Tomas Bata University in Zlín
Faculty of Applied Informatics

Doctoral Thesis

Regression Models for Software Project Effort Estimation

Regresní modely pro odhad úsilí softwarového projektu

Author:	**Huynh Thai Hoc**
Degree programme:	Engineering Informatics
Degree course:	Software Engineering
Supervisor:	Assoc. Prof. Ing. Zdenka Prokopová, CSc.
Consulting Supervisor:	Assoc. Prof. Ing. Petr Šilhavý, Ph.D.

Zlín, October 2023

ACKNOWLEDGEMENT

Completing my PhD degree has been a momentous achievement, and I am deeply thankful to all the individuals and institutions who supported and encouraged me throughout this extraordinary journey.

I want to express my heartfelt gratitude to my exceptional supervisors, doc. Ing. Zdenka Prokopova, CSc., who provided invaluable guidance and unwavering patience, doc. Ing. Petr Šilhavý, Ph.D, for his continuous support and doc. Ing. Radek Šilhavý, Ph.D., for his constructive feedback that significantly contributed to the development of my research.

I also sincerely appreciate the Faculty of Applied Informatics at Tomas Bata University for allowing me to partake in this doctoral course, which has allowed me to expand my knowledge and excel in my field of interest.

Special thanks are due to the colleagues whose supportive environment enriched my learning journey. The collaboration and knowledge sharing among faculty and staff at the Faculty of Applied Informatics, UTB, has been truly invaluable.

Furthermore, I wish to extend my appreciation to doc. Ing. Anežka Lengálová, Ph.D., and Ing. Dagmar Svobodová, MSc., for their assistance in improving my English language skills.

I am deeply grateful to Professor Hoàng Lê Minh, Head of the Faculty of Information Technology at Văn Lang University, Dr.Sc., and Doc. Dr Hông Vân Lê from the Institute of Mathematics, Czech Academy of Sciences, for their inspiring encouragement.

My sincere thanks go to RNDr. Martin Fajkus, Ph.D., for his collaboration on the research published in the mathematics journal.

I also want to express my gratitude to Professor Nguyen Tien Zung from the Institut de Mathématiques de Toulouse, as well as Dr. Susely Figueroa Iglesias, Dr. Nguyen Thi Thuy Nga, and all my colleagues at Torus Actions, for their dedicated guidance during my internship.

I am thankful to František Brázdilík and his colleagues for their invaluable support in visa matters related to my study abroad.

My most profound appreciation goes to my family, especially my spouse, Nguyen Thi Hanh Uyen, for her unwavering support and sacrifices essential to my academic pursuits.

Last but not least, I extend my heartfelt gratitude to my parents, Huynh Van Chien and Tran Thi Thoi, my siblings, my friends, and my children, Huynh Nguyen Minh Quan, Huynh Nguyen Gia Huy, and Huynh Minh Thanh, for their love and encouragement throughout my studies.

ABSTRAKT

Odhad úsilí při vývoji softwaru, resp. odhad pracnosti vývoje softwarových projektů, hraje klíčovou roli v oblasti vývoje softwaru a má velký vliv na plánování projektů a přidělování zdrojů. Předkládaná práce přináší významné pokroky v oblasti odhadu úsilí při vývoji softwaru zavedením inovativních technik jako je tzv. přenosové učení (transfer learning) a analýzy datových souborů, s cílem zvýšit přesnost odhadu úsilí, konkrétně v rámci rozšíření metody funkčních bodů. Kromě toho jsou v předkládané práci zkoumané různé přístupy k identifikaci faktorů analýzy funkčních bodů a relevantních kategoriálních faktorů, které přispívají ke zlepšení odhadu úsilí, včetně vícenásobné lineární regrese, neuronových sítí atd.

Prostřednictvím rozsáhlé série experimentů autor práce identifikuje nové faktory ovlivňující odhad úsilí, což vede k přesnějším odhadům ve srovnání se základními modely. Dále je v práci popsaná aplikace technik LIME (Local Interpretable Model-agnostic Explanations) a SHAP (SHapley Additive exPlanations), které umožňují hlubší vhled do černé skříňky predikčních modelů.

Provedený výzkum byl zaměřen na hodnocení účinnosti předem natrénovaných modelů a návrh využití metod tzv. hlubokého učení (deep learning) v kombinaci se strategiemi pro vyvažování kategoriálních proměnných s cílem zlepšit odhad úsilí. Výsledky jasně ukazují, že zahrnutí relevantních faktorů a využití hlubokého učení, jakož i technik přenosového učení, výrazně zlepšuje odhad úsilí při vývoji softwaru. Toto zlepšení odhadu úsilí nabízí týmům zabývajícím se vývojem softwaru přesnější prostředky, což v konečném důsledku vede ke zlepšení plánování a řízení projektů.

Předkládaná práce celkově přispívá k teoretickým i praktickým aspektům odhadu úsilí tím, že poskytuje nové poznatky a inovativní strategie pro zvýšení přesnosti odhadu úsilí při vývoji softwarových projektů.

Key words in Czech: *Odhadování pracnosti vývoje softwarových projektů, metoda funkčních bodů, regresní modely, hluboké učení*

ABSTRACT

Effort estimation plays a crucial role in the domain of software development, employing an influence on project planning and resource allocation. This thesis advances the field of Software Development Effort Estimation (SDEE) by introducing novel transfer learning and dataset balancing techniques to enhance effort estimation accuracy, focusing on the function point analysis. It explores multiple linear regression, feedforward neural networks, and ensemble methods to identify factors affecting effort estimation.

Through a comprehensive series of experiments, this study uncovers new factors that significantly improve effort estimation, resulting in more precise estimates when compared to baseline models. Furthermore, it employs the application of Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP) techniques to provide deeper insights into the black-box of predictive models.

This research evaluates the effectiveness of pre-trained models and suggests using deep learning methods in combination with strategies for balancing categorical variables to enhance effort estimation. The results indicate that incorporating relevant factors and employing deep learning and transfer learning techniques enhances SDEE. This improvement in effort estimation offers software development teams a more accurate means of estimation, ultimately leading to improved project planning and management.

In summary, this thesis contributes to both theory and practice in effort estimation by offering innovative insights and strategies to boost accuracy.

Key words: *Software development effort estimation, function points methods, regression models, deep learning, Ensemble, deep learning with balancing dataset, transfer learning, LIME, SHAP.*

Contents Of the Thesis

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

Abbreviation	Definition
ADASYN	Adaptive Synthetic Sampling Approach
AFP	Adjusted Function Points
ANN	Artificial Neural Network
CMMI	Capability Maturity Model Integrated
COCOMO	Constructive Cost Model
CSBSG	Chinese Software Benchmarking Standard Group
DL	Deep learning
DLMLP	Deep learning - Multilayer perceptron
DLMLPB	Deep learning - Multilayer perceptron with balancing dataset
DTF	Decision Tree Forest
EI	External Inputs
EIF	External Interface File
EO	External Outputs
EQ	External Inquiry
FFNN	FeedForward Neural Network
FPA	Function Point Analysis
GMM	Gaussian Combination Model
GMM	Gaussian Combination Model
GSC	General Systems Characteristic
HGBoost	Histogram Gradient Boosting
IFPUG	International Function Point Users Group
ILF	Internal Logical File
IQR	Interquartile Range
IS	Industry Sector
ISBSG	International Software Benchmarking Standards Group
LIME	Local Interpretable Model-agnostic Explanations
MAE	Mean Absolute Error
MBRE	Mean Balance Relative Error
MFP	Modified Function Points
MIBRE	Mean Inverted Balance Relative Error
MLP	Multilayer perceptron
MLR	Multiple Linear Regression
MMRE	Mean Magnitude of Relative Error

MRE	The Magnitude of Relative Error
MSE	Mean Square Error
NESMA	Netherlands Software Metrics Association
PDR	Productivity Rate
PRED(x)	Prediction at level x
ReLU	Rectified Linear Unit
RF	Random Forest
RQ	Research Question
RS	Relative Size
SA	Standardised Accuracy
SDEE	Software Development Effort Estimation
SDO	Software Development Organization
SHAP	SHapley Additive exPlanations
SMOTE	Synthetic Minority Over-sample Technique
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SVR	Support Vector Regression
TL-Case 1	Transfer Learning – Case 1
TL-Case 2	Transfer Learning – Case 2
TL-Case 3	Transfer Learning – Case 3
UFP	Unadjusted Function Points
VAF	Value Adjustment Factor
WBS	Work Breakdown Structure

1. INTRODUCTION

This section presents the motivation of the thesis, identifies the specific problem that needs to be addressed, and presents the research questions. Additionally, it outlines the objectives of the study.

1.1 Motivation

The motivation for this thesis is essential to provide the estimating field with a new approach to the effort estimation problem, which might supplement current practices. The following are the key drivers behind this motivation:

i) The absence of categorical variables might result in less effort estimation accuracy measured by traditional function point analysis (FPA) estimation methods. It is a foundational technique for measuring the functional software size of projects from the user's perspective [1]. Allan J. Albrecht developed this technique in 1979 at IBM, which was extended by the International Function Point Users Group (IFPUG) [1]. It is a measure based on function points and productivity rate. However, in the early stage of software project development, the productivity rate of that project might be unknown. In addition, the complexity of FPA weight metrics values might be affected by many factors (software development methodologies, systems characteristics). Suppose the complexity weights assigned to the components (External Inputs (EI), External Outputs (EO), External Inquiry (EQ), External Interface File (EIF), Internal Logical File (ILF)) are not appropriately identified (for example, assigns higher complexity weights to relatively simple components or lower weights to more complex components), it might lead to inaccurate effort estimation. Hence, to estimate the effort required for software development in the initial stages of FPA, the thesis considers estimating the effort by incorporating essential categorical variables such as Industry Sector and Relative Size alongside factors of FPA.

ii) The unavailability of pre-trained models for software effort estimation: Transfer learning is a type of deep learning that involves using a pre-trained model to solve a new problem with similar features or structure [2]. In the context of SDEE, transfer learning could enhance estimation model precision by leveraging knowledge from analogous projects or domains [2]–[4], significantly decreasing the time and resources necessary for model training. Leveraging transfer learning might mitigate the challenge of limited data availability often faced in software estimation. However, despite its potential benefits, several studies have compared the performance of transfer learning with the deep learning approach regarding SDEE. However, they did not propose a pre-trained model [3]–[5]. This issue promotes the thesis to build a pre-trained learning model by leveraging the advantages of transfer learning

and comparing the performance of transfer learning to the deep learning approach in terms of SDEE.

iii) The existing models utilized for effort estimation remain unclear black-boxes covering their internal mechanisms. Consequently, understanding the rationale behind their predictions becomes a formidable challenge for scientists and practitioners. Advanced methodologies such as Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP) offer promising solutions to tackle this pressing limitation while emphasising the importance of result interpretation. LIME generates locally faithful explanations for individual predictions, showing how features (such as EI, EO, EQ, EIF, ILF, and relevant categorical variables) contribute to each prediction, significantly positively/negatively impacting the accuracy of software project effort estimation. On the other hand, SHAP provides a unified framework for interpreting the output of any deep learning model by attributing the prediction to different features' contributions. By providing interpretable and transparent insights into the models' decision-making processes, these techniques delegate researchers to understand the influential factors and their complexity within the software development context. Consequently, leveraging LIME and SHAP can significantly enhance the validity and trustworthiness of effort estimation models, leading to more informed and scientifically driven project management decisions.

1.2 Problem Statement

Among the various approaches to estimating software effort estimation, one common technique in the software industry is FPA. This method is advantageous as it estimates the size of the software. However, as mentioned in publication [6], it is essential to note that FPA has limitations. One significant drawback is that it relies on fixed complexity weight values established using data from IBM in the 1970s [1]. Given the technological progress and changes to the present year (2023), these values have become outdated. Furthermore, due to the unique nature of each company, using these fixed values tends to result in less accurate estimates.

Therefore, this study proposes a group of factors to estimate effort estimation by incorporating categorical variables such as Industry Sector and Relative Size along with factors of FPA based on the International Software Benchmarking Standards Group (ISBSG) [7] as the historical dataset. The first study uses various approaches such as regression model, random forest, ensemble approach, and deep learning based on feedforward neural network with multilayer perceptron (DLMLP) [8], [9] to determine the factors of FPA incorporate with Industry Sector and Relative Size lead to more accurate effort estimation. In addition, the effectiveness of these models is further explored by employing balanced datasets in the DLMLP model to address the limitation of imbalanced data, a common

issue in effort estimation research. The thesis might hardly examine all known algorithms and all combinations of factors of FPA. Therefore, selecting some experimental algorithms and combinations of factors are also matters of concern.

As highlighted in the motivation section, the advantages of transfer learning are substantial [2], [10], [11]. It enhances estimation model accuracy by leveraging insights from related projects, significantly reducing the resources and time required for model training. This thesis proposes a transfer learning technique that effectively estimates effort using the ISBSG dataset. Simultaneously, the thesis evaluates the applicability of this technique across similar datasets such as Albrecht, China. Additionally, an endeavour is undertaken to construct a pre-trained model obtained from the ISBSG dataset, intended as a reusable library for researchers.

On the other hand, several machine-learning approaches were adopted to increase effort estimation accuracy[12]–[19]. However, the resultant models remain a mystery (black box). The comprehensive comparison of these influential factors within the predictions holds critical importance, as it might give researchers invaluable insights grounded in the predictions. As illustrated in the motivation section, this research endeavour extensively analyses predicted efforts via LIME [20] and SHAP [21]. Specifically, the focus lies on dissecting LIME and SHAP within the DLMLP model to illuminate how factors, especially categorical variables such as Industry Sector and Relative Size, impact effort estimation. Nevertheless, due to limited time for analysis across all experimental models, this study concentrates on DLMLP models, prioritizing a comprehensive exploration within this confined scope.

To sum up, this study addresses the challenges in software effort estimation by focusing on FPA limitations and proposing a group of factors that integrates FPA factors with categorical variables. The research selectively employs various methodologies to determine the most accurate effort estimation approach based on that group of factors. Moreover, it harnesses transfer learning's benefits, tailoring a model using the ISBSG dataset and evaluating its applicability across diverse datasets. The study also sheds light on the black-box of models, employing LIME and SHAP techniques for analysis.

1.3 Research Questions and Hypothesis

In this thesis, five RQs and hypothesis must be answered:

1. **RQ1**: Which yields greater accuracy in software effort estimation: DLMLP, MLR, or Random Forest?
   $\mu DLMLP =$ Mean accuracy of DLMLP
   μMLR = Mean accuracy of MLR
   $\mu RF =$ Mean accuracy of Random Forest
   - H_1: $\mu DLMLP > \mu MLR$ and $\mu DLMLP > \mu RF$

This hypothesis states that DLMLP accuracy is greater than MLR and RF in software effort estimation.

- The null hypothesis $H_0: \mu DLMLP \leq \mu MLR \; or \; \mu DLMLP \leq \mu RF$
  This hypothesis states that DLMLP is either less or equally accurate as at least one of the other two methods in software effort estimation.

2. **RQ2**: Does dataset balancing enhance the predictive accuracy of DLMLP methods in software effort estimation?
   $\mu DLMLPB =$ Mean accuracy of DLMLP with Balancing
   $\mu DLMLP =$ Mean accuracy of DLMLP without Balancing
   - $H_1: \mu DLMLPB > \mu DLMLP$
     This hypothesis states that the accuracy of deep learning with a balancing dataset is more significant than without balancing.
   - The null hypothesis $H_0: \mu DLMLPB \leq \mu DLMLP$
     This hypothesis states that deep learning with a balancing dataset is either less or equally accurate as deep learning without balancing.
3. **RQ3**: For software effort estimation, does a combined ensemble of MLR, Random Forest, and DLMLP outperform each standalone model?
   $\mu ENS =$ Mean accuracy of Ensemble
   - $H_1: \mu ENS > \mu DLMLP$, $\mu ENS > \mu MLR$, and $\mu ENS > \mu RF$
     This hypothesis states that the accuracy of the ensemble is greater than DLMLP, MLR, and RF in software effort estimation.
   - The null hypothesis H_0:
     $$\mu ENS \leq \mu DLMLP, or \; \mu ENS \leq \mu MLR, or \; \mu ENS \leq \mu RF$$
     This hypothesis states that the ensemble is either less or equally accurate as at least one of the other three methods.
4. **RQ4**: Does transfer learning offer any accuracy advantages over conventional DLMLP approaches in software effort estimation?
   $\mu TL =$ Mean accuracy of Transfer Learning
   - $H_1: \mu TL > \mu DLMLP$
     This hypothesis states that the accuracy of deep learning by applying transfer learning is more significant than that of deep learning without applying transfer learning.
   - The null hypothesis $H_0: \mu TL \leq \mu DLMLP$
     This hypothesis states that deep learning by applying transfer learning is either less or equally accurate as deep learning without applying transfer learning.
5. **RQ5**: Do the categorical variables (IS and RS) influence effort estimation accuracy?
   $\beta IS =$ Regression coefficient for IS
   $\gamma RS =$ Regression coefficient for RS
   - **Null Hypothesis (H_0)**: $\beta IS = \gamma RS = 0$ (indicating that IS and RS do not affect the accuracy of effort estimation)

- **Alternative Hypothesis (H_1)**: βIS or γRS is not equal to 0 (indicating RS or IS effect on the accuracy of effort estimation)

1.4 Objectives of the Thesis

This section outlines the objectives pursued in this research, focusing on advancing the state-of-the-art in the identified issues. The specific research objectives present in this thesis can be summarized as follows:

1. To enhance the accuracy of effort estimation in terms of FPA.
2. To evaluate the efficacy of various estimation methodologies, such as multiple linear regression, random forest, deep learning based on multilayer perceptrons, deep learning with balanced datasets; ensemble techniques established by incorporating multiple linear regression, random forest, and deep learning models; transfer learning for effort estimation. This evaluation involves validating the results using appropriate datasets.
3. To introduce a pre-trained model based on the ISBSG dataset, providing a comprehensive and reliable foundation for effort estimations. The relevant other datasets illustrate the performance of effort estimation based on the pre-trained model.
4. To leverage advanced techniques such as LIME and SHAP to gain comprehensive insights into the contribution and local importance of different features, namely EI, EO, EQ, EIF, ILF, IS, and RS, within the proposed effort estimation models in terms of FPA.

Thus, the research objective of this thesis is to establish innovative approaches for estimating the effort required in software product development. These approaches are rigorously compared with the performance of effort estimation based on the ISBSG dataset and other relevant datasets, facilitating the identification of superior estimation techniques with practical applicability.

2. THE CURRENT STATE OF THE ISSUES DEALT WITH

This section presents the literature review of software development effort estimation.

2.1 Software Development Effort Estimation

The process of SDEE is complex. Building software might require the highest quality and lowest resources. The resources of a project might include budget, time, and staff resources. They are used to manage, design, develop, enhance, or maintain a project. The duration, budget, or effort to complete a project might depend on the project's number of modules or use cases. However, how much funding is spent on a project might always be a challenge for project

managers/team leaders who are responsible for the entire project [22]. The less accurate estimation effort might result in the inexactly allocating budget to complete all project phases and lead to project failure [23].

Software project failures are the most prominent illustration of the difficulty in managing massive, distributed software systems [24]. According to the Standish Group, many software companies still put no practical software costs forward or work within strict schedules – and completed their projects behind cost overruns – (48%- 65%) or failed to complete them at all – (48 % - 56%) [25]. Compared to the estimation, most projects' efforts and deadlines are overrun. If the software cost is underestimated, the project is inefficient, and the final cost certainly is exceeded. Finally, these overestimated projects often extend and consume more resources than anticipated, even if they are completed on time. At the same time, the functionality and quality of these undervalued initiatives are reduced to meet the plan's requirements [26]. These might result in the organisation losing the bid or wasting time, funds, employees, and other resources, resulting in a monetary loss or even collapse.

Regarding methods to estimate effort estimation, they might be classified into two kinds of methods, as given in Table 2-1, including non-algorithmic and algorithmic-based techniques [27]. Non-algorithmic techniques are effort estimation based on expert judgment or project expertise [28], such as Expert Judgment, Analogy, Wideband Delphi, or Work Breakdown Structure. On the other hand, algorithmic techniques might be based on the formula to measure the effort of software projects in terms of functional software size [29] or use case points [30], NESMA [31], etc.

Expert opinion refers to the informed judgment provided by an individual or group of experts regarding a particular subject or an unknown measurement [32]. In the context of the current investigation, expert judgment is employed for estimating the effort required for software projects. Expert judgment is crucial in software estimation, especially in two significant scenarios. Firstly, when empirical data is limited or not easily obtainable, and secondly, when tackling the estimation of intricate, unclear, or poorly defined problems [33]. These circumstances form the basis for the extensive acceptance of expert judgment as an approach to software estimation. It is important to note that the accuracy of such estimations largely relies on the extent to which a new project aligns with the expert's experience and expertise.

