prediction systems. *ACM SIGSOFT Software Engineering Notes*, *30*(4), 1–6. https://doi.org/10.1145/1082983.1083166

Marcus, G. (2018). *Deep Learning : A Critical Appraisal*. 1–27.

Mendes, E., & Kitchenham, B. (2004). Further comparison of cross-company and within-company effort estimation models for Web applications. *Proceedings - International Software Metrics Symposium*, 348–357. https://doi.org/10.1109/METRIC.2004.1357920

Mensah, S., Keung, J., Bosu, M. F., & Bennin, K. E. (2018). Duplex output software effort estimation model with self-guided interpretation. *Information and Software Technology*,

94(September 2017), 1–13. https://doi.org/10.1016/j.infsof.2017.09.010

Mensah, S., Keung, J., MacDonell, S. G., Bosu, M. F., & Bennin, K. E. (2018). Investigating the Significance of the Bellwether Effect to Improve Software Effort Prediction: Further Empirical Study. *IEEE Transactions on Reliability*, *67*(3), 1176–1198. https://doi.org/10.1109/TR.2018.2839718

Menzies, T., Yang, Y., Mathew, G., Boehm, B., & Hihn, J. (2016). Negative Results for Software Effort Estimation. *Empirical Software Engineering*, *22*(5), 2658–2683. http://arxiv.org/abs/1609.05563

Minku, L. L., & Yao, X. (2013a). Software effort estimation as a multiobjective learning problem. *ACM Transactions on Software Engineering and Methodology*, *22*(4). https://doi.org/10.1145/2522920.2522928

Minku, L. L., & Yao, X. (2013b). Ensembles and locality: Insight on improving software effort estimation. *Information and Software Technology*, *55*(8), 1512–1528. https://doi.org/10.1016/j.infsof.2012.09.012

Mittas, N., Papatheocharous, E., Angelis, L., & Andreou, A. S. (2015). Integrating non-parametric models with linear components for producing software cost estimations. *Journal of Systems and Software*, *99*, 120–134. https://doi.org/10.1016/j.jss.2014.09.025

Moløkken-Østvold, K., & Jørgensen, M. (2004). Group processes in software effort estimation. *Empirical Software Engineering*, *9*(4), 315–334. https://doi.org/10.1023/B:EMSE.0000039882.39206.5a

Nassif, A. B., Ho, D., & Capretz, L. F. (2013). Towards an early software estimation using log-linear regression and a multilayer perceptron model. *Journal of Systems and Software*, *86*(1), 144–160. https://doi.org/10.1016/j.jss.2012.07.050

Neill, D. B. (2013). Using artificial intelligence to improve hospital inpatient care. *IEEE Intelligent Systems*, *28*(2), 92–95. https://doi.org/10.1109/MIS.2013.51

Ng, H., Nguyen, V. D., Vonikakis, V., & Winkler, S. (2015). *Deep Learning for Emotion Recognition on Small Datasets Using Transfer Learning*. 443–449.

Ning, C., & You, F. (2018). Data-driven stochastic robust optimization: General computational framework and algorithm leveraging machine learning for optimization under uncertainty in the big data era. *Computers and Chemical Engineering*, *111*, 115–133. https://doi.org/10.1016/j.compchemeng.2017.12.015

Pai, D. R., McFall, K. S., & Subramanian, G. H. (2013). Software effort estimation using a neural network ensemble. *Journal of Computer Information Systems*, *53*(4), 49–58. https://doi.org/10.1080/08874417.2013.11645650

Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Pasupa, K., & Sunhem, W. (2016). *A Comparison between Shallow and Deep Architecture Classifiers on Small Dataset*.

Perez, L., & Wang, J. (2017). *The Effectiveness of Data Augmentation in Image Classification using Deep Learning*. http://arxiv.org/abs/1712.04621

Poggio, T., Kawaguchi, K., Liao, Q., Miranda, B., Rosasco, L., Boix, X., Hidary, J., & Mhaskar, H. (2018). *Theory of Deep Learning III : explaining the non-overfitting puzzle*. *073*.