The primary idea of the analogy technique involves representing a software project using various variables or characteristics and then identifying completed projects that share similar attributes [34]. These variables might include several inputs, outputs, application domains, etc. By leveraging the effort values of these completed projects, we might create an effort estimation for the new project. This approach might be considered a combination of completed projects and expert judgment techniques. According to M. Shepperd et al. [34], while estimating by analogy is simple, several challenges must be addressed. Firstly, we must

determine the most effective way to describe projects. This determination could involve considering factors such as the application domain, the number of inputs, the number of distinct entities referenced, and the number of screens. It is crucial to select available variables when the prediction is needed, and they should accurately characterise the project as much as possible. The second challenge involves assessing project similarity and establishing confidence in the analogies. How do we determine which projects are comparable, and how much can we rely on these analogies?

Additionally, it is crucial to determine the optimal number of analogies to search. Too few analogies might result in including outlier projects, while too many could dilute the impact of the closest analogies. Lastly, we must address how to utilize the known effort values from analogous projects to derive an estimate for the new project. Potential approaches include using means and weighted means, where closer analogies significantly influence the estimate. The selection depends on the specific context and data available, and careful consideration must be given to ensure the most appropriate and reliable estimation process.

Delphi is an organised technique for interactive estimation and symmetric prediction derived from expert questionnaires [35], [36]. The method might be presented in the following steps. First, a series of questions are provided in a questionnaire, and experts must respond anonymously to these questions over multiple rounds. Next, the coordinator summarises the predictions made by the experts in the previous round. Experts are expected to provide justifications for their choices and have the opportunity to review the questions and answers of other experts. Through several rounds, a consensus among the experts is reached regarding the parameters and stability of the results. This iterative process reduces the range of possible answers and brings the group of experts closer to the correct solution [37]. A variation of the Delphi method known as Wideband Delphi was proposed by Boehm [38]. The term "wideband" indicates more significant interaction and communication among the participating experts compared to the original Delphi method.

Table 2-1: The classification of techniques adopted in software effort estimation

Type	Estimation method	Description
Non-algorithmic	Expert Judgment	Estimations based on expert experiences and/or intuition [24]
	Analogy	Estimations are based on the actual cost of similar completed projects [24]
	Price-to-win	Estimations based on customer budgets [24]

	Bottom-up	Estimations based on customer budgets [24]
	Top-up	Estimations based on customer budgets [24]
	Wideband Delphi	Customer and technical teams are involved in the estimation process [24]
	Planning Poker	Estimations based on collaboration/consensus among team members like Wideband Delphi [24]
Algorithmic	SLOC	Estimations based on the previous data of the completed project. It cannot compare different programming language lines of code [24]
	Function Point Analysis	Estimations based on counting essential software components [28]
	Object Point	Estimations are based on objects' numbers and complexity. Estimation Steps: counting objects, the classification of things, and the weight of items related to complexity [24]
	COCOMO	There are four COCOMO methods [28]: Simple COCOMO, Intermediate COCOMO, Detailed COCOMO, and Testing steps· COCOMO II.
	Use Case Points	Estimations derived by counting use cases [29]

The work breakdown structure (WBS) breaks down an engineering project into smaller components such as subprojects, tasks, subtasks, and work packages [35]. It holds significance as a planning tool that establishes a logical framework linking objectives, resources, and activities. During the project's execution, the WBS plays a vital role as it tracks the progress of subtasks against the project plan. The primary objectives of creating the WBS include identifying work tasks, required resources, necessary, and other pertinent details with the level of precision specified in the original plan. Additionally, it enables early accuracy assessment and allows for corrective replanning, if necessary, during the project's execution.

FPA is a foundational technique for measuring the functional software size of projects from the user's perspective [1]. Allan J. Albrecht developed this technique in 1979 at IBM. Then, it was extended by the IFPUG [1]. According to [39], FPA estimates software development/maintenance independently of the technology used for implementation; for example, the functional size should be the same regardless of the problem domain, programming language, or development type. The use cases can be beneficial in estimating software effort

estimation early in the project – before gathering the necessary information – during the software life cycle requirements [40]. Use cases are anticipated to provide an accurate estimate of the software effort estimation of the future system in question. Publication [40] surveyed the strategies used to elicit, describe, and model requirements. They claimed that use cases were used in the initial stages of over half of these software initiatives. As a result, using use cases for software effort estimation has grown in popularity.

The Netherlands Software Metrics Association (NESMA) FPA method [31] follows the same rules as the IFPUG FPA method. ISO accepted it as an international standard in 2005 [41]. NESMA, the user group for function points in the Netherlands, recommends three types of function point counts based on the level of detail achievable: detailed, estimative, and indicative. The detailed function point count corresponds to the IFPUG count. In the estimative function point count, the following steps are followed: (1) identification of all functions belonging to the five types: ILF, EIF, EI, EO, and EQ; (2) calculation of the total unadjusted function point count by assuming low complexity for each data function point and average complexity for each transaction point.

2.2 Regression Models in SDEE

Several studies have utilised regression models to enhance estimating effort in software engineering. For instance, Sharma and Chaudharyin [39] applied MLR to estimate the effort required for agile software development. They developed three models based on MLR with stepwise regression to estimate agile software development effort and assessed the performance metrics using Mean Squared Error (MSE) and Mean Magnitude of Relative Error (MMRE). The findings revealed that their proposed model outperformed three commonly used techniques: decision trees, stochastic gradient boosting, and random forests. Hai et al. [40] considered productivity rate (PDR) as the dependent variable and independent variables, including Value Adjustment Factor (VAF), EI, EO, EQ, EIF, and ILF for measuring the effort of FPA, based on the ISBSG 2018/release R2 and multiple regression model. The authors concluded that their approach could potentially outperform existing methods.

In 2013, Nassif et al. [41] compared three models for estimating software: effort decision tree forest (DTF), decision tree, and MLR. The authors stated that the DTF model fared better than the other two models according to all evaluation criteria, and statistical tests were used to verify its robustness. Based on the findings, they concluded that the DTF model is a good choice for forecasting software efforts.

Furthermore, the VAF plays a crucial role in improving the accuracy of Adjusted Function Points based on 14 General System Characteristics. However, according to ISBSG, this component may have gone uncounted for most projects recently, and the VAF is assumed to be one of them. As a result of this problem,

effort estimation in Function Point Analysis may be inaccurate. Prokopova et al. [42] introduced Modified Function Points (MFP) methodologies based on the regression model approach in 2018 and investigated the impact of VAF on software effort estimation accuracy. Three techniques for estimating effort were tested in the variants without and with the VAF factor based on ISBSG. As a result, the VAF factor has no bearing on estimating precision.

Naïve Bayes [38] is a well-known probabilistic classifier in data mining. For calculations, it is assumed that project characteristics are unrelated. Even though this assumption is false in most cases, Naïve Bayes outperforms other complex approaches in various practical applications such as text categorisation [43] and kernel density estimation [44]. Zhang et al. [45] presented the Bayesian regression and Expectation Maximization algorithm to predict effort estimation based on missing values on historical datasets. Their approach was based on the assumption that the characteristics of projects are unrelated. They used the ISBSG and Chinese Software Benchmarking Standard Group (CSBSG) datasets released in 2006 as observational datasets. Moreover, this study also proposed the Missing Data Toleration and Missing Data Imputation strategy were proposed to handle missing data. This paper used PRED(0.25) and Wilcoxon signed-rank tests of the MREs as performance metrics.

In 2013, Fedotova et al. [29] discussed the most popular approaches used in software effort estimation. It introduced a study conducted in a software development organisation applying the Capability Maturity Model Integrated (CMMI) architecture. Currently, a software development organisation (SDO) forecasts software initiatives based on the judgment of a single professional. The drawbacks of this approach prompted the SDO to replace the current effort measurement method with a structured one. The stepwise MLR technique was chosen and implemented for the software development and testing processes. The MLR findings were compared to the forecasts given by the region expert. As a result, the model collected for the research team outperformed expert judgments. However, no satisfactory model was found for the development team, and a recommendation for obtaining data from new variables was introduced.

Another research regarding effort estimation by adopting a regression model was presented in the publication [46]. The authors compared stepwise with Lasso regression. COCOMO81, Desharnais, and Maxwell datasets were used in this study. They stated that Lasso-based and stepwise regression illustrated different preferences. However, they concluded that Lasso-based selection was preferable to stepwise regression. MMRE, PRED (0.25), SA, etc., were performance metrics.

Silhavy P et al. [47] employed the stepwise regression technique to develop distinctive estimation models for each segment. Their research involved a comparative analysis of the estimation accuracy achieved by these models in contrast to clustering-based models and the IFPUG. Their findings revealed that the proposed model yielded improved estimation accuracy compared to baseline

approaches, including non-clustered functional point analysis and clustering-based models.

2.3 Multilayer Perceptron in SDEE

On the other hand, several machine-learning techniques have been used to predict SDEE [12]. One of the techniques mentioned in previous reports [39] is MLP. Ramessur and Nagowah [48] proposed a model that assesses and forecasts effort estimation while considering various parameters influencing performance. The model was validated using a variety of regression algorithms, including linear regression, K-nearest neighbour, decision tree, polynomial kernel, radius basis function, and MLP. As a result, the model produced more accurate estimates with lower error values when using the MLP method. MLP was also proposed by Suyash Shukla et al. [49] to improve the SDEE. The authors introduced a technique based on a genetic algorithm to adjust the complexity weight metrics of Function Point.

Furthermore, Somya Goyal et al. [14] used MLP with the Back Propagation algorithm to develop a non-linear technique for effort estimation. Their goal was to compare neural network models unbiasedly, using ISBSG Release 11 as a practical dataset with over 5000 completed projects as a practical dataset, Adjusted Function Points (AFP) and other categorical variables as input factors for their models. On the other hand, deep learning is an approach suggested in previous papers [48]. However, the problem is that which method leads to more accurate effort estimation efficiency according to FPA has not been mentioned.

In 2020, M.Ochodek et al. [50] created the Deep-COSMIC-UC model to address functional needs that lack specific information. The suggested model was an advanced convolutional neural network. They aimed to develop a brand-new prediction engine that could roughly forecast the COSMIC size of use cases based solely on their names. This model may compare the COSMIC size of the use case based on the raw text that represents the use-case name.

In 2016, Nassif A et al. [51] discussed four neural network models: MLP, along with three other models (general regression neural network, radial basis function neural network, and cascade correlation neural network) using the ISBSG dataset. They were compared based on predictive accuracy, a tendency to over/underestimate and how they classify input importance. The cascade correlation neural network performed the best on most datasets based on the mean absolute residual criterion.

Madheswaran and Sivakumar [52] used a Multilayer Feed Forward Neural Network to accommodate the COCOMO model's prediction performance so that the estimated effort was relevant to the actual effort. The network was trained using the backpropagation learning technique by repeatedly processing training samples and comparing the network's prediction with the actual effort. They applied the COCOMO I dataset to train and test the network, and it was

discovered that the proposed neural network model enhances the model's estimation accuracy. The COCOMO model's test comparison was made to the trained neural networks.

Mukherjee et al. [53] considered the adoption of a neural network for optimizing project effort estimation. For improved accuracy in effort estimation, they utilized a two-layer FFNN with sigmoid neurons in the hidden layer and linear neurons in the output layer. The LevenbergMarquardt backpropagation algorithm was used to train the network. They studied the COCOMO 81 dataset with evaluation criteria as MRE. They concluded from the experimental results that their suggested model outperforms Anupama Kaushik et al.'s proposed and Constructive Cost models.

The FeedForward Neural Network (FFNN) technique to predict the duration of a new software project was proposed by [9]. Two models were generated from the ISBSG (2009, Release 11) data, whose projects were measured in adjusted function points. The accuracy of this FFNN was compared against that of a statistical regression model. An accuracy comparison was made based on an ANOVA observing its assumptions of residuals. Results accepted the following hypothesis: The accuracy of duration prediction for an FFNN was statistically better than that obtained from a statistical regression model when an adjusted function points value obtained from new software development projects was used as the independent variable.

Based on data acquired from the CMMI organisation [54], comprising 163 software development projects, Pai et al. [55] developed software effort estimation models utilising Artificial Neural Network (ANN) ensembles and regression analysis. The paper's primary focus was on creating an effective experimental design in order to obtain superior effort estimation results. They also compared ANNs and multiple regression analyses regarding software effort estimation. They discovered two intriguing outcomes. First, other than size (function points), other variables were not valuable for estimating software projects. Second, a correctly designed ANN ensemble improved regression analysis estimates and could yield excellent effort estimate estimations.

2.4 Balancing Dataset in SDEE

Balancing the dataset is a common preprocessing step in various machine-learning applications to address class imbalances [56]. Class imbalances occur when the distribution of classes in the dataset is uneven, with one or a few classes having significantly more or fewer instances than others. Machine learning algorithms operate assuming a balanced class distribution within the dataset. As a result, classifiers tend to exhibit a bias towards the majority class when confronted with imbalanced datasets. According to Liu et al. [56], in these scenarios, the minority class often represents the focal point of interest.

In image processing, several researchers [57]–[59] proposed several approaches to dealing with imbalanced datasets in their study. Many methods might be used, such as over-sampling/ under-sampling [60], class weighting [61], and SMOTE [62]. In 2018, Min Zhu et al. [63] proposed class weights random forest to deal with imbalanced datasets in medical applications. Their approach addresses assigning individual weights for each class. In 2022, Islam A. et al. [64] proposed oversampling based on K-Nearest Neighbor to tackle imbalanced image datasets. This method involved identifying pivotal and secured regions for augmentation, subsequently generating synthetic data instances for the minority class. The researchers reported its superiority over other approaches, such as SMOTE and ADASYN, in terms of performance.

Zhijun Ren et al. [65] also introduced an innovative approach involving loss function weighting to enhance the efficacy of intelligent diagnostic models when dealing with imbalanced data. In this method, the weight of each sample's loss was determined by considering factors such as the distribution and convergence of samples and classes. By incorporating weighted losses during model training, the contribution of each class to parameter updates was balanced, thereby mitigating the impact of dominant majority classes in imbalanced training datasets.

However, research on balancing datasets based on attributes to enhance the effectiveness of effort estimation models is currently limited in the field of effort estimation. This issue poses a challenge as models trained on imbalanced datasets may exhibit bias towards the majority category, resulting in an inadequate performance for the minority category. In such instances, the model may achieve a high accuracy rate by simply predicting the majority category for all instances, but this approach holds limited practical value [66].

The term "balanced dataset" pertains to the distribution of data in a dataset, aiming to achieve equilibrium concerning the observed attribute [60], [67]. For example, when examining the ISBSG training dataset, the Industry Sector attribute encompasses categories like banking, government, financial, and more [7]. If the number of instances for each category (banking, government, financial) is equal in the training data, it might be classified as a balanced dataset regarding the industry sector. Conversely, if there is a notable discrepancy in the number of instances among the categories within the industry sector, it might be labelled as an imbalanced dataset.

Within the context of imbalanced datasets related to the classification variable Industry Sector, this thesis aims to comprehensively evaluate the effectiveness of effort estimation by utilizing both balanced and imbalanced datasets. The primary objective is to assess and compare the performance of effort estimation models when trained on datasets that have been balanced versus those that remain unbalanced. Through this rigorous evaluation, the thesis seeks to elucidate the influence of dataset-balancing techniques on the accuracy of effort estimation outcomes. By addressing this crucial aspect, the research contributes valuable

insights into the importance of data balance in effort estimation, aiding practitioners in making informed decisions and enhancing the reliability of their estimation processes.

2.5 Ensemble Approach in SDEE

The ensemble (vice versus sole) methodology [68] aims to combine multiple models to create a prediction model. In 2023, Shukla et al. [69] proposed an ensemble model for estimating effort estimation based on use case points. There were five different techniques to create different ensemble models. They include linear regression, K-nearest neighbour, decision tree, support vector regression (SVR), and multilayer perceptron as base learners. As a result, the authors stated that the boosting ensemble with SVR as the base learner outperformed the other models.

Beesetti et al. [70] introduced an ensemble approach involving regressor models, utilizing a voting estimator to enhance the predictive accuracy of effort estimation and minimize the bias inherent in individual machine-learning algorithms. The outcomes obtained underscore the superiority of ensemble models over single models when applied to diverse datasets, effectively addressing the bias issue.

In 2022, Somya Goyal [71] presented a heterogeneous stacked ensemble to improve effort estimation based on artificial neural networks, SVR, as base learners. The author used five datasets (Desharnais, Cocomo81, China, Maxwell, and Miyazaki94) achieved from the PROMISE repository as historical datasets. The author concluded that the stacked ensemble outperformed the individual model.

In 2021, P. Suresh Kumar et al. [72] proposed a gradient-boosting regressor model and evaluated its performance against various regression models. Their analysis included models such as stochastic gradient descent, K-nearest neighbour, decision tree, bagging regressor, random forest regressor, Ada-boost regressor, and gradient boosting regressor. This assessment was conducted using two datasets: COCOMO81 and China. The results highlighted the impressive accuracy of the gradient-boosting regressor model, outperforming all other models compared to both datasets.

Varshini et al. [17] suggested a stacked ensemble approach based on Random Forest for estimating the effort required for software development. The authors' findings indicated that the proposed Random Forest stacking approach outperforms single models. This approach might be further for their ability to improve prediction accuracy.

M. Hosni et al. [73] proposed a heterogeneous ensemble that used K-nearest Neighbor, Support Vector Machine, MLP, and Regression Trees—four well-known machine learning approaches. Two widely used datasets were utilised to test the ensemble, and three linear rules were used to assess its performance. The

results revealed that the proposed heterogeneous ensemble technique performed quite well, and no particular optimum combiner rule can be suggested in light of the data.

In 2019, P. K. M. Passakorn [74] conducted a research studied to explore the applicability of machine-learning techniques, which had demonstrated excellence in recent data science competitions, in estimating software effort. The investigation examined 14 machine learning methods, including popular approaches liked gradient boosting machine and deep learning, using 13 industry-standard software effort estimation datasets from PROMISE 2015. The study utilised a widely adopted ranking evaluation method for estimating software effort. Notably, the research found that combining multiple effort estimators into a stacked ensemble, such as taking the average of predicted effort levels, resulted in more accurate estimations compared to the individual performance of any of the 14 examined estimators. The study considered the average values derived from the most accurate overall stacked ensemble in determining the estimated effort values. Furthermore, the investigation revealed that employing the boosting principle to create an ensemble improved performance in estimating software effort.

Moreover, Palaniswamy and Venkatesan [75] employed an ensemble technique to enhance prediction accuracy. Traditionally, determining hyperparameters involved a time-consuming process of trial and error tailored to the specific problem and dataset. To address this issue, the researchers' utilised Particle Swarm Optimization (PSO) and Genetic Algorithms to adjust the hyperparameters dynamically. The study constructed a stacking ensemble model using data from the ISBSG dataset, which incorporated information from diverse software projects across countries and companies.

In a related investigation by KK Anitha et al. in 2021 [76], software effort estimation was examined using ensemble techniques and machine/deep-learning algorithms. The authors conducted experiments on multiple datasets, including Albrecht, China, Desharnais, Kemerer, Maxwell, Kitchenham, and Cocomo81, to evaluate the performance of various models. Comparing different ensemble techniques and assessing several stacking models, their results revealed that the proposed random forest stacking method exhibited superior performance when applied to diverse datasets, outperforming SVM, decision trees, and neural networks.

Based on these insights, exploring the effectiveness of a stacking ensemble that incorporates MLR, Random Forest, and Deep learning models for estimation tasks would be worthwhile. This approach could potentially yield improved estimation accuracy, leveraging the strengths of each model while mitigating their weaknesses through ensemble learning techniques. Further experimentation and evaluation would be necessary to validate the performance and generalizability of such a stacking ensemble for estimation tasks.

2.6 Transfer Learning in SDEE

This technique has been developed to improve the performance of a target task by leveraging the knowledge gained while solving a related but different problem [2], [10], [11], [77]. It involves using a pre-trained model to train a new model on a different task. Typically, the pre-trained model is trained on a large dataset, and the learned features are used to initialise the new model's parameters.

Minku et al. [5] investigated the application of transfer learning techniques and demonstrated that incorporating cross-company data might enhance performance. In 2014, Minku et al. [78] proposed a novel framework explicitly designed to capture the relationship between cross-company and within-company projects. This framework facilitated the mapping of cross-company models to the within-company context.

In 2015, Ekrem et al. [77] studied whether transfer learners might be used to predict software effort. They discovered that using the same transfer learning method could estimate transfer effort for cross-company and cross-time challenges. They argued that an organisation's past data might be helpful to present situations, or data from another company may be used for local solutions. They also discovered that transfer learning was a promising research subject in which relevant cross-data was moved across time intervals and domains. However, whether or not the transfer learning-based model should be adjusted to fit the local dataset needs to be stated.

Kocaguneli et al. [3] studied transfer learning in effort estimation. Their study used the Tukutuku dataset, including 125 projects from 8 companies with 19 independent variables, and they used Cocomo81 and Nasa93 for transfer learning. Their findings indicated that a single transfer learning approach might effectively tackle both the challenge of cross-company learning and cross-time learning. Their results dared the prevailing misconceptions: first, historical organizational data held no relevance in the current context, and second, the notion that data from different organizations could not contribute to localized solutions. These findings highlighted transfer learning's potential as a robust avenue, enabling the seamless transfer of pertinent cross-data across various domains and time intervals. However, several noteworthy gaps persist within this domain. Firstly, the data employed for training such models might be subject to limitations, particularly when investigating specific domains or distinct periods. This potential limitation could give rise to data imbalance issues, subsequently influencing the predictive efficacy of the models. Secondly, it is noteworthy that, at present, a conspicuous absence of proposed models or readily available pre-trained libraries exists for researchers to incorporate into their studies within the context of effort estimation. Integrating pre-trained models might be pivotal in unlocking the data's latent potential and catalysing the development of novel applications within this domain. Therefore, this aspect requires thorough discussion to address these limitations and identify potential future research directions.

In 2021, Leandro Minku [4] studied the necessity of using transfer learning in effort estimation. The study explored the potential of transfer learning by investigating whether treating Cross-Company projects as multiple data streams for ongoing learning could enhance effort estimation. An extended model named OATES enabled multi-stream online learning, compared against Dycom and other approaches. Results demonstrated OATES's improved predictive performance when Cross-Company project availability was limited, recommending its adoption for such cases. Conversely, with ample Cross-Company projects available, OATES performed comparably to the state-of-the-art, implying their optional but non-detrimental use. This finding underscores the significance of transfer learning in enhancing SEE outcomes.

To sum up, several studies on transfer learning in effort estimation have been conducted; however, the development of pre-trained models in this domain remains scarce, indicating a significant gap in the availability of such resources. The thesis proposes creating a pre-trained model for effort estimation, leveraging the extensive ISBSG dataset. By harnessing the information within the ISBSG dataset, the proposed pre-trained model holds promising potential for accurately estimating software development efforts. This pioneering approach seeks to bridge the gap and contribute to advancing transfer learning methodologies in the context of effort estimation, thereby enhancing the practicality and efficacy of this essential software engineering task.

2.7 Absence of Categorical Variables in FPA

Effort estimation might still be the most challenging process for estimators in software engineering. This challenge might be due to the diverse project lifecycle models, which may need varying resources at distinct phases of the project [79]. The standard estimation [80] requires more effort to record activities, increasing the difficulty and duration of the estimate. Furthermore, the experience of software developers, the software team's project history in the same business domain, and various other characteristics, as well as the relationships between these factors, are sometimes not accurately predicted [81].

Moreover, as equation (4) mentioned, effort estimation is measured based on the function points and PDR. PDR is the number of hours required to complete one function point. Function points (AFP or UFP) are calculated by assessing the five main components of a software system: EI, EO, EQ, EIF, and ILF. Each of these components is assigned a weight based on its complexity. However, the PDR of that project might be unknown in the early stage of software project development. At the same time, the complexity of FPA weight metrics values might be affected by many factors (such as software development methodologies or systems characteristics) [82]. Suppose the complexity weights assigned to the components (EI, EO, EQ, EIF, ILF) are not appropriately identified; for example, assign higher complexity weights to relatively simple components or lower

weights to more complex components. In that case, it might lead to inaccurate effort estimation.

Table 2-2 presents a comprehensive overview of categorical variables in the context of effort estimation, highlighting the collective endeavours of various researchers to integrate these variables to enhance estimation accuracy. The table enumerates diverse categorical variables alongside their corresponding references, signifying the studies investigating the potential impact of these variables on effort estimation. These categorical variables encompass development type, platform, language, industry sector, organization type, relative size, application type, business area, primary programming language, application group, first data system, methodology, and count approach. The table collectively underscores the significance of categorical variables in contributing to the refinement of effort estimation models.

Table 2-2: The survey of categorical variables

No.	Categorical variables	References
1	Development Type	[83]–[89]
2	Development Platform	[9], [83], [86]–[89]
3	Language Type	[8], [86]–[90]
4	Industry Sector	[47], [86], [88], [91]–[95]
5	Organisation Type	[83], [84], [89]
6	Relative Size	[47], [88], [94], [96]
7	Application Type	[83], [84], [88], [91]
8	Business Area Type	[47], [94], [97]
9	Primary Programming Language	[85], [86]
10	Application Group	[86]
11	1st Data System	[86], [87]
12	Used Methodology	[86], [88]
13	Count Approach	[94], [96]

The thesis explores innovative approaches that transcend the conventional reliance on complexity weights assigned to components and productivity measures. Specifically, this study endeavours to harness proposed methods such as deep learning, deep learning with balancing techniques, ensemble models, or transfer learning. By directing attention to the fundamental factors of EI, EO, EQ, EIF, and ILF alongside categorical variables, these novel approaches strive to elevate the accuracy of effort estimation. This research aims to push the boundaries of traditional effort estimation methodologies, paving the way for more advanced and precise techniques that account for diverse project characteristics and intricacies.

2.8 Model Explainability Approaches: LIME, SHAP

In deep learning, model explainability refers to the ability to interpret and understand the decision-making process of a deep learning model. Deep learning models are frequently regarded as opaque systems due to their complexity and difficulty in interpretation. Model explainability techniques provide insights into why a model makes specific predictions or decisions, shedding light on its internal workings. Two commonly used model explainability techniques in deep learning are LIME and SHAP. LIME stands for "Local Interpretable Model-agnostic Explanations". It is a technique used for explaining the predictions made by machine learning models [20]. It provides interpretable explanations at the local level, which means it explains why a particular instance or example was classified or predicted in a certain way. It is beneficial for black-box models, where comprehending the internal workings of the model is challenging. SHAP [98] is another technique used to explain machine learning model predictions. Similar to LIME, SHAP provides interpretable explanations at the local level. However, SHAP is based on game theory and uses Shapley values to attribute the contribution of each feature to the prediction [21].

To understand clearly LIME, Ribero M et al. [20] gave an example of a medical scenario where a machine learning model was utilised to predict the likelihood of a patient having the flu based on input features such as sneeze, weight, headache, no fatigue, and age. Their example presented the model that produced a flu classification for a given patient. Next, the LIME technique was employed to analyse and comprehend the factors influencing flu prediction to provide transparency and interpretability. For instance, LIME highlighted the sneeze and headache attributes as crucial features in the prediction process. These findings contributed to the interpretability of the model and assisted healthcare professionals in making informed decisions based on the LIME explanations.

To enhance interpretability, SHAP was employed as an alternative to LIME. SHAP utilises game theory and Shapley values to determine the contribution of each feature to the prediction. By leveraging SHAP, healthcare professionals, for

example, might have gained insights into the importance of features such as sneezing and headaches, enabling informed decision-making based on their expertise.

In 2023, Assia Najm et al. [99] employed the SHAP technique to illustrate the effectiveness of model-agnostic approaches in elucidating estimated effort predictions derived from the SVR-RBF model, which was optimized through an artificial immune network within both agile and non-agile contexts. The authors highlighted that this intricate black-box model necessitated interpretation through a range of model-agnostic explanation techniques despite its impressive performance.

Both LIME and SHAP techniques might offer valuable insights into the contribution of various features within the effort estimation models. These methods facilitate understanding the local importance assigned to each feature for a specific instance, thereby enabling more transparent and interpretable explanations. In the context of effort estimation, features such as EI, EO, EQ, EIF, ILF, Industry Sector, or Relative Size might be effectively assessed using LIME and SHAP, providing deeper insights into their significance and impact on the effort estimation. They dissect the model's inner workings, clarifying how specific features (such as EI, EO, EQ, EIF, ILF, IS, and RS) influence positively/negatively the predicted effort. This analysis results in a more profound grasp of the complex relationships and interactions among these features, enhancing the interpretability and reliability of the effort estimation. LIME and SHAP are tools for deciphering model predictions and invaluable aids for decision-making, offering a deeper appreciation of the variables. Integrating LIME and SHAP into the effort estimation landscape enables stakeholders to obtain accurate predictions and comprehend the underlying mechanisms driving those predictions.

3. METHODOLOGY

This section presents the concept of function point analysis for software size. Data collection and subsequent preprocessing ensure high-quality datasets. Model development involves creating predictive models such as deep learning ensembles, while model explainability techniques shed light on predictions rational. Finally, comparison criteria are presented to validate models.

3.1 Function Point Analysis

A Function Point counts the quantity and complexity of a software's functional capabilities depending on user requirements [79]. It was popularised and distributed by the IFPUG [79] in 1986. The IFPUG, the FPA's current regulatory agency, is in charge of improving and developing the norms outlined in the Counting Practices Manual [80]. Since the organisation's inception, the original

FCPA approach has been recognised as the IFPUG - FPA ISO/IEC 20926:2010 is the current standard for it. Similar techniques derived from the baseline FPA include COSMIC, FiSMA, Mark-II, and NESMA [6].

As presented in Figure 3-1, there are several steps to count function points. First, the type of project should be clarified. It might be development, application, or enhancement function point count [100]. In the case of development type, function points can be counted at every development project stage. Application counts are based on the number of function points delivered, excluding any transformation effort (e.g., prototypes or temporary solutions) and any already implemented functionality, and counting the number of Added, Changed, or Removed functions in the enhancement function point. The next step is to collect enquiries on the application and system's technical specifications. These might contribute to determining the type of counting (data or transactional functions) that might be used and the applications and scope's boundaries [100]. Its border defines the distinction between the being examined program and the external applications.

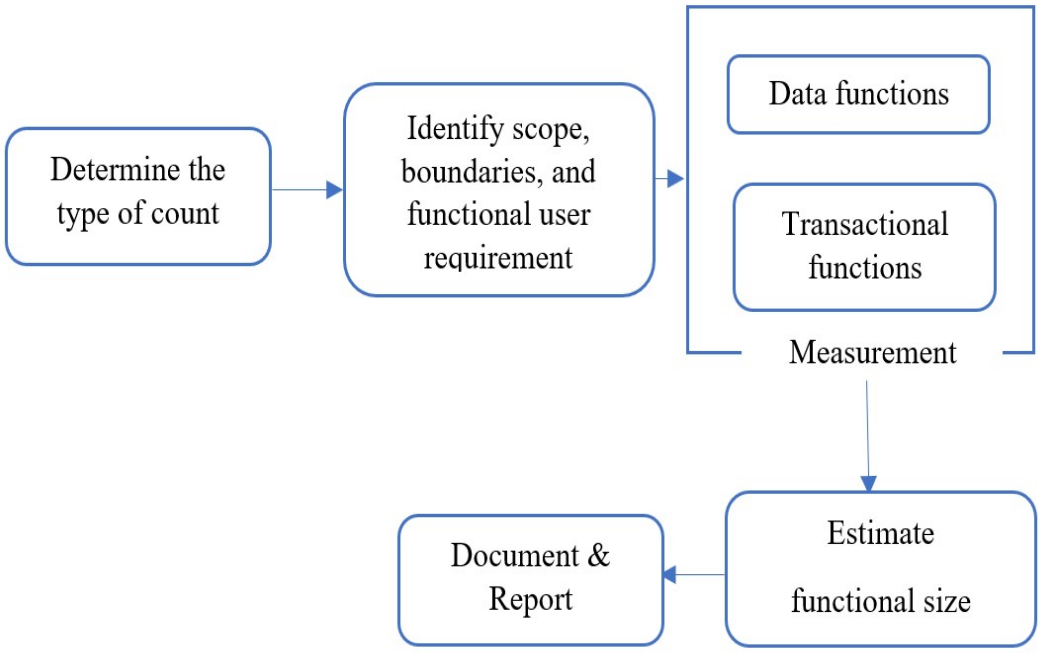


Figure 3-1: The diagram of function point counting [79]

In the next step, this method counts a size attribute as the number of transaction and data function types produced by software projects. The transaction function contains EI, EO, and EQ, while the data function includes EIF and ILF. ILF is a file that is kept by the counted application. The EIF is a file held by another application beyond the border. Table 3-1 shows the complexity weights of each component.

The FPA has the most characteristics that can be applied to estimate software projects in their initial stages [101]. First, function points can be fully allotted based on the requirements or design standards. The projects are in their initial

phases. Second, they have nothing to do with language programming, specialist development tools, or data processing in general [102]. Furthermore, because the function points are built from the user's point of view, non-technical users of the software may find them easier to grasp [103].

A linear combination of size attributes with appropriate three degrees of complexity weights is built to count function points. This function count is also known as UFP. The UFP formula is shown in equation (1).

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} BCs_{ij} \times CWs_{ij} \quad (1)$$

where BCs_{ij} is the count of component i at level j, and CWs_{ij} is an appropriate complexity weight given in Table 3-1.

VAF is measured based on 14 GSCs (see Table 3-2) that rate the overall operation of the application process under consideration. GSCs are commercial constraints imposed on non-technological users. Each attribute includes a description that can be used to calculate the degree of influence. The VAF formula is defined as follows:

$$VAF = 0.65 + 0.01 \times \sum_{i=1}^{14} F_i \times Degree_{Influence} \quad (2)$$

Table 3-1: Complexity weights of FPA components.

Size Attribute	Complexity Weight (CWs)		
	Low	Medium	Large
EI	3	4	6
EO	4	5	7
EQ	3	4	6
EIF	5	7	10
ILF	7	10	15

where F_i represents the GSC factor's effect, the degree of influence ($Degree_{Influence}$) is displayed in Table 3-3.

Table 3-2: General Systems Characteristics (GSCs)

GSC Factors	Characteristic	Description
F1	Data communications	Does the system require backup and recovery?

F2	Distributed Functions	Are Data Required for Communication?
F3	Performance	Does the system include a distributed processing function?
F4	Heavily Used Configuration	Is critical performance required?
F5	Transaction Rate	Will the system work during heavy loads?
F6	Online Data Entry	Does the system require direct data input?
F7	End-User Efficiency	Are multiple screens or operations needed for data inputs?
F8	Online Update	Are the main files up to date?
F9	Complex Processing	Are inputs, outputs, files, and queries intricate?
F10	Reusability	Is internal processing complicated and complex?
F11	Installation Ease	Is the code designed for reuse?
F12	Operational Ease	Are Conversions and Installation Included in Design?
F13	Multiple Sites	Is the application designed for multiple installations in different locations?
F14	Facilitate Change	Is the application designed to make it easy for users to make changes?

The AFP can be calculated using the following equation:

$$AFP = UFP \times VAF \quad (3)$$

Table 3-3: The degree of influence [1]

Influence	**Degree of Influence**
None	0
Insignificant	1
Moderate	2
Average	3
Significant	4
Strong significant	5

IFPUG-FPA [79] is widely used for calculating software's functional size and complexity based on user requirements. AFP can be used as an input to estimate the effort. The efforts in terms of IFPUG-FPA is equal to AFP multiplied by PDR.

$$Effort_{\text{IFPUG-FP}} = AFP \times PD \quad (4)$$

3.2 Data Collection

The thesis incorporates the ISBSG (version R1/2020) [7] and other datasets as valuable historical datasets for the research study. By utilising these datasets, the study aims to leverage the wealth of data and insights available within the ISBSG and other datasets to contribute to the effort estimation methodologies.

3.2.1 Selection of FPA Dataset (ISBSG)

The ISBSG dataset contains 9,592 finished software projects. There are a total of 251 documented attributes. These are separated into a variety of categories, such as Summary Work Effort (SWE), total effort in hours recorded against the project; the adjusted functional point (AFP) of the project at the final count; VAF, the adjustment to the function points that take into account various technical and quality characteristic; functional point variables (EI, EO, EQ, EIF, ELF), PDR, and other categorical variables. Furthermore, there are many categorical variables in the ISBSG dataset (see *Table 2-2*). Clarifying these variables might be impossible in this study, selecting the representative variables using a survey to investigate recent studies that used categorical variables.

Based on that survey, the Industry Sector (IS) is the most studied among the above-mentioned categorical variables. Moreover, Relative Size (RS) is the most recent study from 2018-2020. This research proposes a novel approach to estimate the SDEE based on IS, RS and EI, EO, EQ, EIF, and ILF. The selection of IS and RS from practical time limitations prompted an in-depth analysis of their influence on effort estimation. This focused approach neither dismisses nor diminishes the relevance of other variables. Instead, it lays the foundation for further investigation into a broader spectrum of categorical variables. Consequently, this research emphasizes the significance of IS and RS while acknowledging the potential for a more expansive exploration in the future.

3.2.2 Other Datasets

To diversify the dataset and enhance the prominence and robustness of the evaluation outcomes, in addition to utilizing ISBSG, this study also incorporates similar datasets based on function points from the PROMISE repository [104]. They include Albrecht, Desharnais, Kitchenham, and China datasets. These supplementary datasets enrich the research scope, providing a multi-faceted perspective on the performance of MLR, RF, DLMLP, ensemble, and transfer

learning techniques. The brief information on those datasets is presented in Table 3-4. Their attributes are illustrated in Table 3-5.

Table 3-4: Brief information on other datasets studied in this thesis

No	Dataset	Source	No.features	No.records	Effort unit
1	Desharnais	[105]	12	81	Person-hours
2	Albrecht	[102]	8	24	Person-hours
3	Kitchenham	[106]	10	145	Person-hours
4	China	[107]	19	499	Person-hours

- Desharnais dataset: This was introduced in J.M. Desharnais' master thesis [105] and is publicly available in the Promise repository. It has 81 software projects, which were collected from 10 organisations in Canada between 1983 and 1988. It has twelve features: Effort, PointsNonAdjust, Adjustment, PointsAjust, etc.
- Albrecht dataset [102] includes information on IBM software projects made in the 1970s. The Albrecht dataset has eight features: Input (EI), Output (EO), Inquiry (EQ), File (EIF/ILF), Effort, etc.
- Kitchenham dataset: The Kitchenham dataset [106] is a well-known dataset commonly used in software engineering research. The dataset consists of information collected from various software projects and includes attributes such as AFP, Effort, etc.
- China dataset: The China dataset [107] consists of 499 projects obtained from various companies in China. It contains 19 recorded attributes: Input (EI), Output (EO), Enquiry (EQ), File (EIF), Interface (ILF), Effort, etc. This dataset was made publicly available in 2010.

Table 3-5: Attributes of other datasets

No	Dataset	Attributes
1	Desharnais	TeamExp, ManagerExp, YearEnd, Length, Effort, Transactions, Entities, PointsNonAdjust, Adjustment, PointsAjust, Language
2	Albrecht	Input (EI), Output (EO), Inquiry (EQ), File (EIF), FPAdj, RawFPCount, AFP, Effort
3	Kitchenham	duration, AFP, Estimate, SWE
4	China	AFP, Input (EI), Output (EO), Enquiry (EQ), File (EIF), Interface (ILF), Added, Changed, Deleted, Resource, Duration, AdjFactor, SWE

3.3 Preprocessing Techniques

3.3.1 ISBSG Dataset

The ISBSG dataset includes various attributes, such as Project Rating, Development Type, Productivity, Industry Sector, Relative Size, and more. In order to ensure that this dataset offers high-quality data valuable for training models, it should be filtered based on the following criteria:

- The Project Rating field is designated with an ISBSG rating code of A, B, C, or D. As mentioned in ISBSG and several publications, the study chose high-quality projects by exclusively considering data projects with A and B ratings. This action led to the number of projects being reduced to 8,619.
- EI, EO, EQ, ILF, ELF, and industry sector, relative size; we have excluded all those not counted, resulting in 1,654.
- Productivity rate values (PDR) that fall outside of the Q1 (first quartile) - 1.5 × IQR to Q3 (third quartile) + 1.5 × IQR range may be eliminated, where IQR is the abbreviation of the InterQuartile Range. As a result, the final number of projects is 1,073 projects. Figure 3-2 presents the boxplot of the productivity rate before and after removing the outlier. The boxplot of factors of PFA, such as SWE, AFP, EI, EO, EQ, ILF, and ELF, is also illustrated in Figure 3-3 before and after removing the outlier based on the productivity rate. The number of projects for each category of Relative Size and Industry Sector is presented in Figure 3-4 after removing the outlier based on productivity rate.
- Categorical variables are often transformed into numeric labels to facilitate efficient processing and analysis. Various widely used techniques might be employed in Python for converting categorical variables into numerical form. Notable among these are the LabelEncoder and one-hot encoding methods. The LabelEncoder library, for instance, operates by assigning a distinct integer value to each category within the input variable. This approach retains the dimensionality of the data, which can be beneficial in situations where preserving the original feature space is essential. After the data filtration process, the encoding outcomes for Relative Size and Industry Sector are illustrated in Table 3-6 and Table 3-7, respectively.