Pospieszny, P., Czarnacka-chrobot, B., & Kobylinski, A. (2018). An effective approach for software project effort and duration estimation with machine learning algorithms. *Journal*

of Systems and Software, *137*, 184–196. https://doi.org/10.1016/j.jss.2017.11.066

Prechelt, L. (1998a). *Early Stopping - But When?* (pp. 55–69). https://doi.org/10.1007/3-540-49430-8_3

Prechelt, L. (1998b). Automatic early stopping using cross validation: Quantifying the criteria. *Neural Networks*, *11*(4), 761–767. https://doi.org/10.1016/S0893-6080(98)00010-0

Qi, F., Jing, X. Y., Zhu, X., Xie, X., Xu, B., & Ying, S. (2017). Software effort estimation based on open source projects: Case study of Github. *Information and Software Technology*, *92*, 145–157. https://doi.org/10.1016/j.infsof.2017.07.015

Rahman, M. T., & Islam, M. M. (2019). A Comparison of Machine Learning Algorithms to Estimate Effort in Varying Sized Software. *Proceedings of 2019 IEEE Region 10 Symposium, TENSYMP 2019*, 137–142. https://doi.org/10.1109/TENSYMP46218.2019.8971150

Raskutti, G., Wainwright, M. J., & Yu, B. (2014). Early Stopping and Non-parametric Regression: An Optimal Data-dependent Stopping Rule. In *Journal of Machine Learning Research* (Vol. 15).

Rijwani, P., & Jain, S. (2016). Enhanced Software Effort Estimation Using Multi Layered Feed Forward Artificial Neural Network Technique. *Procedia Computer Science*, *89*, 307–312. https://doi.org/10.1016/j.procs.2016.06.073

Rush, C., & Roy, R. (2001). Expert judgement in cost estimating: Modelling the reasoning process. *Concurrent Engineering Research and Applications*, *9*(4), 271–285. https://doi.org/10.1177/1063293x0100900404

Sarro, F., & Petrozziello, A. (2018). Linear programming as a baseline for software effort estimation. *ACM Transactions on Software Engineering and Methodology*, *27*(3).

https://doi.org/10.1145/3234940

Satapathy, S. M., & Rath, S. K. (2017). Empirical assessment of machine learning models for effort estimation of web-based applications. *ACM International Conference Proceeding Series*, 74–84. https://doi.org/10.1145/3021460.3021468

Sawada, Y., & Kozuka, K. (2015). *Transfer Learning Method using Multi-Prediction Deep Boltzmann Machines for a small scale dataset*. 6–9.

Seo, Y. S., & Bae, D. H. (2013). On the value of outlier elimination on software effort estimation research. *Empirical Software Engineering*, *18*(4), 659–698. https://doi.org/10.1007/s10664-012-9207-y

Sharma, P., & Singh, J. (2018). Systematic literature review on software effort estimation using machine learning approaches. *Proceedings - 2017 International Conference on Next Generation Computing and Information Systems, ICNGCIS 2017*, 54–57. https://doi.org/10.1109/ICNGCIS.2017.33

Shepperd, M., & MacDonell, S. (2012). Evaluating prediction systems in software project estimation. *Information and Software Technology*, *54*(8), 820–827. https://doi.org/10.1016/j.infsof.2011.12.008

Song, L., Minku, L. L., & Xin, Y. A. O. (2019). Software effort interval prediction via Bayesian inference and synthetic bootstrap resampling. *ACM Transactions on Software Engineering and Methodology*, *28*(1). https://doi.org/10.1145/3295700

Song, L., Minku, L. L., & Yao, X. (2018). A novel automated approach for software effort estimation based on data augmentation. *ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 468–479.

https://doi.org/10.1145/3236024.3236052

Song, Q., Shepperd, M., & Mair, C. (2005). Using grey relational analysis to predict software effort with small data sets. *Proceedings - International Software Metrics Symposium*, *2005*, 321–330. https://doi.org/10.1109/METRICS.2005.51

Souza, R., Lucena, O., Garrafa, J., Gobbi, D., Saluzzi, M., Appenzeller, S., Rittner, L., Frayne, R., & Lotufo, R. (2018). NeuroImage An open , multi-vendor , multi- fi eld-strength brain MR dataset and analysis of publicly available skull stripping methods agreement. *NeuroImage*, *170*, 482–494. https://doi.org/10.1016/j.neuroimage.2017.08.021

Tong, D. L., & Mintram, R. (2010). Genetic Algorithm-Neural Network (GANN): A study of neural network activation functions and depth of genetic algorithm search applied to feature selection. *International Journal of Machine Learning and Cybernetics*, *1*(1–4), 75–87. https://doi.org/10.1007/s13042-010-0004-x

Tsoukalas, A., Albertson, T., & Tagkopoulos, I. (2015). From data to optimal decision making: a data-driven, probabilistic machine learning approach to decision support for patients with sepsis. *JMIR Medical Informatics*, *3*(1), e11. https://doi.org/10.2196/medinform.3445

Tukey, J. (1977). *Exploratory data analysis*. http://theta.edu.pl/wp-content/uploads/2012/10/exploratorydataanalysis_tukey.pdf

Vinay Kumar, K., Ravi, V., Carr, M., & Raj Kiran, N. (2008). Software development cost estimation using wavelet neural networks. *Journal of Systems and Software*, *81*(11), 1853–1867. https://doi.org/10.1016/j.jss.2007.12.793

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, *2018*.

https://doi.org/10.1155/2018/7068349

Wang, S., Li, B., Liu, Y., Yan, W., Ou, X., Huang, X., Xu, F., & Fu, X. (2018). Neurocomputing Micro-expression recognition with small sample size by transferring long-term convolutional neural network. *Neurocomputing*, *312*, 251–262. https://doi.org/10.1016/j.neucom.2018.05.107

Wason, R. (2018). Deep Learning: Evolution and Expansion. *Cognitive Systems Research*, *52*, 701–708. https://doi.org/10.1016/j.cogsys.2018.08.023

Welch, B. L. (1947). The Generalization of 'Student's' Problem when Several Different Population Variances are Involved. *Biometrika*, *34*(1–2), 28–35. https://doi.org/10.1093/biomet/34.1-2.28

Wen, J., Li, S., Lin, Z., Hu, Y., & Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, *54*(1), 41–59. https://doi.org/10.1016/j.infsof.2011.09.002

Whigham, P. A., Owen, C. A., & MacDonell, S. G. (2015). A baseline model for software effort estimation. *ACM Transactions on Software Engineering and Methodology*, *24*(3). https://doi.org/10.1145/2738037

Wilcox, R. R. (2012). Introduction to Robust Estimation and Hypothesis Testing. In *Introduction to Robust Estimation and Hypothesis Testing*. Elsevier Inc. https://doi.org/10.1016/C2010-0-67044-1

Wolpert, D. H. (1996). The Lack of a Priori Distinctions between Learning Algorithms. *Neural Computation*, *8*(7), 1341–1390. https://doi.org/10.1162/neco.1996.8.7.1341

Wong, K. C. L., Syeda-mahmood, T., & Moradi, M. (2018). Building medical image classifiers with very limited data using segmentation networks. *Medical Image Analysis*, *49*, 105–

116. https://doi.org/10.1016/j.media.2018.07.010

Xia, T., Krishna, R., Chen, J., Mathew, G., Shen, X., & Menzies, T. (2018). *Hyperparameter Optimization for Effort Estimation*. https://github.com/arennax/effort

Xu, H., Huang, C., & Wang, D. (2018). Knowledge-Based Systems Enhancing semantic image retrieval with limited labeled examples via deep learning. *Knowledge-Based Systems*. https://doi.org/10.1016/j.knosys.2018.08.032