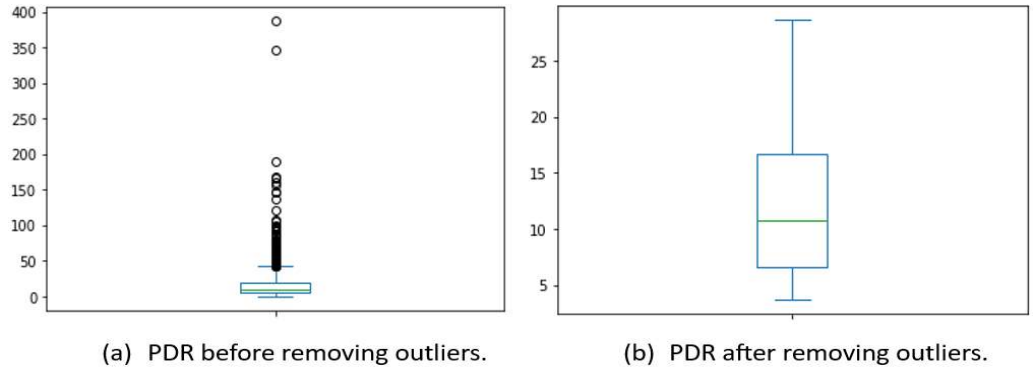
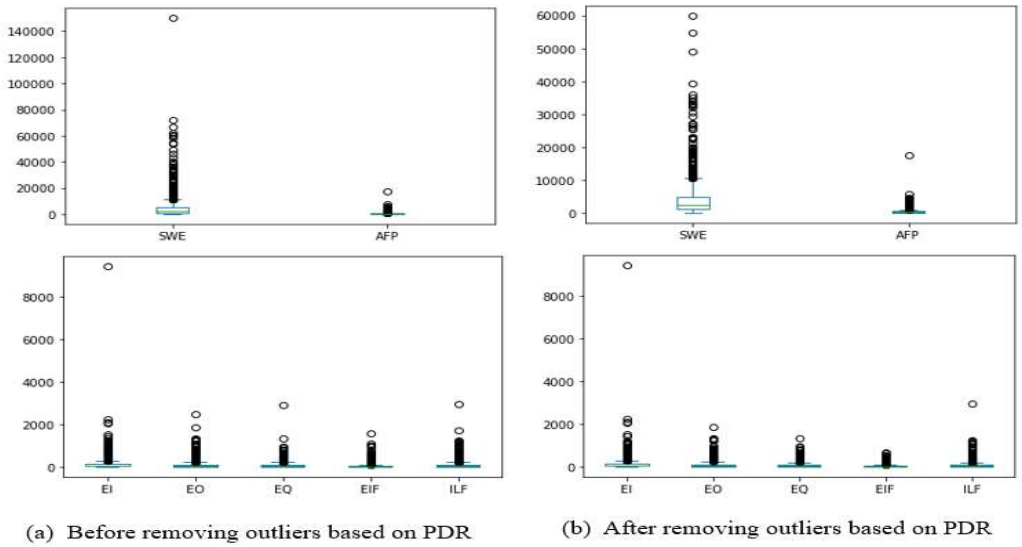


Figure 3-2: Box-plot of productivity rate before and after removing outliers



(a) Before removing outliers based on PDR (b) After removing outliers based on PDR

Figure 3-3: Box-plot of factors of FPA before and after removing outliers based on productivity rate

Table 3-6: Label encoding for Relative Size

No	RS	RS Label
1	L	0
2	M1	1
3	M2	2
4	S	3
5	XL	4
6	XS	5
7	XXL	6
8	XXS	7

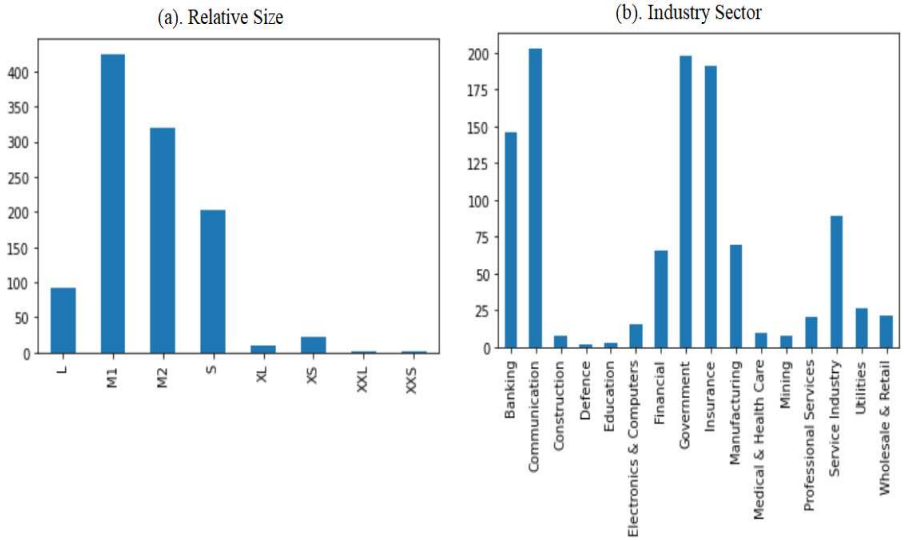


Figure 3-4: The number of selected projects in each RS and IS

Table 3-7: Label encoding for Industry Sector

No	IS	IS Label
1	Banking	0
2	Communication	1
3	Construction	2
4	Defence	3
5	Education	4
6	Electronics & Computers	5
7	Financial	6
8	Government	7
9	Insurance	8
10	Manufacturing	9
11	Medical & Health Care	10
12	Mining	11
13	Professional Services	12
14	Service Industry	13
15	Utilities	14
16	Wholesale & Retail	15

- Moreover, the counting methods developed by the IFPUG for FPAs are essential to this investigation. Therefore, out of 1,073 projects, 1045 belong to the IFPUG category, referred to as Dataset 1 and used primarily for thesis study. The remaining projects fall under the NESMA category (Dataset 2), which is utilized to evaluate the effectiveness of the transfer learning approach.

Table 3-8: Division of ISBSG dataset based on the Counting Approach

No	Dataset	Counting Approach	No. Records
1	Dataset 1	IFPUG	1045
2	Dataset 2	NESMA	28

3.3.2 Other Datasets

As mentioned in 3.2.2, besides the ISBSG dataset presented above, the study expands its analysis to incorporate other datasets, including Desharnais, Albrecht, Kitchenham, and China. The primary objective for these additional datasets is to evaluate the effectiveness of MLR, RF, DLMLP, and ensemble models and use them to clarify the performance of the transfer learning approach. Thus, the initial step involves identifying and selecting key features significantly influencing the model's performance.

A Pearson correlation analysis is conducted on those datasets to identify the key features significantly influencing the actual/effort values. This comprehensive examination aims to uncover the interrelationships between input

factors and the corresponding effort required in software development projects. By quantifying the strength and direction of linear associations, the analysis provides valuable insights into which features exert the most pronounced impact on effort estimation accuracy.

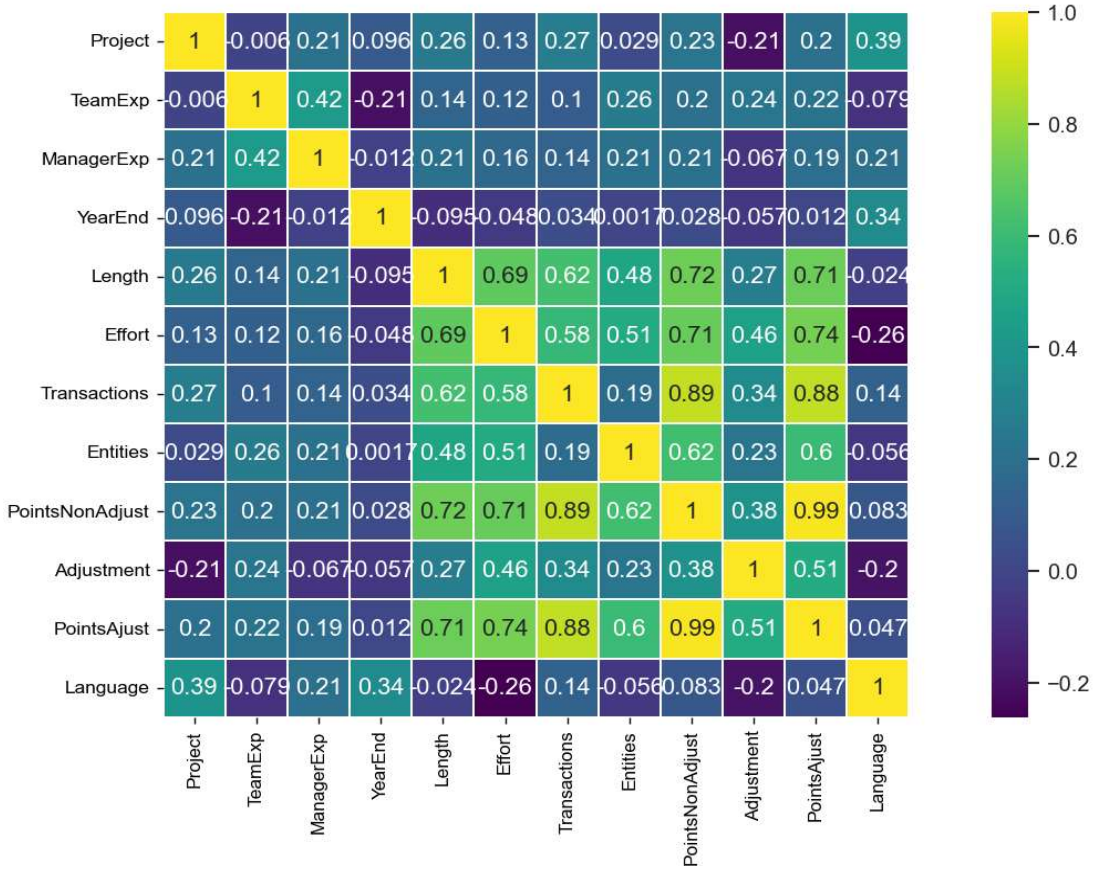


Figure 3-5: The Pearson correlation of features on the Desharnais dataset

The obtained results, depicted in Figure 3-5, Figure 3-6, Figure 3-7, and Figure 3-8, reveal crucial insights into the relationship between the attributes and their relevance for effort estimation. Using a threshold greater than 0.5 is a strategic approach to discern the most influential attributes that significantly impact the accuracy of effort estimation models. This thresholding technique allows us to focus on attributes with stronger correlations, effectively filtering out less impactful factors and streamlining the feature selection process.

In the case of Desharnais, the attributes of Length, Transactions, Entities, and PointsAdjust exhibit a positive impact on the effort required. Given the high correlation coefficients observed, particularly with PointsAjust at 0.74, it is determined that PointsNonAdjust provides redundant information and, therefore, has been excluded from further analysis. For Albrecht, the attributes of Input, Output, Inquiry, File, RawFPCount, and AdjFP affect the effort estimation significantly. Furthermore, in the context of Kitchenham, the duration, AFP, and Estimate attributes hold considerable importance, while for China, the attributes of AFP, Input, Output, Enquiry, File, and Added positively influence the actual

development efforts. These findings provide valuable guidance for accurately estimating software effort by considering the influential attributes in each dataset.

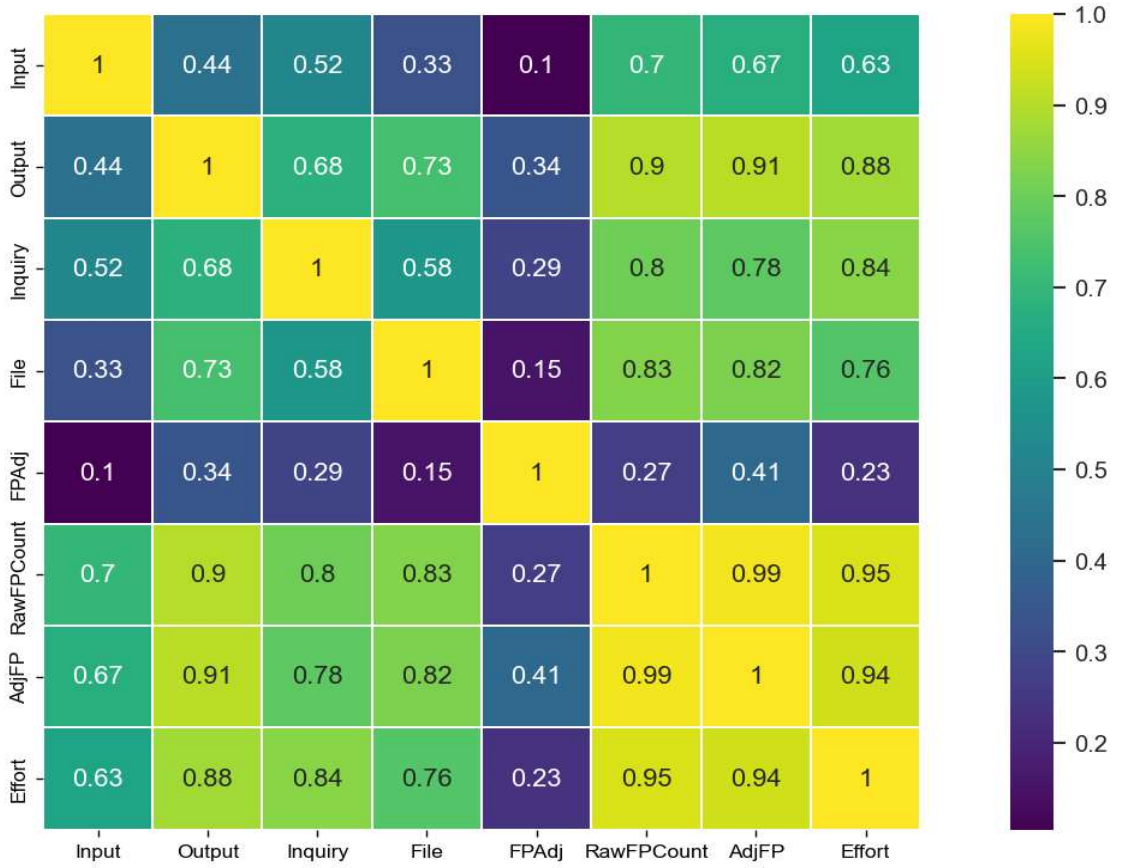


Figure 3-6: The Pearson correlation of features on the Albrecht dataset

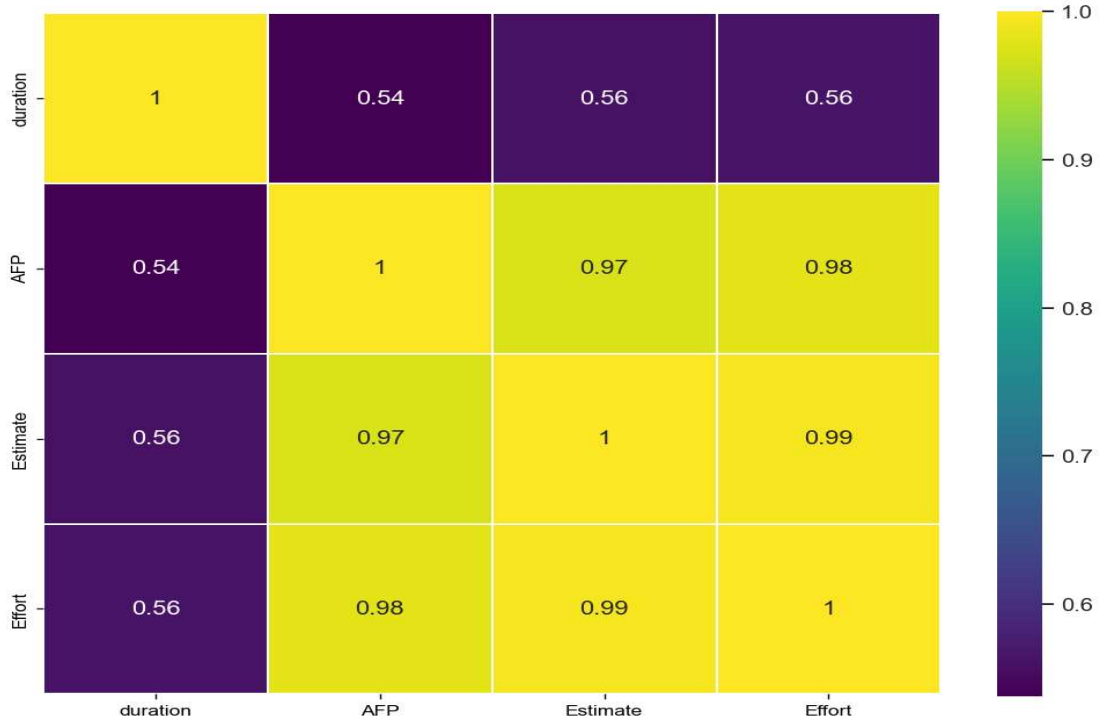


Figure 3-7: The Pearson correlation of features on the Kichenham dataset

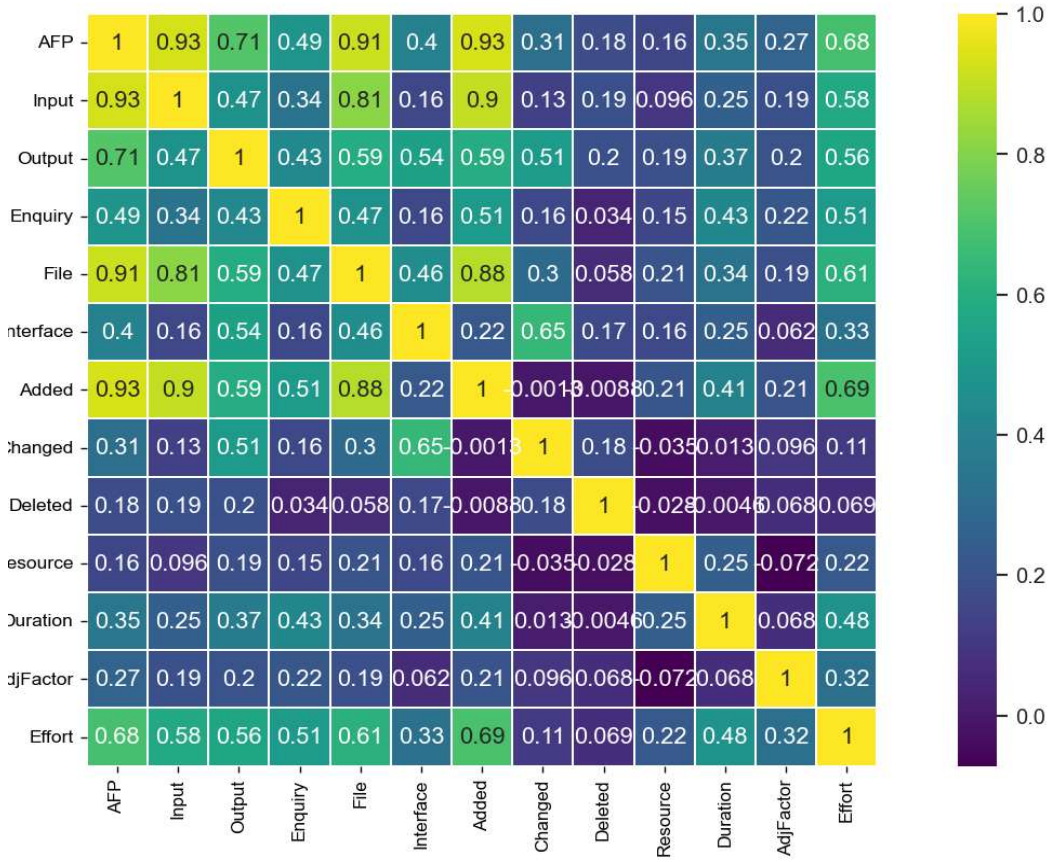


Figure 3-8: The Pearson correlation of features on the China dataset

3.3.3 Dataset Description

The datasets are divided into two segments to showcase the experimental results of the studied models and evaluate their effectiveness in effort estimation. For small datasets, 20% is designated for testing to ensure meaningful evaluation. For larger datasets, allocating 15% for testing and reserving the remainder for training enables the model to leverage ample data for robust training. Consequently, datasets with limited data, such as Desharnais, Albrecht, and Dataset 2, are assigned a 20% test allocation, while Dataset 1, Kitchenham, and China datasets receive an 85% training allocation. Detailed descriptions of each data type are provided in accompanying tables, outlining the key attributes and their ranges, ensuring a clear understanding of the dataset's composition.