Xue-Wen Chen, & Xiaotong Lin. (2014). Big Data Deep Learning: Challenges and Perspectives. *IEEE Access*, *2*, 514–525. https://doi.org/10.1109/ACCESS.2014.2325029

Yang, X., Lo, D., Xia, X., Zhang, Y., & Sun, J. (2015). Deep Learning for Just-in-Time Defect Prediction. *Proceedings - 2015 IEEE International Conference on Software Quality, Reliability and Security, QRS 2015*, 17–26. https://doi.org/10.1109/QRS.2015.14

Yuen, K. K. (1974). The two-sample trimmed t for unequal population variances. *Biometrika*, *61*(1), 165–170. https://doi.org/10.1093/biomet/61.1.165

Zander, S., Nguyen, T., & Armitage, G. (2005). Automated traffic classification and application identification using machine learning. *Proceedings - Conference on Local Computer Networks, LCN*, *2005*, 250–257. https://doi.org/10.1109/LCN.2005.35

Zare, F., Zare, H. K., Fallahnezhad, M. S., Khademi Zare, H., & Fallahnezhad, M. S. (2016). Software effort estimation based on the optimal Bayesian belief network. *Applied Soft Computing Journal*, *49*, 968–980. https://doi.org/10.1016/j.asoc.2016.08.004

Zhang, D., & Tsai, J. J. P. (2005). *Machine Learning Applications in Software Engineering* (Vol. 16). WORLD SCIENTIFIC. https://doi.org/10.1142/5700

Zhang, W., Yang, Y., & Wang, Q. (2015). Using Bayesian regression and EM algorithm with missing handling for software effort prediction. *Information and Software Technology*,

58, 58–70. https://doi.org/10.1016/j.infsof.2014.10.005

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, *115*, 213–237. https://doi.org/10.1016/j.ymssp.2018.05.050

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(2), 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x

APPENDIX

SELECTED PRIMARY STUDIES FOR THE SYSTEMATIC LITERATURE REVIEW

ID	REFERENCE
P1	Kocaguneli, E., Menzies, T., Keung, J., Cok, D., & Madachy, R. (2013). Active Learning and effort estimation: Finding the essential content of software effort estimation data. *IEEE Transactions on Software Engineering*, *39*(8), 1040–1053. https://doi.org/10.1109/TSE.2012.88
P2	Azzeh, M., & Nassif, A. B. (2015). Analogy-based effort estimation: A new method to discover set of analogies from dataset characteristics. *IET Software*, *9*(2), 39–50. https://doi.org/10.1049/iet-sen.2013.0165
P3	Bosu, M. F., & Macdonell, S. G. (2019). Experience: Quality benchmarking of datasets used in software effort estimation. *Journal of Data and Information Quality*, *11*(4). https://doi.org/10.1145/3328746
P4	Dejaeger, K., Verbeke, W., Martens, D., & Baesens, B. (2012). Data mining techniques for software effort estimation: A comparative study. *IEEE Transactions on Software Engineering*, *38*(2), 375–397. https://doi.org/10.1109/TSE.2011.55
P5	Kocaguneli, E., Menzies, T., Bener, A. B., & Keung, J. W. (2012). Exploiting the essential assumptions of analogy-based effort estimation. *IEEE Transactions on Software Engineering*, *38*(2), 425–438. https://doi.org/10.1109/TSE.2011.27
P6	Mensah, S., Keung, J., MacDonell, S. G., Bosu, M. F., & Bennin, K. E. (2018). Investigating the Significance of the Bellwether Effect to Improve Software