Table 3-9: Data description of training – Dataset 1

	SWE	AFP	EI	EO	EQ	ILF	EIF
mean	4,081.72	421.99	125.95	95.35	71.29	83.42	28.14
std	5,765.54	792.75	361.07	162.24	120.97	160.19	59.67
min	64.00	9.00	0.00	0.00	0.00	0.00	0.00
0.25	1,053.00	116.00	21.00	14.00	6.00	14.00	0.00
0.50	2,391.00	225.00	55.00	42.00	27.00	41.00	5.00

0.75	4,691.00	459.00	126.50	104.50	86.50	89.50	30.00
max	59,809	17,518	9,404	1,831	1,306	2,955	644

Table 3-10: Data description of testing – Dataset 1

	SWE	AFP	EI	EO	EQ	ILF	EIF
mean	4,796.13	514.46	150.12	109.94	84.20	111.52	37.51
std	6,221.87	784.25	288.92	177.07	131.85	186.31	79.18
min	38.00	9.00	0.00	0.00	0.00	0.00	0.00
0.25	1,148.25	116.00	24.75	17.75	9.75	15.00	0.00
0.50	2,330.50	259.00	65.50	42.00	33.00	45.00	11.00
0.75	5,706.75	485.50	145.50	129.00	93.25	107.25	42.25
max	39,358	5,684	2,221	1,337	820	1,252	634

Table 3-11: Data description of training – Dataset 2

	SWE	AFP	EI	EO	EQ	ILF	EIF
mean	22.00	22.00	22.00	22.00	22.00	22.00	22.00
std	2,915.77	518.91	175.00	177.36	44.82	109.14	24.95
min	412.00	43.00	0.00	15.00	0.00	0.00	0.00
0.25	1,707.75	149.25	13.00	72.50	0.75	22.75	0.00
0.50	1,852.98	273.00	93.50	104.00	9.50	42.50	11.00
0.75	2,545.50	589.75	245.66	164.55	100.74	146.97	27.25
max	3,831.25	471.25	151.25	261.00	19.00	105.25	35.47

Table 3-12: Data description of testing – Dataset 2

	SWE	AFP	EI	EO	EQ	ILF	EIF
mean	2,202.17	150.50	29.00	85.33	3.83	23.50	8.00
std	981.21	56.23	37.22	36.16	4.40	24.44	10.06
min	951.00	67.00	0.00	25.00	0.00	0.00	0.00
0.25	1,636.25	121.00	6.50	72.50	0.75	14.00	0.00
0.50	1,986.00	157.00	16.00	90.00	3.50	14.50	3.50
0.75	2,931.25	197.50	33.00	106.00	4.00	24.75	16.75
max	3,524.00	226.00	100.00	129.00	12.00	70.00	21.00

Table 3-13: Data description of training – Desharnais

	SWE	Length	Transactions	Entities	PointNon Adjust	PointsAjust
mean	4667.06	11.47	180.23	116.63	296.86	282.71
std	4336.56	7.71	151.49	84.52	191.64	198.12
min	546.00	1.00	9.00	7.00	73.00	62.00
0.25	2282.00	6.00	88.00	52.00	167.00	140.00
0.50	3542.00	9.00	139.00	89.00	258.00	241.00
0.75	5817.00	13.00	223.00	145.00	377.00	340.00
max	23940.0	39.00	886.00	387.00	1127.00	1127.00

Table 3-14: Data description of testing – Desharnais

	SWE	Length	Transactions	Entities	PointNon Adjust	PointsAjust
mean	6587.00	12.43	189.81	145.50	335.31	315.75
std	4554.40	6.30	112.44	85.05	123.44	124.96
min	840.00	4.00	58.00	34.00	92.00	86.00
0.25	3368.75	8.00	111.50	98.25	261.00	227.00
0.50	5127.50	12.50	170.00	122.50	327.50	320.50
0.75	9691.50	15.50	244.50	181.00	436.50	426.00
max	14987.0	27.00	451.00	332.00	507.00	507.00

Table 3-15: Data description of training – Albrecht

	SWE	Input	Output	Inquiry	File
mean	24.116	44.526	48.895	17.789	19.789
std	31.268	40.218	37.962	21.283	16.592
min	2.900	7.000	12.000	0.000	5.000
25%	7.050	25.000	18.000	2.000	6.500
50%	11.800	40.000	38.000	13.000	15.000
75%	18.650	46.500	65.000	20.500	29.000
max	105.200	193.000	150.000	75.000	60.000

Table 3-16: Data description of testing – Albrecht

	SWE	Input	Output	Inquiry	File
mean	13.36	24.00	41.00	13.40	8.20
std	11.38	11.81	23.78	9.63	3.70
min	0.50	10.00	15.00	1.00	3.00
25%	7.50	15.00	20.00	6.00	6.00
50%	8.90	27.00	41.00	16.00	9.00
75%	21.10	28.00	60.00	20.00	11.00
max	28.80	40.00	69.00	24.00	12.00

Table 3-17: Data description of training – Kitchenham

	SWE	duration	AFP	Estimate
mean	3,390.35	206.74	528.43	3,018.40
Std	10,635.36	141.35	1,687.21	7,504.70
min	219.00	37.00	15.36	121.00
25%	874.00	114.00	121.52	900.00
50%	1,584.00	166.00	240.84	1,770.00
75%	2,972.00	259.00	464.00	2,895.00
max	113,930.00	946.00	18,137.48	79,870.00

Table 3-18: Data description of testing – Kitchenham

	SWE	duration	AFP	Estimate
mean	1954.67	205.25	524.47	2177.25
Std	1906.19	100.28	352.65	1809.82
min	286.00	40.00	74.40	200.00
25%	813.50	142.25	178.77	862.75
50%	1316.00	193.50	535.14	1587.00
75%	2586.00	238.75	796.19	2805.75
max	8656.00	432.00	1292.56	8690.00

Table 3-19: Data description of training – China

	SWE	Input	Output	Enquiry	File	Added
mean	3,876.23	148.17	107.93	60.89	86.88	342.86
std	6,354.29	262.42	216.00	104.55	172.35	609.13
min	26.00	0.00	0.00	0.00	0.00	0.00
25%	712.50	25.00	12.00	6.00	10.00	36.50
50%	1,801.00	60.00	41.00	24.00	32.00	128.00
75%	3,807.50	145.00	108.00	67.00	79.50	323.00
max	49,034.00	2,221.00	2,455.00	952.00	1,732.00	4,943.00

Table 3-20: Data description of testing – China

	SWE	Input	Output	Enquiry	File	Added
mean	4,091.27	238.97	135.15	64.32	107.77	426.81
std	6,970.38	934.65	240.14	109.14	315.91	1,380.06
min	117.00	0.00	0.00	0.00	0.00	0.00
25%	661.75	32.00	19.00	10.00	14.00	44.75
50%	1,993.00	71.50	42.00	28.50	41.50	158.50
75%	3,845.25	162.00	136.00	75.50	87.75	332.25
max	54,620.00	9,404.00	1,241.00	772.00	2,955.00	13,580.00

3.3.4 Balancing Dataset Technique

Several common approaches might be adopted to tackle the problem of imbalanced training datasets in deep learning based on data level. They might be as follows:

- Data resampling: this approach involves updating the distribution of the training dataset by either oversampling the minority category or undersampling the majority category [60]. Oversampling techniques include duplication of minority samples and generating synthetic data using techniques such as the Synthetic Minority Over-sample Technique (SMOTE) [62] and the Adaptive Synthetic Sampling Approach (ADASYN)

[108]. On the other hand, undersampling involves reducing the number of samples from the majority category to match the minority category.

- Class weighting: this approach might assign weights to different categories during training [61]. By assigning higher weights to the minority category, the model might be encouraged to pay more attention to these samples and reduce the bias towards the majority category.
- Generating Augmented Data: data augmentation is the process of generating synthetic data that share characteristics with the original dataset while incorporating purposeful differences or alterations [109], [110]. This procedure comprises applying particular methods or modifications to the current data. The main goal of creating augmented data is to increase the dataset's diversity, scalability, and representativeness, which helps machine learning models perform better and generalise across various applications. Gaussian combination model (GMM) [111] might be used to generate those datasets. It assumes that the provided data points represent samples [110].

The ISBSG dataset encompasses the industry sector feature, which is crucial in the analysis. As depicted in Table 3-7, the dataset comprises sixteen distinct industry sectors: Banking, Government, Financial, and others. However, it is essential to note that the distribution of projects across these industry sectors is imbalanced, as indicated in Figure 3-4. Consequently, this section aims to investigate the performance of the deep learning model when applied to a balanced dataset. The class weighting approach is utilised specifically for the industry sector feature to achieve this. By assigning appropriate weights to each category within the industry sector, the deep learning model might effectively account for the inherent class imbalance, leading to more accurate and reliable predictions across different industry sectors. The following diagram of this approach is given in Figure 3-9. Dataset 1 serves as the historical dataset employed in this methodology.

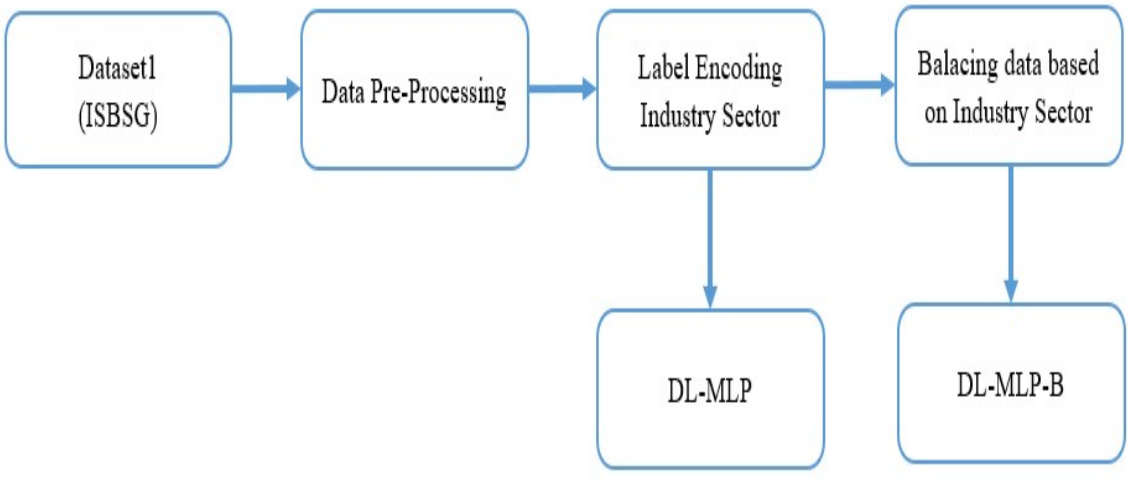


Figure 3-9: The architecture of the DLMLP model with/without balancing based on industry sector factors

3.4 Model Development

This section presents the model development and the thesis study models based on multiple linear regression, deep learning, transfer learning, deep learning with balancing datasets, and ensemble model, which incorporates multiple linear regression and deep learning.

3.4.1 Multiple Linear Regression Model

The MLR technique is employed for statistical analysis to establish the connection between a dependent and two or more independent variables. Multiple regression aims to predict the dependent variable's value based on the independent variables' value [23]. In the MLR model, the dependent variable is commonly denoted as the response or outcome variable, whereas the independent variables are termed predictor variables or covariates. It might be used to predict software effort estimation based on a given set of independent variables. The formulation of MLR involves an equation that expresses a direct association between a dependent variable and a set of p independent variables $X_1, X_2, \dots, X_p$ as follow:

$$y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon \quad (5)$$

where y is the response variable, it stands for the output of the model; $X_1, X_2, \dots, X_p$ are predictors or independent variables; β_0 is an intercept, $\beta_1, \beta_2, \dots, \beta_p$ are regression coefficients, and ε is presented as an error residual. The intercept and regression coefficients are unknown values. The regression model estimates these coefficients based on the observed data, and the goal is to find the values of the coefficients that best fit the data.

Assuming that there are n records in the dataset, where each record i includes a value for the dependent variable y_i values for p independent variables $X_{\mathrm{i}1}, X_{\mathrm{i}2}, \dots, X_{ip}$, the multiple linear regression equation for record i can be expressed as:

$$y_i = \beta_0 + \beta_1 X_{\boldsymbol{i1}} + \beta_2 X_{\mathrm{i}2} + \cdots + \beta_p X_{ip} + \varepsilon_i, i = \overline{1..n} \quad (6)$$

Equation $y_i = \beta_0 + \beta_1 X_{\boldsymbol{i1}} + \beta_2 X_{\mathrm{i}2} + \cdots + \beta_p X_{ip} + \varepsilon_i, i = \overline{1..n}$ (6) could be written as follows:

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \quad (7)$$

If the inverse $(\mathbf{X}^{\boldsymbol{T}}\boldsymbol{X})^{-\mathbf{1}}$ exists, the values of the coefficients that minimise the sum of squared differences across all n records might be then estimated using the least squares method [112]. As a result, vector $\boldsymbol{\beta}$ is given by:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} X_{11} & \cdots & X_{1q} \\ \vdots & \ddots & \vdots \\ X_{n1} & \cdots & X_{nq} \end{pmatrix} \times \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_q \end{pmatrix} \quad (8)$$

$$\boldsymbol{\beta} = (\boldsymbol{X}_{\boldsymbol{nq}}^{\boldsymbol{T}}\boldsymbol{X}_{\boldsymbol{nq}})^{-\mathbf{1}}\boldsymbol{X}_{\boldsymbol{nq}}^{\boldsymbol{T}}\boldsymbol{y} \quad (9)$$

Moreover, singular value decomposition (SVD) [113] is a matrix factorisation technique used in linear algebra. It decomposes a matrix into three separate matrices, providing valuable insights into the properties and structure of the original matrix. For a given matrix X_{nq}, the SVD might be expressed as:

$$X_{nq} = U_{nm}\Sigma_{mq}V_{qq}^{T} \quad (10)$$

where U_{nm}, V_{qq} are orthogonal matrices, $U^T U = I$, $V^T V = I$, and Σ is a diagonal matrix containing the square roots of eigenvalues from U and V in descending order [113]. Vector $\boldsymbol{\beta}$ is given as follows:

$$\boldsymbol{\beta} = \mathbf{V}_{\boldsymbol{qq}}(\boldsymbol{\Sigma}_{\boldsymbol{mq}}^{\boldsymbol{T}}\boldsymbol{\Sigma}_{\boldsymbol{mq}})^{-\mathbf{1}}\boldsymbol{\Sigma}_{\boldsymbol{mq}}^{\boldsymbol{T}}\boldsymbol{U}_{\boldsymbol{nm}}^{\boldsymbol{T}}\boldsymbol{y} \quad (11)$$

3.4.2 Random Forest

Random Forest (RF), introduced in 2001 by Breiman [114], is a kind of ensemble of decision trees trained via the bagging method (or sometimes the pasting method). Several poor models are joined to build a superior model. Each tree categorises the attributes of a new entity. The forest chooses the category with the most votes and averages the outputs of the different trees. The growth process of each tree in a random forest, as described in [115], [116], can be summarised as follows:

- Sampling: N cases are randomly selected from the original data with replacements to form the training set for each tree. The number of cases in the training set is equal to N.
- Variable Selection: At each tree node, a subset of m variables is chosen randomly from the total M input variables. The value of m is much smaller than M. The node is then split based on the best split determined using the selected m variables.
- Constant Variable Selection: Throughout growing the random forest, the value of m remains constant for all the trees.
- Maximum Growth: Each tree is grown to its fullest extent without the use of any pruning techniques.

To summarise, the growth process of each tree in a random forest involves sampling cases with replacement, selecting a subset of variables at each node, constant variable selection across all trees, and allowing each tree to grow to its maximum extent without pruning. According to Mustapha et al. [117], it outperformed several other classification models and was also resistant to over-fitting and relatively user-friendly [118].

3.4.3 Gradient Boosting

Boosting involves adding new models to an existing ensemble in a systematic manner, as proposed by Leo Breiman [114]. At each iteration, a new weak learner model is trained by considering the errors of the ensemble learned so far. Boosting algorithms were originally entirely algorithm-driven, but later, a statistical framework was developed for boosting methods, such as the gradient boosting machine [119]–[121]. This approach involves sequentially fitting new models to improve the accuracy of the response variable. The idea is to construct new base learners most similar to the negative gradient of the loss function, which connects to the entire ensemble. This learning process minimises the traditional squared error loss function by iteratively fitting errors. Extreme gradient boosting (XGBoost) and Histogram Gradient Boosting (HGBoost) are both implementations of gradient boosting, a machine-learning technique employed for predictive modelling.

XGBoost is a widely used gradient-boosting implementation incorporating a gradient-boosting framework with various optimizations to enhance speed and precision. This algorithm is an ensemble of gradient boosting that takes advantage of second-order derivatives of the loss function to identify the most efficient and precise base classifier [122]–[124]. Unlike traditional gradient boosting, XGBoost employs second-order gradients. XGBoost supports three primary forms of gradient boosting [125] with different variations:

- Gradient Boosting Algorithm: XGBoost includes the traditional gradient boosting algorithm, also known as the gradient boosting machine. This algorithm incorporates a learning rate, which controls the contribution of each tree in the ensemble.
- Stochastic Gradient Boosting: XGBoost offers stochastic gradient boosting, introducing sub-sampling techniques at multiple levels. This approach includes sub-sampling at the row level (sampling a subset of data points), column level (sampling a subset of features), and column per split level (sampling a subset of features for each split). These techniques enhance diversity and reduce overfitting.
- Regularized Gradient Boosting: XGBoost provides regularized gradient boosting with L^1 (Lasso) and L^2 (Ridge) regularization. This regularization helps control the complexity of the model and prevent overfitting.

On the other hand, Histogram Gradient Boosting (HGBoost), also known as histogram-based gradient boosting, is another boosting ensemble that utilizes feature histograms to quickly and accurately identify the optimal splits [122], [126]. Compared to traditional gradient boosting, HGBoost is more efficient regarding processing speed and memory usage.

3.4.4 Multi-layer Perceptron Model

Deep learning (DL) is a specialized area within machine learning that involves using neural networks with multiple layers to acquire intricate data

representations, drawing inspiration from the structure/function of the human brain [127], [128]. It involves a machine learning algorithm class that learns these representations through non-linear transformations applied to the input data. These networks can be composed of different layers, such as convolutional, pooling, and recurrent layers, in addition to fully connected layers. DL improves prediction accuracy by replying on multiple processing layers to gain knowledge representations of the data with varying levels of complexity. According to [127], DL has advanced dramatically in various other disciplines, including natural language processing, visual object classification, and image recognition, and has achieved state-of-the-art performance in many applications.

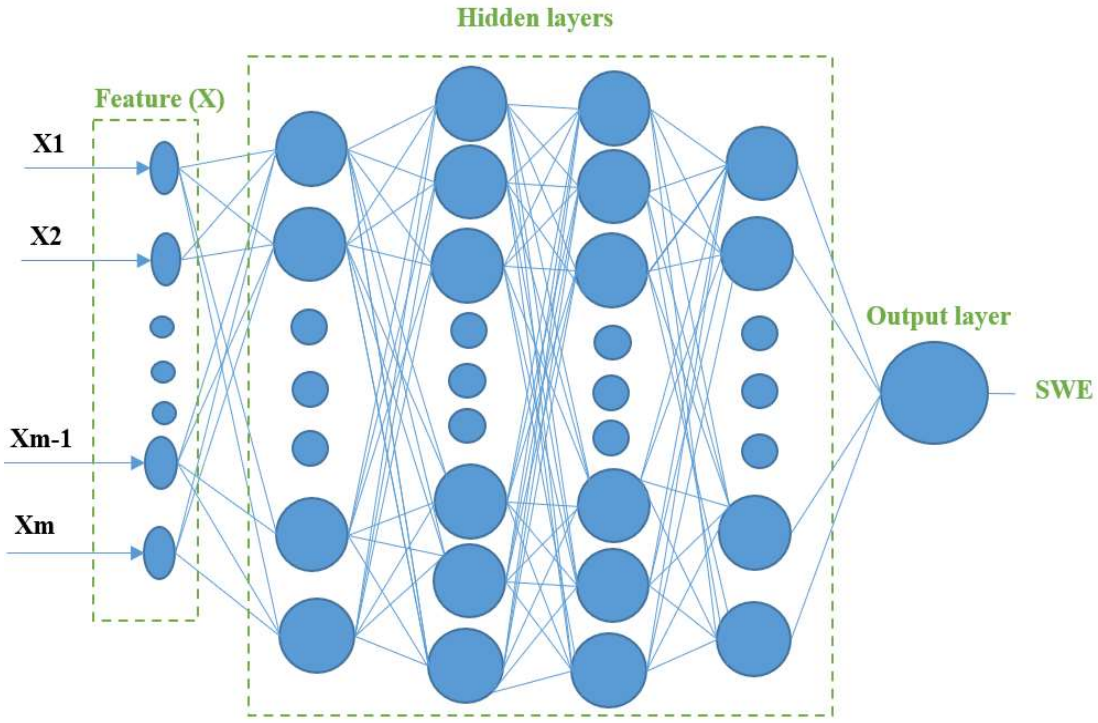


Figure 3-10: The diagram of deep learning with fully connected four hidden layers

DL aims to investigate complex systems in massive amounts of the dataset using backpropagation techniques to show how a machine adjusts the hyperparameters used to measure each class's representation depending on the performance of the preceding layer.