	Effort Prediction: Further Empirical Study. *IEEE Transactions on Reliability*, *67*(3), 1176–1198. https://doi.org/10.1109/TR.2018.2839718
P7	Kocaguneli, E., Menzies, T., & Keung, J. W. (2012). On the value of ensemble effort estimation. *IEEE Transactions on Software Engineering*, *38*(6), 1403–1416. https://doi.org/10.1109/TSE.2011.111
P8	Mittas, N., & Angelis, L. (2013). Ranking and clustering software cost estimation models through a multiple comparisons algorithm. *IEEE Transactions on Software Engineering*, *39*(4), 537–551. https://doi.org/10.1109/TSE.2012.45
P9	Azzeh, M., Neagu, D., & Cowling, P. I. (2011). Analogy-based software effort estimation using Fuzzy numbers. *Journal of Systems and Software*, *84*(2), 270–284. https://doi.org/10.1016/j.jss.2010.09.028
P10	Song, L., Minku, L. L., & Xin, Y. A. O. (2019). Software effort interval prediction via Bayesian inference and synthetic bootstrap resampling. *ACM Transactions on Software Engineering and Methodology*, *28*(1). https://doi.org/10.1145/3295700
P11	Azzeh, M., Nassif, A. B., & Minku, L. L. (2015). An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation. *Journal of Systems and Software*, *103*, 36–52. https://doi.org/10.1016/j.jss.2015.01.028
P12	Sarro, F., & Petrozziello, A. (2018). Linear programming as a baseline for software effort estimation. *ACM Transactions on Software Engineering and Methodology*, *27*(3). https://doi.org/10.1145/3234940
P13	Araújo, R. D. A., Oliveira, A. L. I., & Soares, S. (2011). A shift-invariant morphological system for software development cost estimation. *Expert*

	Systems with Applications, *38*(4), 4162–4168. https://doi.org/10.1016/j.eswa.2010.09.078
P14	Khalifelu, Z. A., & Gharehchopogh, F. S. (2012). Comparison and evaluation of data mining techniques with algorithmic models in software cost estimation. *Procedia Technology*, *1*, 65–71. https://doi.org/10.1016/j.protcy.2012.02.013
P15	Moeyersoms, J., Junqué De Fortuny, E., Dejaeger, K., Baesens, B., & Martens, D. (2015). Comprehensible software fault and effort prediction: A data mining approach. *Journal of Systems and Software*, *100*, 80–90. https://doi.org/10.1016/j.jss.2014.10.032
P16	Roh, S. B., Oh, S. K., & Pedrycz, W. (2011). Design of fuzzy radial basis function-based polynomial neural networks. *Fuzzy Sets and Systems*, *185*(1), 15–37. https://doi.org/10.1016/j.fss.2011.06.014
P17	Mensah, S., Keung, J., Bosu, M. F., & Bennin, K. E. (2018). Duplex output software effort estimation model with self-guided interpretation. *Information and Software Technology*, *94*(September 2017), 1–13. https://doi.org/10.1016/j.infsof.2017.09.010
P18	Rijwani, P., & Jain, S. (2016). Enhanced Software Effort Estimation Using Multi Layered Feed Forward Artificial Neural Network Technique. *Procedia Computer Science*, *89*, 307–312. https://doi.org/10.1016/j.procs.2016.06.073
P19	Pendharkar, P. C. (2015). Ensemble based point and confidence interval forecasting in software engineering. *Expert Systems with Applications*, *42*(24), 9441–9448. https://doi.org/10.1016/j.eswa.2015.08.002
P20	Minku, L. L., & Yao, X. (2013b). Ensembles and locality: Insight on improving

	software effort estimation. *Information and Software Technology*, *55*(8), 1512–1528. https://doi.org/10.1016/j.infsof.2012.09.012
P21	Dolado, J. J., Rodriguez, D., Harman, M., Langdon, W. B., & Sarro, F. (2016). Evaluation of estimation models using the Minimum Interval of Equivalence. *Applied Soft Computing Journal*, *49*, 956–967. https://doi.org/10.1016/j.asoc.2016.03.026
P22	Whigham, P. A., Owen, C. A., & MacDonell, S. G. (2015). A baseline model for software effort estimation. *ACM Transactions on Software Engineering and Methodology*, *24*(3). https://doi.org/10.1145/2738037
P23	Oliveira, A. L. I., Braga, P. L., Lima, R. M. F., & Cornélio, M. L. (2010). GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. *Information and Software Technology*, *52*(11), 1155–1166. https://doi.org/10.1016/j.infsof.2010.05.009
P24	Idri, A., Hosni, M., & Abran, A. (2016). Improved estimation of software development effort using Classical and Fuzzy Analogy ensembles. *Applied Soft Computing Journal*, *49*, 990–1019. https://doi.org/10.1016/j.asoc.2016.08.012
P25	Sree, P. R., & Ramesh, S. N. S. V. S. C. (2016). Improving Efficiency of Fuzzy Models for Effort Estimation by Cascading & Clustering Techniques. *Procedia Computer Science*, *85*(Cms), 278–285. https://doi.org/10.1016/j.procs.2016.05.234
P26	Mittas, N., Papatheocharous, E., Angelis, L., & Andreou, A. S. (2015). Integrating non-parametric models with linear components for producing software cost

	estimations. *Journal of Systems and Software*, *99*, 120–134. https://doi.org/10.1016/j.jss.2014.09.025
P27	Lokan, C., & Mendes, E. (2014). Investigating the use of duration-based moving windows to improve software effort prediction: A replicated study. *Information and Software Technology*, *56*(9), 1063–1075. https://doi.org/10.1016/j.infsof.2014.02.008
P28	Abdelali, Z., Mustapha, H., & Abdelwahed, N. (2019). Investigating the use of random forest in software effort estimation. *Procedia Computer Science*, *148*, 343–352. https://doi.org/10.1016/j.procs.2019.01.042
P29	Bardsiri, V. K., Jawawi, D. N. A., Bardsiri, A. K., & Khatibi, E. (2013). LMES: A localized multi-estimator model to estimate software development effort. *Engineering Applications of Artificial Intelligence*, *26*(10), 2624–2640. https://doi.org/10.1016/j.engappai.2013.08.005
P30	Minku, L. L., & Yao, X. (2013a). Software effort estimation as a multiobjective learning problem. *ACM Transactions on Software Engineering and Methodology*, *22*(4). https://doi.org/10.1145/2522920.2522928
P31	Pendharkar, P. C. (2010). Probabilistic estimation of software size and effort. *Expert Systems with Applications*, *37*(6), 4435–4440. https://doi.org/10.1016/j.eswa.2009.11.085
P32	Samareh Moosavi, S. H., & Khatibi Bardsiri, V. (2017). Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation. *Engineering Applications of Artificial Intelligence*, *60*(May 2016), 1–15. https://doi.org/10.1016/j.engappai.2017.01.006

P33	Qi, F., Jing, X. Y., Zhu, X., Xie, X., Xu, B., & Ying, S. (2017). Software effort estimation based on open source projects: Case study of Github. *Information and Software Technology*, *92*, 145–157. https://doi.org/10.1016/j.infsof.2017.07.015
P34	Zare, F., Zare, H. K., Fallahnezhad, M. S., Khademi Zare, H., & Fallahnezhad, M. S. (2016). Software effort estimation based on the optimal Bayesian belief network. *Applied Soft Computing Journal*, *49*, 968–980. https://doi.org/10.1016/j.asoc.2016.08.004
P35	Kocaguneli, E., & Menzies, T. (2013). Software effort models should be assessed via leave-one-out validation. *Journal of Systems and Software*, *86*(7), 1879–1890. https://doi.org/10.1016/j.jss.2013.02.053
P36	López-Martín, C., & Abran, A. (2015). Neural networks for predicting the duration of new software projects. *Journal of Systems and Software*, *101*, 127–135. https://doi.org/10.1016/j.jss.2014.12.002
P37	Lopez-Martin, C. (2011). A fuzzy logic model for predicting the development effort of short scale programs based upon two independent variables. *Applied Soft Computing Journal*, *11*(1), 724–732. https://doi.org/10.1016/j.asoc.2009.12.034