Figure 3-10 illustrates the flow diagram for deep learning with four fully connected layers involving an input layer, hidden layers where each neuron participates in a weighted linear summation, and an output layer that produces the network's final output. Each hidden layer involves a non-linear activation function applied to the previous layer's output. During training, the weights of connections between neurons are updated by computing the difference between predicted and actual outputs for each example in the training set.

The MLP is a supervised machine learning algorithm and foundational feedforward neural network architecture employed extensively in deep learning

research and applications. It comprises multiple layers of interconnected artificial neurons (perceptrons). It builds a network to simulate the human brain's processing [129]. The MLP trains on a dataset to learn the function $f: R^m \rightarrow R$, where m is the number of dimensions for input. Function f is used to learn the improved weight values corresponding to each network link to achieve a minor discrepancy between the estimated and actual values in terms of effort estimation.

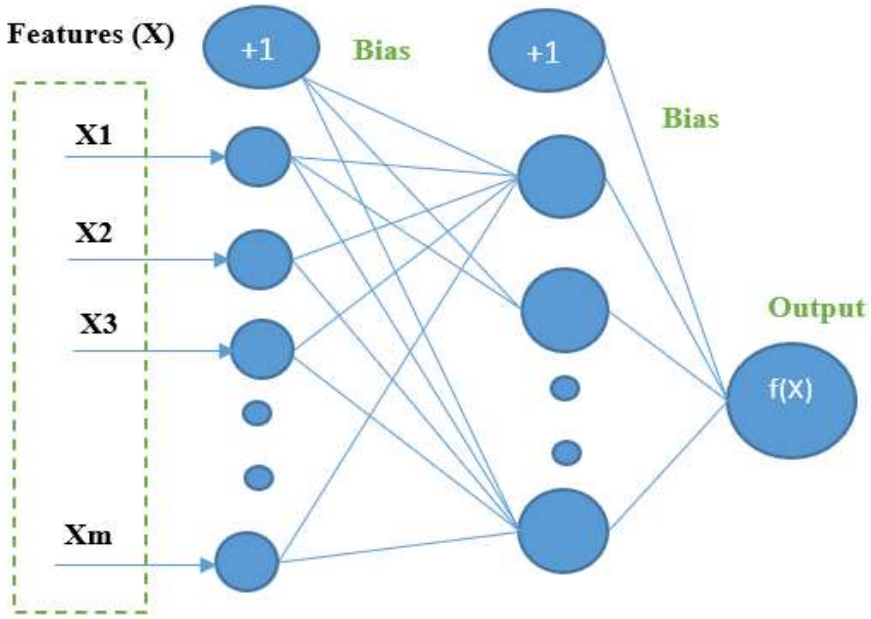


Figure 3-11: The diagram of one hidden layer of MLP

The MLP architecture comprises an input, output, and one or more hidden layers. The Input layer receives input data, which is subsequently propagated through the hidden layers to generate the output. Figure 3-11 shows the diagram of one hidden layer of MLP. The input layer, located on the left most layer, comprises a group of neuron features (X) that represent the input features. Every neuron in the hidden layer participates in a weighted linear summation $x_1w_1 + x_2w_2 + x_3w_3 + \cdots + x_mw_m$ to the values from the previous layer, followed by an activation function. The activation function used in each neuron can vary, but common choices include the sigmoid function, ReLU (rectified linear unit), and tanh (hyperbolic tangent) function [130], [131]. The values propagated from the preceding hidden layer are accepted by the output layer and transformed to produce output values.

3.4.5 Transfer Learning Technique

Transfer learning involves the notions of domain and task [132]. A domain consists of a marginal probability distribution over the feature space and the feature space itself. A collection of features found in a dataset may be described as a feature space (X). The marginal probability distribution $P(X)$ represents the marginal probability of a random variable in the presence of other random variables. Given all possible outcomes of a different random variable, the chance

of a current event is referred to as a marginal probability. Two domains that are distinct from one another could have separate feature spaces with various marginal probability distributions.

The source domain might be denoted as $D_S = \{(x_{S1}, y_{S1}), \dots, (x_{Sn}, y_{Sn})\}$, where $x_{Si} \in X_S$ is the data instance and $y_{Si} \in Y_S$ is the corresponding output variable. Similarly, the target domain is defined as $D_T = \{(x_{T1}, y_{T1}), \dots, (x_{Tn}, y_{Tn})\}$ where $x_{Ti} \in X_T$, and $y_{Ti} \in Y_T$. In most cases, $0 \leq Tn \leq Sn$. If X_S differs with X_T or $P(X_S)$ differs with $P(X_T)$, then the source domain differs from the target domain ($D_S \neq D_T$). On the other hand, a learning task is defined as a pair $T = \{y, P(Y|X)\}$. If $T_S \neq T_T$ so either $Y_S \neq Y_T$ or $P(Y_S|X_S) \neq P(Y_T|X_T)$.

According to Pan and Yang [132], traditional machine learning is learning where the target and the source domain are the same ($D_S = D_T$) and their learning task is the same. In the case of the feature spaces between the source (X_S) and target domain (X_T) are different or the marginal probability distributions between the source ($P(X_S)$) and target domain $P(X_T)$ are different, we state that the domains are different ($D_S \neq D_T$. Transfer learning can be classified into three primary types: inductive, transductive, and unsupervised transfer learning.

Inductive transfer learning involves leveraging machine learning techniques when the target task differs from the source task, regardless of the similarity between their respective domains. This approach makes it possible to train a model on a source task and then apply it to other tasks without requiring a complete retraining process [133]. Instead, partial retraining of specific layers or components may be employed. Notably, during the training process for a specific task, the model has the potential to acquire shared features from the data, which can help address other tasks effectively.

According to Arnold et al. [134] and Kocaguneli et al. [3], the source and target tasks are identical in transductive transfer learning, while the source and target domains are distinct. It involves using a pre-trained model to make predictions on a new dataset, and the predictions are used to train a new model [132]. This type of transfer learning is proper when there is no available labelled data for the target task, and the pre-trained model can generate pseudo-labels for the new dataset [135]. The new model is then trained on the pseudo-labelled data.

Lastly, in unsupervised transfer learning, the target task exhibits differences from the source task while maintaining a certain level of relevance or connection. It involves training a pre-trained model on an unsupervised task, such as autoencoders [136], and the learned features are used to initialise a new model. This type of transfer learning is advantageous when the target task has limited labelled data, and the pre-trained model can be used to transfer knowledge from the unsupervised task to the target task.

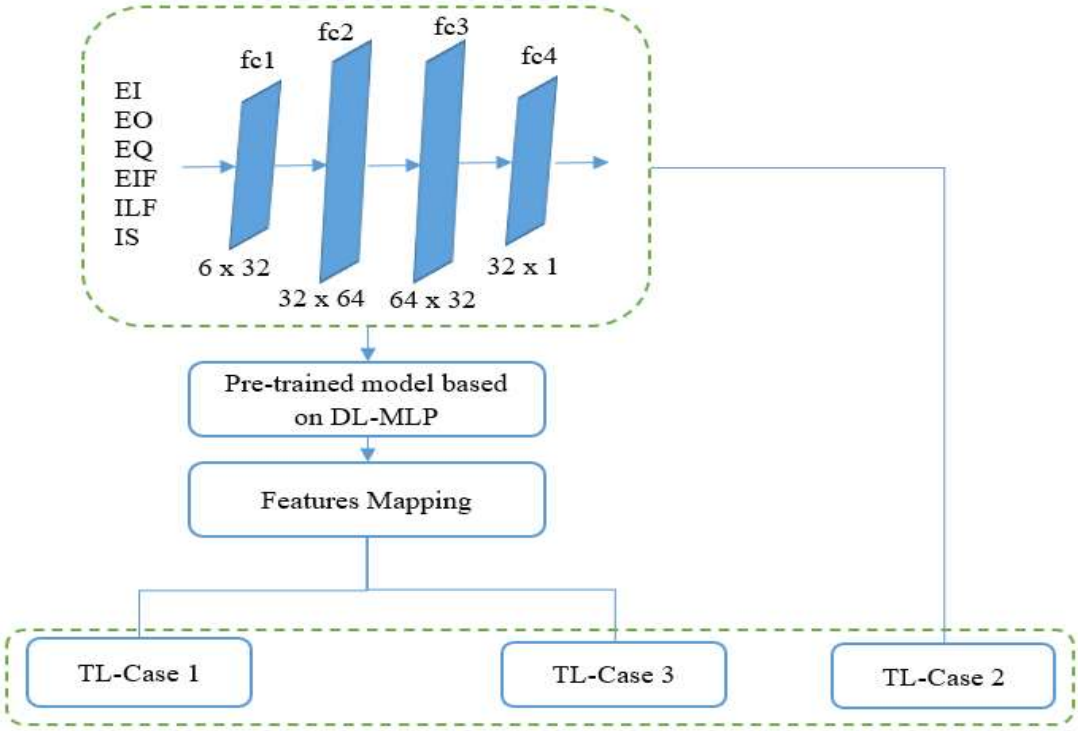


Figure 3-12: The diagram of the transfer learning model

This thesis uses Dataset 1 as the source and Dataset 2, Albrecht, and China as targets. Dataset 1 and Dataset 2 share the same input and output features, whereas the remaining datasets exhibit similarities in their features but have fewer input features than Dataset 1. Dataset 1 comprises a significantly more extensive set of 1045 projects than Dataset 2, which consists of only 28 projects. Additionally, when contrasting Dataset 1 with other datasets, such as Albrecht and China, it becomes evident that Dataset 1 is more significant than the others. As outlined in Section 1, using transductive transfer learning involves leveraging machine learning techniques in scenarios where the target task is similar to the source task. In this section, we intend to explore the application of transductive transfer learning by employing a pre-trained model trained on Dataset 1, incorporating the datasets above in the transfer learning process.

Figure 3-12 illustrates the diagram of transfer learning models, where Dataset 1 is used as an extensive dataset to build the pre-trained model, and Dataset 2, Albrecht, China, are used to clarify the performance of transfer learning models. Features mapping involves translating the characteristics of the new dataset into a format that the pre-trained model might understand. The pre-trained model was initially designed to work with six input features (EI, EO, EQ, EIF, ILF, and Industry Sector). Those features were chosen based on the best performance of effort estimation obtained from those features presented in 5.2.1. This step updates the pre-trained model's input layer to match the new input's size. The scenario is defined into three cases as described below:

- TL-Case 1: Using DLMLP models trained based on Dataset 1 to validate the performance of effort estimation based on a testing dataset of Dataset 2.
- TL-Case 2/DLMLP: Using DLMLP models, train them based on 80% of Albrecht, China, Dataset 2 and validate the performance of effort estimation based on 20% of those datasets.
- TL-Case 3: This is a transfer learning approach. DLMLP models trained by Dataset 1 are called pre-trained models and continue to train based on 80% of Albrecht, China, and Dataset 2 and validate the performance of effort estimation based on 20% of the remaining datasets.

3.4.6 Ensemble Model: Incorporating Multiple Linear Regression, Random Forest, and Deep Learning Models

Ensemble learning combines the predictions of multiple machine learning models to improve the accuracy and generalisation of the overall model. The idea behind ensemble learning is that by incorporating the predictions of multiple models, the variance and bias of the overall model might decline, leading to better performance on unseen data. In 1990, Hansen et al. [68] proposed that utilising an ensemble of neural networks with a majority agreement technique could produce better results than using a single predictor. In this context, an ensemble refers to a group of predictors, and ensemble learning is a method that integrates predictions from multiple models, referred to as the ensemble method. Bagging, boosting, and stacking are three popular types of ensemble methods, as noted in a publication [124].

- Bagging and pasting: a strategy that trains each predictor using the same training algorithm on distinct random subsets of the training set. The procedure is called bagging when sampling is performed with replacement; otherwise, (without replacement) is named pasting. Both bagging and pasting allow for numerous samples of training cases across multiple predictors. However, only bagging allows for various examples of the exact predictor. Once all predictors have been trained, the ensemble can forecast a new instance by aggregating all predictors' predictions.
- Boosting: Any ensemble method that may consolidate numerous ineffective learners into one robust learner is called boosting. Most motivating approaches require predictors to forecast sequentially, with each attempt to correct its predecessor. Adaptive Boost (AdaBoost) and Gradient Enhancement are the most common boosting methods.
- Stacking: David Wolpert [128] proposed in 1992, taking prior predictions as feed to determine the final prediction (blender/meta learner).
- Voting: Voting methods combine predictions from multiple models by taking the most votes (for classification) or averaging (for regression). There are different types of voting ensembles, such as hard voting, where

the class with the majority vote is chosen, and soft voting, where the class probabilities are averaged.

As mentioned in Section 2.5, an ensemble learning approach is employed to enhance prediction accuracy. Select two base regression models, MLR and RF, for creating ensemble models through stacking regressors. These ensemble models produce predictions, which are then integrated with the output of a DLMLP model using a voting-by-averaging method for regression. The performance of this approach is evaluated with MLR, RF, and DLMLP approaches.

3.5 Model Explainability - Interpretability

In software effort estimation, where precise predictions are pivotal for effective project planning and resource allocation, a pressing challenge arises from the black-box of the predicted models, especially DLMLP models, chosen in this section due to time constraints. These models leverage input features such as EI, EO, EQ, EIF, ILF, IS, and RS. However, their opacity makes it difficult for stakeholders to comprehend the inner workings of these models and how predictions are generated.

Explainability techniques, such as LIME and SHAP, become essential to address this issue. These techniques are pivotal in bridging this gap by unveiling the intricate relationships between these input features and the predicted effort. Doing so gives stakeholders a transparent view of the estimation process, enabling them to understand better the underlying factors influencing model predictions. This transparency enhances the estimation model's credibility and empowers decision-makers to make informed decisions regarding software project planning and resource allocation.

3.5.1 LIME

In the context of effort estimation using the LIME model explainability framework, assigning positive and negative values to independent variables indicates their influence on predicted effort. This numeric representation plays a crucial role in comprehending the impact of each variable on model predictions. LIME is a valuable tool for gaining insights into individual predictions generated by predictive models. Its fundamental purpose lies in approximating predictive models locally using interpretable models. LIME's primary objective is to address the fundamental question: 'Why did the model produce this specific prediction for a given instance?'

To illustrate, applying LIME in the context of effort estimation assists in illuminating how each feature (e.g., EI, EO, EQ) contributed to the predicted effort for a specific instance. LIME dissects the contributing factors underlying a prediction, facilitating an in-depth understanding of the role played by each feature in the model's decision-making process. The positive and negative values associated with the feature indicate their impact on the predicted effort.

- Positive Contribution: When a feature has a positive value, an increase in that variable's value tends to result in a higher predicted effort. For example, suppose variables such as EI, EO, EQ, etc., have positive contributions. In that case, it suggests that the effort required for software development is expected to increase as these features increase in function points.
- Negative Contribution: Conversely, when a feature has a negative value, an increase in that feature's value tends to lead to a lower predicted effort. For example, if EIF and ILF have negative contributions, it implies that as the function points of these features increase, the effort required for software development is expected to decrease.

3.5.2 SHAP

SHAP provides a unified approach to attribute the contribution of each feature to the predicted effort estimation. It assigns a value to each feature, representing its impact on the prediction in the context of the other features. These values are called SHAP values.

- Positive SHAP Value: A positive SHAP value for an independent variable signifies that the presence or increase in that variable contributes positively to the predicted effort. Higher values or complexity for EI, EO, EQ, etc., features are associated with increased effort.
- Negative SHAP Value: On the other hand, a negative SHAP value for a variable suggests that the presence or increase in that variable contributes negatively to the predicted effort. For features such as EIF and ILF, negative SHAP values indicate that higher values or complexity in these variables are associated with decreased effort in the effort estimation model.

By examining these SHAP values, analysts and practitioners might gain valuable insights into the relative importance and impact of different features on predicted effort. SHAP values provide a quantified understanding of each feature's influence and consider the intricate interactions and dependencies between these features.

It is worth noting that the sum of SHAP values across all features typically equals the difference between the model's prediction for a specific instance and the average prediction for all data points. This sum highlights the collective contribution of each feature in explaining the model's prediction for a particular case, further enhancing the grasp of feature importance in effort estimation.

3.6 Comparison Criteria

Several metrics might be used to validate the accurate performance of the proposed model compared with other models. These include the Magnitude of Relative Error (MRE) (12), MMRE (13), and other measures [137]. MMRE

measures the average magnitude of the relative errors between predicted and actual observed values. MAE stands for Mean Absolute Error, measuring the magnitude of the discrepancies between expected and actual results, making it an insightful metric to assess predictive accuracy. It is suitable for interpretability.

Moreover, the prediction level at x ($\mathrm{PRED}(x)$) (16) is considered further research criteria. It is a metric used to evaluate the accuracy of a predictive model's performance. It calculates the percentage of predictions within a specific error threshold "x". Although publications [138] stated that this metric might have some significant disadvantages, it is still widely used to validate the effort estimation accuracy [139].

However, [140] suggests not using these metrics because of their bias; further criteria are used to improve the experiment's efficiency. SA is a denotation of standardised accuracy (11), where $\overline{MAE_p}$ is the average value of an enormous number (typically 1000), runs of random guessing [141]; it was proposed by [140]. Except for $\mathrm{PRED}(x)$ and SA objectives are to be maximised; all remaining evaluation measures are minimised. In addition, Mean Balance Relative Error (MBRE) and Mean Inverted Balance Relative Error (MIBRE) data are considered as additional study criteria. MIBRE is especially beneficial when predicted values are near 0 since MBRE may become infinite. These criterion formulas are as follows:

$$MRE_i = \frac{|y_i - \hat{y}_i|}{y_i} \tag{12}$$

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \tag{13}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{14}$$

$$SA = \left(1 - \frac{MAE}{\overline{MAE_p}}\right) \times 100 \tag{15}$$

$$PRED(x) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} 1, if\ MRE_i \leq x \\ 0, otherwise \end{cases} \tag{16}$$

$$MBRE = \frac{1}{n}\sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{min\ (y_i, \hat{y}_i)} \tag{17}$$

$$MIBRE = \frac{1}{n}\sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{max\ (y_i, \hat{y}_i)} \tag{18}$$

where n is the number of measurements; y_i is the observed value, and $\hat{y}_i$ is the predicted value.

4. EXPERIMENTS

This section presents the conceptual framework of the thesis as well as the experiment of models such as MLR, RF, DLMLP, transfer learning, deep learning with balancing dataset, the ensemble by incorporating regression, random forest and deep learning, and model explainability.

4.1 Conceptual Framework of the Study

4.1.1 The Framework of the Study

As shown in Figure 4-1, there are four primary phases, including collecting the datasets, data preprocessing, building the proposed models, and measuring the performance of proposed models based on performance metrics.

In the first step, the thesis collects datasets. As mentioned in Section 3.2.1, the study mainly uses the ISBSG dataset released in 2020, and further study performance of studied models by using other datasets is detailed in Section 3.2.2.

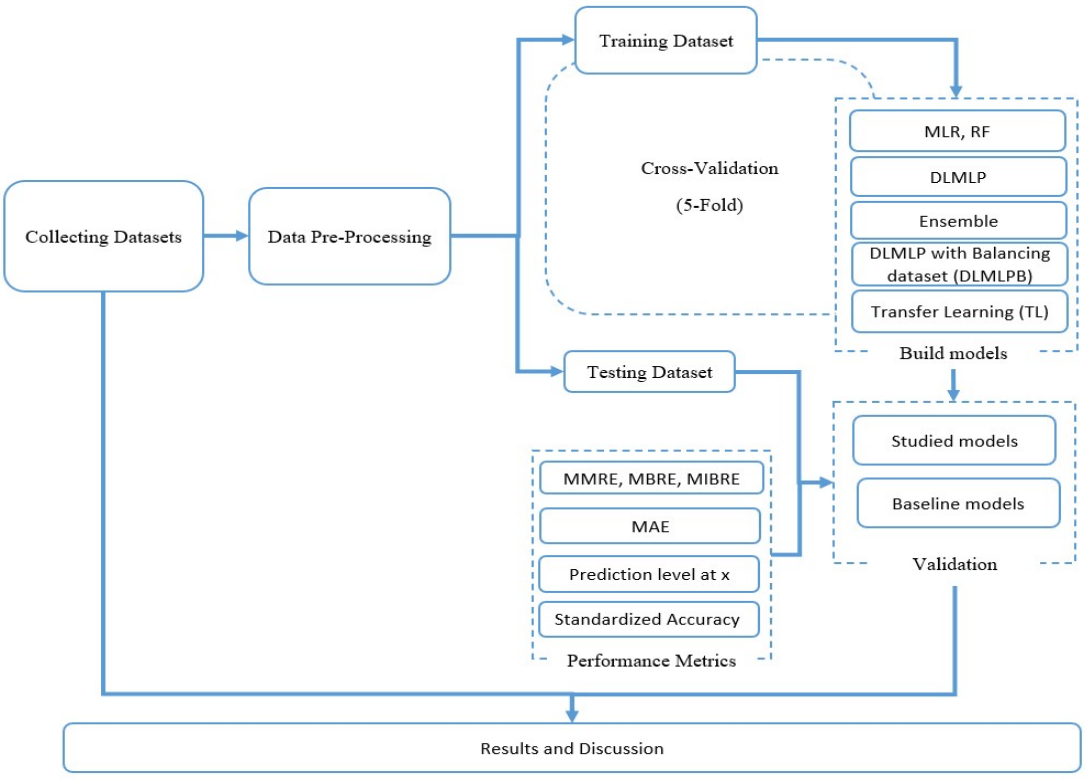


Figure 4-1: The flow diagram of the proposed software effort estimation

Next, data preprocessing is vital in preparing and refining the raw datasets before training models. The process of data preprocessing and the results of this

process are illustrated in Section 3.3. The following presents the steps of data preprocessing might summarized as:

- ISBSG dataset: The primary dataset employed in this study. This dataset includes factors relevant to FPA, such as EI, EO, EQ, EIF, ILF, AFP, and a range of categorical variables. The central focus of this thesis is to investigate the impact of categorical variables in conjunction with FPA factors on effort estimation. Due to time constraints, the study narrows its scope to six key predictors denoted as P1, P2, P3, P4, P5, and P6 (see Section 4.1.2). The data preprocessing for the ISBSG dataset is presented in Section 3.3.1. The study adheres to the IFPUG approach, creating two distinct datasets: Dataset 1 and Dataset 2. Dataset 1 is selected based on IFPUG criteria, while the remaining projects are allocated to Dataset 2 (see Table 3-8).
- Other datasets (Albrecht, Desharnais, Kitchenham, and China): In alignment with the research objectives for the Albrecht, Desharnais, Kitchenham, and China datasets, Pearson correlation analysis was conducted (see Section 3.3.2). This analysis aims to identify the key features significantly influencing actual/effort values in the context of software development projects. However, the ISBSG dataset serves as the primary dataset in this study. The purpose of the study is to influence categorical variables along with factors of FPA based on predictors (see Section 4.1.2), making the application of Pearson correlation analysis less suitable. Instead, the analysis for the ISBSG dataset focuses on categorical variable exploration and tailored data preprocessing methods, as detailed in the data preprocessing section.

Following that, the thesis studies proposed models, MLR (see Section 4.2), RF (see Section 4.3), and DLMLP (see Section 4.4). Further study is conducted on the proposed models, including ensemble (see Section 4.7), deep learning with balancing dataset (see Section 4.6) and deep learning with transfer learning (see Section 4.5). Ensemble models might combine multiple models, including MLR, RF, and DLMLP, to achieve better performance, while deep learning with transfer learning might leverage pre-trained models to improve learning efficiency and accuracy. Those models are trained based on the training datasets.

The study designs eleven predictors from P1 to P6, P_A, P_D, P_C, P_K, and $P_{Dataset2}$ (see Section 4.1.2). For models that adopted predictors P1 to P6, they use training Dataset 1 (see Table 3-9). The models adopted predictor P_A, P_D, P_C, P_K, and $P_{Dataset2}$ use training datasets of Albrecht (Table 3-15), Desharnais (Table 3-13), China (Table 3-19), Kitchenham (Table 3-17), and Dataset 2 (Table 3-11), respectively. The cross-validation with 5-fold is employed for all studied models in the training process. The performance of those models obtained from P1 to P6 is validated based on testing Dataset 1 (see Table 3-10). The performance of those models obtained from P_A, P_D, P_C, P_K, and $P_{Dataset2}$ use the corresponding testing datasets (see Table 3-16, Table 3-14, Table 3-20, Table 3-18, and Table 3-12).

Last but not least, the impact of parameter settings on the models on the accuracy of trained effort estimation techniques is widely acknowledged.

Nevertheless, determining the most suitable parameter values is challenging due to the many predictors considered in this study. A grid search [144] approach is introduced to address this challenge. This method systematically tests different parameter combinations and selects the one with the highest accuracy, as indicated by the minimum MAE. The parameters of each model used in this study are presented below.

The details of predictors and the whole configuration of proposed models are shown in the following sections.

4.1.2 Predictors

A series of experiments (Figure 4-1) is being carried out to evaluate and compare the models' estimation accuracy. Five proposed models are being studied in this thesis: MLR, RF, DLMLP, ensemble models, and transfer learning approach.

Regarding a group of factors that might positively impact effort estimation in terms of FPA: in this study, the predictors, including AFP, EI, EO, EQ, EIF, ILF, RS, and IS, are categorised into six groups, with each group comprising different combinations of techniques as follows:

- P1: AFP
- P2: EI, EO, EQ, EIF, ILF
- P3: AFP, IS
- P4: EI, EO, EQ, EIF, ILF, IS
- P5: AFP, IS, RS
- P6: EI, EO, EQ, EIF, ILF, IS, RS
- $P_{Dataset2}$: the predictor based on EI, EO, EQ, EIF, ILF, IS, RS and Dataset 2.

Setting one of the lists (from P1 to P6) as independent variables and SWE is the dependent variable. The aim of using these predictors is to find the most appropriate combination that leads to the highest performance in effort estimation.

According to the findings of the Pearson correlation of features on the Promise repository in section 3.3.2, the following factors are chosen as predictors for Desharnais (P_D), Albrecht (P_A), Kitchenham (P_K), and China (P_C):

- P_D: Length, Transactions, Entities, PointsAjust
- P_A: Input (EI), Output (EO), Enquiry (EQ), File (EIF)
- P_K: duration, AFP, estimate
- P_C: Input (EI), Output (EO), Enquiry (EQ), File (EIF), Added

This study is interested in evaluating deep learning models using transfer learning. Thus, the predictors selected for analysis exhibit similar characteristics to Dataset 1. Moreover, the thesis intends to use P_A and P_C as predictors to build a model based on the pre-trained model obtained from Dataset 1; thus, the study chooses similar features to Dataset 1. As a result, predictors for P_D, P_A, P_K, and P_C are given as follows:

- P_D: Length, Transactions, Entities, and PointsAjust

- P_A: Input (EI), Output (EO), Enquiry (EQ), File (EIF)
- P_K: duration, AFP, estimate
- P_C: Input (EI), Output (EO), Enquiry (EQ), File (EIF)

4.1.3 Experimental Framework

The methodology has been developed to accomplish four main objectives. The steps undertaken to achieve these objectives are described below:

1. **Develop DLMLP, RF and MLR**: The study intends to construct efficient prediction models and compare their performance. The process involves the following steps:
    - Implementing DLMLP, RF and MLR models.
    - Conducting initial assessments to compare the performance of these models with each other.
    - Conducting experiments on predictors: P1, P2, P3, P4, P5, P6, P_D, P_A, P_K, P_C.
2. **Apply class weighting**: This phase explores class weighting techniques to address the class imbalance. The process includes the following:
    - Implementing class weighting methodologies in DLMLP.
    - Conducting experiments on P1, P2, P3, P4, P5, P6 based on Dataset1.
3. **Apply ensemble**: Similar to the previous steps, the study further explores using ensemble techniques to enhance prediction accuracy. Specifically:
    - Implementing stacking ensemble (SE) techniques for MLR and RF, the voting by averaging for SE and DLMLP approach; compare the performance of this approach with MLR, RF, and DLMLP.
    - Conducting experiments using ensemble methods on P1, P2, P3, P4, P5, P6, P_D, P_A, P_K, P_C.
4. **Apply transfer learning**: The research explores the potential benefits of leveraging insights gained from previous training to enhance performance on new datasets. This approach includes:
    - Choosing the best DLMLP model obtained from P1 to P6 as the pre-trained model.
    - Applying the transfer learning based on that pre-trained model and continuing to train on a new set of datasets.
    - Conducting experiments on four datasets (Dataset 2, Albrecht, China) which share the same target feature ('effort') and the same source Input (EI), Output (EO), Enquiry (EQ), File (EIF) with Dataset 1.
5. **Evaluate Performance**: The final step involves assessing the performance of the adopted methods by:
    - Comparing their performances under different scenarios.
    - Evaluating the accuracy of effort estimation achieved by these methods.

4.2 Regression Experiment

The MLR algorithm is implemented using the robust linear regression algorithm available in the Scikit-learn library, a widely used machine learning library. The training dataset is divided into five folds using the K-Fold function [142], [143] to ensure robust evaluation and minimize bias. This technique allows us to create multiple training-validation splits, comprehensively assessing the model's performance. The shuffle and random_state parameters are incorporated during the data shuffling process to introduce randomness and ensure reproducibility. The shuffle parameter randomly reorders the data, while the random_state parameter sets a fixed seed, guaranteeing that the results can be replicated for further analysis.

Next, for each fold generated by the 5-fold technique, the data is further divided into training and validation sets, leveraging the indices provided by the splitting process. This division enables the model to learn patterns from the training set and validate its performance on unseen data. An MLR is created using the LinearRegression() function with default parameters (such as fit_intercept set to be True). By employing this function with default parameters, we construct a multivariate linear regression model capable of capturing complex relationships between the independent and target variables. The code then trains the model on the training set using the fit() function and uses it to predict the target variable on the validation set using the predict() function.

4.3 Random Forest Experiment

RF is a popular machine learning algorithm used for classification and regression tasks. It might be used for predicting both numerical and categorical variables. It might be computationally efficient and is capable of handling large datasets. RF is designed to mitigate overfitting by combining multiple decision trees and using random subsets of data and features. Having sufficient trees in the ensemble might help reduce the overfitting risk and improve generalization performance.

Table 4-1: The experimental-based parameters of RF

No	Parameters	Values
1	n_estimators	{120,150, 180, 210}
2	max_depth	{5, 10, 15, 20, 25, 30}
3	random_state	42
4	min_samples_split	Default value
5	min_samples_leaf	Default value
6	max_features	Default value
7	Num folds	5

The experimental parameters for RF, as outlined in Table 4-1, with 'n_estimators' and 'max_depth' organized into distinct value sets; ‘random_state’ is set to 42; while leaving ‘min_samples_split’, ‘min_samples_leaf’, and ‘max_features’ at their default values. The experimentation is conducted within a cross-validation with 5-fold. Model performance is evaluated using the MAE, and the optimal model configuration is determined based on achieving the minimum MAE through the grid search [144] process.

- **n_estimators**: The number of decision trees in the forest. Increasing the number of estimators can improve performance and increase the computational cost.
- **max_depth**: The maximum depth of each decision tree. Restricting the depth can help prevent overfitting. Set it to None if the trees grow until all leaves are pure or all leaves contain less than min_samples_split samples.
- **random_state**: The seed used by the random number generator. Setting it to a specific value ensures the reproducibility of the results.
- **min_samples_split**: This parameter determines the minimum number of samples required to divide an internal node in constructing a decision tree within the RF. If the number of samples at a node is smaller than min_samples_split, the node is not split further and becomes a leaf node. Increasing this value can prevent overfitting by ensuring that a node has enough samples to make a reliable split.
- **min_samples_leaf**: This parameter sets the minimum number of samples needed to be at a leaf node. If the number of samples at a leaf node is less than min_samples_leaf, additional splitting attempts are not made, and the leaf node is generated. Like min_samples_split, increasing min_samples_leaf can help prevent overfitting and control the size of the tree.
- **max_features**: This parameter determines the maximum number of features that are considered when looking for the best split at each node. The Random Forest algorithm randomly selects a subset of features from the whole feature set, and max_features controls the number of features selected.

4.4 DLMLP Experiment

In DLMLP, each neuron receives input from the previous/input layer. A neuron produces an output sent to the next layer after applying an activation function to the weighted sum of its inputs. Several options are available in activation functions, including sigmoid, tanh, and Rectified Linear Unit (ReLU) [130]. According to Jason Brownlee [131], ReLU is simple to compute and requires few computational resources. ReLU addresses the issue of vanishing gradients, which might impede learning in deep neural networks. By allowing for faster learning

and improved performance [145], ReLU has become a popular choice in many deep-learning applications.

Optimisation algorithms are essential for training deep learning models [128]. Optimisation aims to find the best set of parameters for a model that minimises the loss function, which is the difference between the model's predicted and actual output. Root Mean Squared Propagation (RMSProp) and Adam make adaptive moment estimations to enhance results among Adam, RMSProp, Adaptive Gradient Algorithm and a more robust extension of Adagrad [146]. As a result, the study chose Adam as the optimizer of this model.

According to V.N. Gudivada et al. [147], no rigid or universally established guidelines exist when determining the appropriate number of hidden layers and the number of neurons within each. This study conducted experiments to evaluate different layer configurations for the DLMLP, encompassing architectures with 2, 3, 4, and 5 layers. The optimal architecture of models is identified using the grid search [144] based on the minimum MAE, early stopping with a monitoring criterion based on the minimum MAE also installed.

Figure 4-2 presents an example of deep learning architecture with four fully connected layers. Input variables are collected from Dataset 1, Desharnais, Albrecht, Kitchenham, and China. As mentioned in section 4.1.2, P1 has an input size of 1, P2 has an input size of 5, P3 has an input size of 2, and P4, P5, and P6 have input sizes of 6, 3, and 7, respectively. The input sizes for P_D, P_A, P_K, and P_C were determined as 5, 6, 3, and 6, respectively.

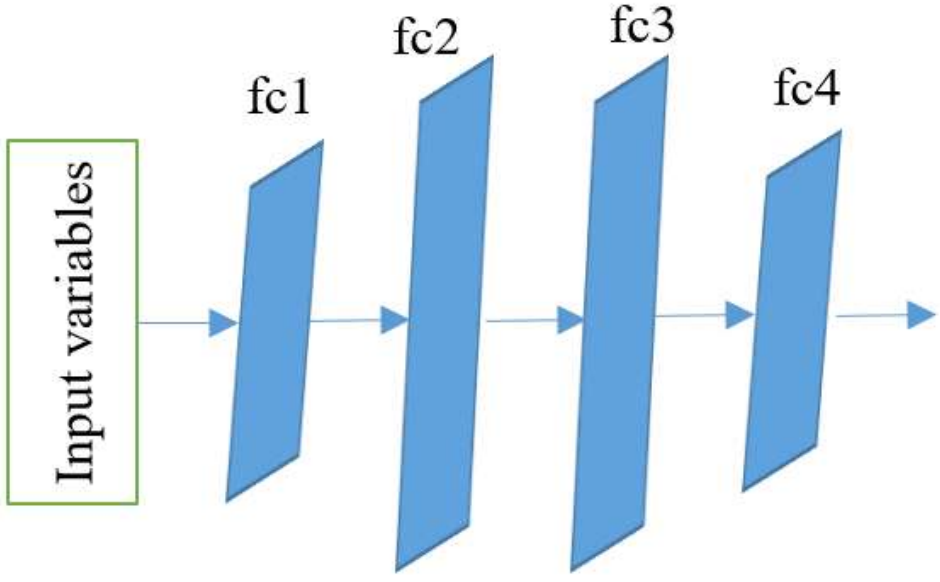


Figure 4-2: An example of the architecture of DLMLP with four fully connected layers

The configuration details provided in Table 4-2 further illuminate the design and training aspects of the deep learning model. Specifically, they encompass essential parameters such as learning rates, batch size, epoch count, rate decay, and cross-entropy loss function. These parameters collectively contribute to the robustness and efficacy of the deep learning model's performance.

Table 4-2: The experimental-based parameters of DLMLP

No	Parameters	Values
1	Learning rate	{0.01,0.001}
2	Batch size	64
3	Epoch	1260
4	Rate decay	0.999
5	Num fold	5
6	Early stop	True, patience = 5
7	Loss function	Cross Entropy Loss

4.5 Transfer Learning Experiment

As mentioned in 3.3.1, Dataset 1 is a widely recognized and standardized dataset containing many software development projects. This thesis chooses the best model built based on Dataset 1 as the pre-trained model (called the ISBSG model). It is trained on a large amount of data, which allows us to learn generalizable features that might be applied to other datasets.

Table 4-2 presents the input and output variables list among studied datasets (Dataset 2, Albrecht, and China). Firstly, considering the output feature, those datasets share the same target variable ('effort'). This similarity indicates that the task/objective of predicting the effort required for software development is consistent between the datasets. Secondly, observation of input variables, those datasets include inputs related to EI, EO, and EQ. They also have input features related to file counts, such as EIF and ILF. Based on these similarities, there is indeed an overlap between Dataset 1 and the studied datasets (Dataset 2, Albrecht, and China) in terms of input variables and output variables. This overlap suggests there is potential for transferring knowledge from the pre-trained model based on Dataset 1 to predicting effort in other datasets.

Table 4-3: The input and output features list among studied datasets

No	Intersect Dataset 1 with	Similarity		Overlap with Dataset 1?
		Inputs	Output	
2	Dataset 2	EI, EO, EQ, EIF, ILF, IS	SWE	Yes
3	Albrecht	Input (EI), Output (EO), Inquiry (EQ), File (EIF)	Effort	Yes
4	China	Input (EI), Output (EO), Inquiry (EQ), File (EIF)	Effort	Yes

According to Pan and Yang [132], it is a kind of inductive transfer learning setting. To address this problem, we might replace the last layer of the pre-trained

model with a new layer that matches the size of new input features in those datasets. As a result, the final layer is trained using the fresh datasets while leaving the other model parameters unchanged. Training the transfer models on the ISBSG pre-trained model might leverage the learned features from the pre-trained model, effectively reducing the data required for training while still achieving high accuracy. Transfer learning based on the ISBSG model involves several steps as follows:

- **Step 1**: Choose the best model obtained from DLMLP based on P1 to P6 as the pre-trained model. As discussed in Section 5.2.1, DLMLP-P4, with six predictors EI, EO, EQ, EIF, ILF, and IS, outperform compared with P1, P2, P3, P5, and P6.
- **Step 2**: Load the pre-trained model: This model is trained on Dataset 1, and then choose the best-proposed model to use as the pre-trained model.
- **Step 3:** Feature mapping as following steps:
  - Extract features using the pre-trained model.
  - Map the features between new input features and the features from the pre-trained model.
- **Step 4**: Freeze all layers except for the last one. In this step, set the 'requires_grad' attribute to False for all layers except the last one. This step ensures the preservation of previously learned features from the pre-trained model, which is crucial as the model is adapted to work with the specific input features of the new dataset. By keeping the lower layers fixed, the model might exclusively focus on adjusting the last layer to accommodate the unique input characteristics of the new dataset. To implement this adaptation, utilize the feature_mapping function. This function performs two essential tasks: first, it reorders the input features to match the model's input order, ensuring the features are in the correct sequence. Second, it prepares the input features by handling any differences between the new dataset and the pre-trained model, such as the absence of certain features or the presence of additional ones. This function allows the model to seamlessly integrate the new dataset while maintaining the integrity of its pre-trained features.
- **Step 5**: Create a new optimiser: A new optimiser is created specifically for the last layer of the model, which is set to require gradients in the previous step. Adam optimiser is chosen as the optimiser (the same optimiser as the pre-trained model).
- **Step 6**: Continue training the model: Train the model on a new dataset. In each iteration, the input is forwarded through the model, the loss is computed, the gradients are computed, and the weights are updated using the optimiser. This process is repeated for several epochs or until the model achieves the minimum MAE criterion based on the early stopping.

4.6 Balancing Dataset Experiment

As shown in Table 4-4 (column before balancing), the number of projects varies significantly across industry sectors. For example, the Construction, Defence, Education, Mining, and Medical Health Care sectors have significantly fewer projects than others. On the other hand, the Communication, Financial, Government, Insurance, and Service Industry sectors have a significantly higher number of projects.

The number of projects in each industry sector might need to be balanced to address this imbalance. In practice, balancing may involve adding more data to underrepresented groups or removing data from overrepresented groups until the number of data points in each group is approximately equal.

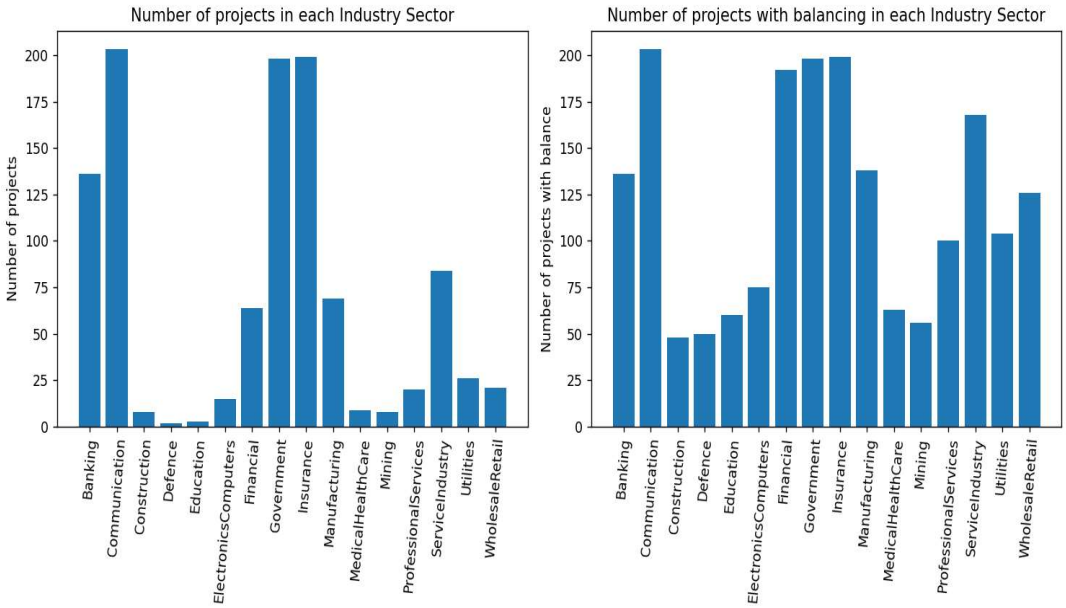


Figure 4-3: Based on the experiment, the histogram of the number of projects in each industry sector before and after balancing

Determining class weights is a critical step to address the class imbalance issue in the dataset. Based on the experiment, a class weighting approach is employed to assign different weights to each industry sector based on the number of projects within each sector. The primary objective is to give more importance to underrepresented sectors while training the model. The class weights are determined as follows:

- For each Industry Sector, the ratio of the number of projects before balancing to the number of projects after balancing is calculated. This ratio reflects the degree of adjustment required to balance the dataset.
- The inverse of these ratios is used as class weights. The less-represented sectors are assigned higher weights, while the overrepresented sectors are assigned lower weights. This approach ensures that the model would pay more attention to the underrepresented sectors during training.

Several experimental iterations are conducted to adjust the class weighting and data balancing techniques in the approach employed to balance the number of projects within each industry sector. The following illustrates these iterations in more detail:

- In the initial iteration, the significant class imbalance in the dataset is observed and addressed by determining the class weights based on the pre-post-balancing project counts for each industry sector, as detailed earlier. The initial class weights are then applied, initiating the baseline experiment.
- Subsequent iterations involve systematic adjustments to the class weights to enhance model performance further.
- Within each iteration, class weights are methodically adjusted based on the outcomes of the preceding iteration to identify the most effective weightings that optimise accuracy without over-balancing the dataset.
- The performance of the model is based on the minimum of MAE.

Figure 4-3 presents the number of projects for each industry sector balancing the dataset by class weighting approach after several experimental iterations. The number of projects is adjusted to achieve the best possible accuracy without over-balancing the dataset. For instance, the Defence sector had only two projects, and it was impossible to balance it further without compromising the integrity of the dataset.

Table 4-4: The number of projects for each industry sector before and after balancing

No	Industry Sector	Before balancing	After balancing
1	Banking	136	136
2	Communication	202	202
3	Construction	8	48
4	Defence	2	50
5	Education	3	60
6	Electronics Computers	15	75
7	Financial	64	192
8	Government	197	197
9	Insurance	199	199
10	Manufacturing	69	138
11	Medical Health Care	9	63
12	Mining	8	56
13	Professional Services	20	100
14	Service Industry	84	168
15	Utilities	26	104
16	Wholesale Retail	21	126

Finally, DLMLPB with a balanced dataset is applied. The configuration of this model is the same as DLMLP presented in Section 4.4.

4.7 Ensemble Model Experiment

In Figure 4-4, the stacking ensemble (SE) for regression incorporates two distinct models: MLR and RF. Data preprocessing is conducted using a 'tree_preprocessor' object to prepare the dataset for modelling. This phase creates two separate pipelines, each tailored to one of the models—MLR and RF. These pipelines integrate preprocessing with the respective model, resulting in two well-defined modelling paths.

These modelling pipelines are combined into a list of tuples. Each tuple pairs the model's name with its corresponding pipeline, establishing a link between preprocessing steps and modelling tasks. A 'StackingRegressor' object is instantiated in the final ensemble setup, employing the 'estimators' list and utilizing an XGBoost model as the ultimate decision-maker.

The training phase involves independent training for each base model, individually refining the MLR and RF models on the training dataset. Predictions generated by these models are thoughtfully aggregated and provided as inputs to the final decision-maker, the XGBoost model. Throughout this training phase, the 'cross_validate' function is pivotal in assessing each model's performance and the ensemble's overall effectiveness. Robustness and generalization are ensured through the use of 5-fold cross-validation.

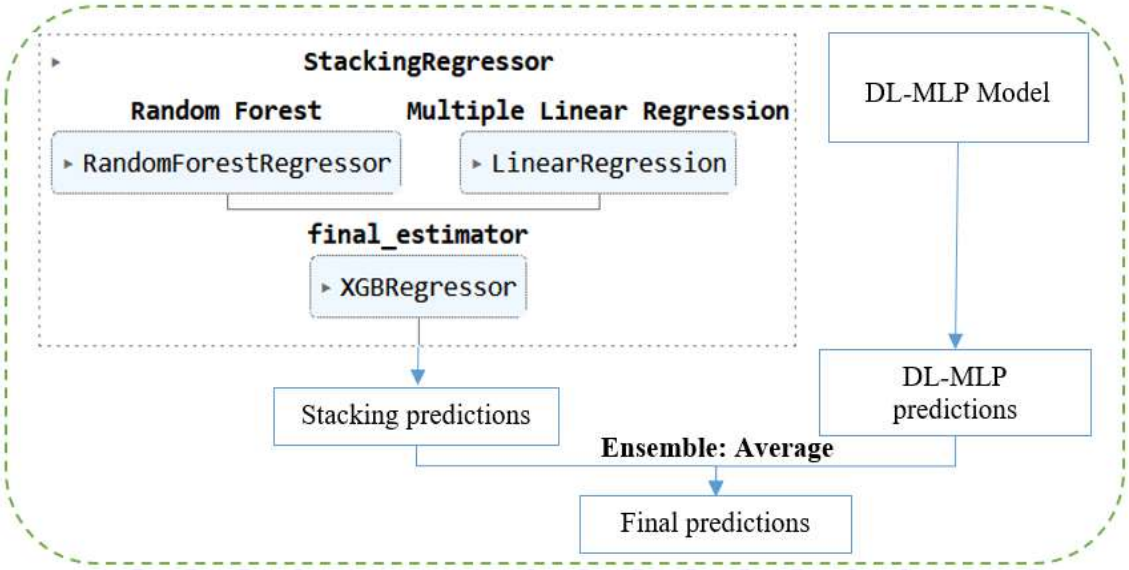


Figure 4-4: The flow diagram of the ensemble model. RF and MLR are used as based estimators, and XGBoost is used as the final estimator. Stacking predictions obtained from RF and DLMLP predictions are ensembled by average to attain the final predictions

Incorporating DLMLP into the SE begins with loading the pre-trained DLMLP model. This model is configured for predictive tasks by setting it to evaluation

mode, ensuring its internal parameters remain unaltered during prediction. Next, testing datasets are prepared and converted into tensor format to align with the DLMLP architecture's requirements. The DLMLP then generates predictions based on the test dataset.

To summarize, the ensemble model has several steps as follows:

- **Step 1 (Defining Base Models)**: Create base models (RF, MLR, DLMLP) using predefined hyperparameters for each model to generate individual predictions.
- **Step 2 (Defining the Meta Model)**: Configure an XGBoost regressor as the meta-model to merge base model predictions.
- **Step 3 (Creating the Stacking Ensemble)**: Set up the stacking ensemble, using StackingRegressor to harmonise base models with the final XGBoost model.
- **Step 4 (Cross-validation)**: Employ 5-fold cross-validation to rigorously evaluate the ensemble's accuracy based on the minimum MAE.
- **Step 5 (Voting by Averaging Predictions)**: Apply a voting mechanism that averages predictions from DLMLP and the stacking ensemble in step 5, producing a unified prediction that balances insights from both sources for enhanced accuracy.

4.8 Model Explainability Experiments

This section uses LIME and SHAP techniques to perform model explainability experiments. Due to time limitations, the thesis only analyses LIME and SHAP based on DLMLP.

As mentioned in Section 3.5, these experiments generate LIME and SHAP explanations, allowing us to discern the positive and negative influences on effort estimation. By analyzing these explanations, we might answer RQ5 and gain an understanding of the features that the model deems significant in making predictions. Through these models' explainability experiments, valuable insights might be gleaned regarding the inner workings of the deep learning model, enabling us to identify potential biases or limitations.

Furthermore, the transparency and trustworthiness of the model's decision-making process might be enhanced as these experiments contribute to developing a more interpretable and reliable AI system. The outcomes derived from these experiments have the potential to advance the field of AI by fostering the creation of AI systems that are both understandable and dependable.

Regarding LIME, the following steps are undertaken to derive and interpret explanations for individual predictions:

- **Step 1**: Begin by instantiating a LIME explainer using the LimeTabularExplainer library.
- **Step 2**: Select an example instance from the testing dataset (see Table 4-5).
- **Step 3**: Utilize the LIME explainer to generate an explanation for the selected instance.

- **Step 4**: The generated LIME explanation offers insights into the contribution of each feature to the prediction for the specific instance. The coefficients of the interpretable model guide the magnitude and direction of each feature's influence. Positive coefficients indicate that increasing a feature's value tends to raise the predicted effort, while negative coefficients suggest the opposite effect.

Regarding SHAP, these values are derived and interpreted as follows:

- **Step 1**: Conversion from LIME to SHAP. The LIME explanation obtained earlier is converted into a format compatible with SHAP, facilitating a broader perspective of feature importance.
- **Step 2**: A SHAP explainer is established using the SHAP.Explainer library. It takes as input the prediction function and the reshaped training data, enabling the computation of SHAP values.
- **Step 3**: SHAP values are computed for the entire test dataset using the SHAP explainer. These values quantitatively represent the contribution of each feature to the prediction for each instance within the dataset.
- **Step 4**: Visualize the representation and interpretation. This visual representation aids in comprehending the relative importance of features across the dataset.

Table 4-5 presents a specific instance that is being used to demonstrate the application of LIME. It presents various features and corresponding values for each project. This instance serves as an example for illustrating how LIME might be utilized to explain and interpret the relationship between the features and the actual effort in that project.

Table 4-5: The scenario of instance for illustrating LIME

Features	Instance	Specific Unit
EI	209	Function Points
EO	129	Function Points
EQ	24	Function Points
EIF	15	Function Points
ILF	83	Function Points
IS	Communication	Function Points
RS	M2	Function Points
Real Effort	10200	Person-Hours

4.9 Baseline Models

This section proposes three baseline models: one statistical model (stepwise-based regression), one simple artificial neural network (ANN) model from previous research, and IFPUG-FPA. IFPUG-FPA is introduced in Section 3.1.

Three baseline models are used to compare the performance of the best model among MLR, RF, and DLMLP based on Dataset 1. The thesis employs a set of metrics, namely MMRE, MBRE, MIBRE, MAE, Pred(0.25), and SA, to evaluate the performance of the best model compared with the baseline models. It is worth noting that the same dataset used for validation by the best-performing model is also employed in assessing the baseline models.

4.9.1 ANN-based Model

A simple ANN-based model with two hidden layers has been employed as the baseline model. The purpose of choosing two hidden layers is to make the model simple and naive to determine the minimum performance that might be expected. If the best model does not perform significantly better than the baseline, it might be overfitting or not appropriately capturing the underly patterns. The parameters adopted in the ANN-based model are presented in Table 4-6.

Table 4-6: The parameters of a simple ANN-based model

No	Parameters	Values
1	Learning rate	0.01
2	Batch size	64
3	Epoch	100
4	Rate decay	0.999
6	Early stop	True, patience = 5
7	Loss function	Cross Entropy Loss

4.9.2 Stepwise-based Regression Model

Stepwise-based regression (SWR) [47], [112] is a technique widely used in statistical modelling, drawing inspiration from previous publications [46], [47], [148]. This approach to multiple linear regression involves an automated process for selecting independent variables and might be summarized as follows:

- Initialization: Begin with a starting model containing predefined terms (backward selection) or a null model (forward selection).
- Model Complexity: Define the desired model complexity, specifying which terms should be included, such as linear, quadratic, or interaction terms.
- Evaluation threshold: Set an evaluation threshold based on the sum of residual errors. This threshold determines whether to add or remove features.
- Iterative Process: The algorithm iteratively adds or removes features while re-evaluating the model at each step.
- Termination: Stepwise regression continues until no further improvement in estimation is achievable based on threshold.

Forward selection initiates with a null model and progressively adds features that meet specific criteria. Conversely, backward selection starts with a full model and removes non-significant features. Consequently, stepwise regression

necessitates two significance levels: one for adding features and another for removing features.

5. RESULTS AND DISCUSSION

The results of the experiment and discussion are illustrated in this section.

5.1 Comparison of Model Performance

This section presents a comprehensive evaluation of effort estimation methods. The evaluation encompasses a range of machine learning techniques, including MLR (see Section 4.2), RF (see Section 4.3), DLMLP (see Section 4.4), DLMLPB (see Section 4.6), the ensemble by incorporating between RF, MLR, DLMLP (see Section 4.7). Two primary evaluation tables, Table 5-1 and Table 5-2, are employed to assess the performance of these methods under various conditions. Table 5-3 illustrates the performance of baseline models.

Table 5-1 focuses on assessing effort estimation methods using Dataset 1. This table provides a detailed analysis of model performance metrics, including MMRE, MBRE, MIBRE, MAE, Pred(0.25)/Pred(0.30), and SA. The rows represent different models, including MLR, RF, DLMLP, the ensemble, and DLMLPB models. The performance evaluation in this table offers insight into the effectiveness of these methods when applied to Dataset 1.

Table 5-1: The performance of effort estimation obtained from MLR, RF, the ensemble, and DLMLPB based on testing of Dataset 1

Predictors /Models	MMRE	MBRE	MIBRE	MAE	PRED		SA
					0.25	0.30	
P1							
MLR	0.9113	1.0057	0.3831	2173.83	0.27	0.32	0.43
RF	0.6879	0.8047	0.3661	2150.85	0.28	0.32	0.46
DLMLP	0.6709	0.7719	0.3637	2066.69	0.29	0.34	0.51
Ensemble	0.5478	0.7229	0.3582	1986.74	0.29	0.34	0.52
DLMLPB	0.6228	0.7702	0.3606	2016.14	0.29	0.34	0.52
P2							
MLR	1.2273	1.3250	0.4134	2254.00	0.26	0.32	0.43
RF	0.9802	1.0675	0.3786	2118.20	0.30	0.38	0.46
DLMLP	0.5526	0.7360	0.3044	1768.61	0.46	0.53	0.58
Ensemble	0.4853	0.6132	0.2920	1669.18	0.46	0.54	0.61
DLMLPB	0.4568	0.5378	0.2874	1464.35	0.47	0.52	0.65

P3							
MLR	0.9081	1.0012	0.3824	2172.63	0.28	0.34	0.45
RF	0.6820	0.7964	0.3626	2123.92	0.32	0.35	0.46
DLMLP	0.6275	0.7812	0.3619	2033.24	0.32	0.36	0.51
Ensemble	0.5362	0.7464	0.3553	2024.51	0.34	0.36	0.52
DLMLPB	0.5639	0.6713	0.3356	1915.79	0.36	0.42	0.54
P4							
MLR	1.1999	1.2851	0.4090	2018.79	0.27	0.32	0.45
RF	0.9784	1.0579	0.3750	2012.14	0.31	0.38	0.47
DLMLP	0.2478	0.4311	0.1572	530.65	0.79	0.84	0.87
Ensemble	0.3119	0.3657	0.2189	1007.28	0.62	0.73	0.75
DLMLPB	0.1871	0.2064	0.1393	494.20	0.82	0.85	0.88
P5							
MLR	1.1551	1.0378	0.5949	2335.22	0.27	0.30	0.4
RF	0.6875	0.8028	0.3645	2145.08	0.32	0.35	0.45
DLMLP	0.6326	0.7830	0.3621	2118.93	0.32	0.36	0.49
Ensemble	0.6115	0.7281	0.3517	1983.02	0.31	0.36	0.52
DLMLPB	0.6855	0.7842	0.3567	2069.72	0.33	0.37	0.51
P6							
MLR	0.8981	1.0129	0.3967	2228.68	0.28	0.32	0.42
RF	0.7756	0.8632	0.3649	2029.29	0.30	0.39	0.48
DLMLP	0.3489	0.4750	0.2219	963.71	0.68	0.71	0.77
Ensemble	0.3599	0.4483	0.2479	1143.07	0.56	0.66	0.72
DLMLPB	0.2586	0.3551	0.1731	550.82	0.76	0.78	0.86

Table 5-2 expands the evaluation by examining the performance of effort estimation methods across a broad spectrum. In addition to Dataset 2, this table incorporates other datasets such as Desharnais, Albrecht, Kitchenham, and China datasets. The evaluation includes a comparison of MLR, RF, the ensemble, and transfer learning cases 1, 2, and 3 (TL-Case1, TL-Case2, TL-Case3, see in Section 3.4.5), along with an ensemble-based approach. As mentioned in Section 3.3.2, TL-Case1 and TL-Case3 for Desharnais and Kitchenham and TL-Case1 for Albrecht and China are not measured due to differences in input features. The metrics used for evaluation are consistent with those in Table 5-1. This

comprehensive analysis allows us to assess the effectiveness of these techniques across diverse datasets.

Table 5-2: The performance of effort estimation obtained from MLR, RF, TL-Case1, TL-Case2, TL-Case3, and the ensemble based on testing of Dataset 2, Desharnais, Albrecht, Kitchenham and China datasets

Preditors/ Models	MMRE	MBRE	MIBRE	MAE	PRED		SA
					0.25	0.30	
P_D							
MLR	0.4202	0.5795	0.3340	2539.94	0.25	0.38	0.28
RF	0.3850	0.5431	0.2989	2514.43	0.44	0.50	0.29
TL-Case1	-	-	-	-	-	-	-
TL-Case2	0.2076	0.2507	0.1693	1333.22	0.68	0.75	0.65
Ensemble	0.2430	0.3397	0.2231	1860.73	0.50	0.75	0.51
TL-Case3	-	-	-	-	-	-	-
P_A							
MLR	3.2224	0.5479	3.1693	7.13	0.4	0.4	0.54
RF	1.8730	1.9115	0.3906	4.73	0.4	0.4	0.65
TL-Case1	-	-	-	-	-	-	-
TL-Case2	0.3201	0.3220	0.1950	1.44	0.6	0.6	0.84
Ensemble	0.8081	0.8397	0.3368	3.66	0.4	0.4	0.61
TL-Case3	0.1088	0.1839	0.1087	0.38	0.8	0.8	0.90
P_K							
MLR	0.7583	0.5207	0.3193	589.35	0.42	0.43	0.64
RF	0.4716	0.4870	0.2364	471.52	0.50	0.57	0.71
TL-Case1	-	-	-	-	-	-	-
TL-Case2	0.2204	0.3413	0.1594	240.88	0.75	0.78	0.81
Ensemble	0.2276	0.2435	0.1644	262.80	0.75	0.78	0.81
TL-Case3	-	-	-	-	-	-	-
P_C							
MLR	1.6664	1.8916	0.4550	2762.83	0.27	0.28	0.25
RF	1.6386	1.8595	0.4507	2595.62	0.27	0.28	0.29
TL-Case1	-	-	-	-	-	-	-
TL-Case2	0.9833	1.0569	0.2659	1034.31	0.58	0.59	0.72

Ensemble	0.9626	1.0241	0.3235	1447.57	0.47	0.59	0.62
TL-Case3	0.2092	0.2325	0.1578	247.21	0.79	0.83	0.93
$P_{Dataset\ 2}$							
MLR	0.6681	0.9829	0.1471	1162.98	0.33	0.34	0.00
RF	0.3031	0.3698	0.2571	677.45	0.66	0.67	0.23
TL-Case1	0.4951	1.0574	0.4539	1165.51	0.17	0.17	0.00
TL-Case2	0.2480	0.2557	0.1796	438.48	0.66	0.83	0.43
Ensemble	0.2182	0.2518	0.1872	482.28	0.66	0.67	0.38
TL-Case3	0.1884	0.2310	0.1731	463.10	0.66	0.83	0.40

Table 5-3 displays the evaluation results for effort estimation derived from three baseline models: ANN-based, SWR-based, and IFPUG-PFA. This table offers an in-depth examination of the performance metrics for these models, primarily focusing on the testing of Dataset 1. The assessment of model performance encompasses the analysis of six predictors, denoted as P1 to P6.

Table 5-3: The performance of effort estimation obtained from baseline models (ANN, SWR, IFPUG) based on Dataset 1

Predictors /Models	**MMRE**	**MBRE**	**MIBRE**	**MAE**	**PRED**		**SA**
					0.25	**0.30**	
P1							
ANN	0.6760	0.7769	0.3592	2067s.64	0.26	0.28	0.47
SWR	1.5740	1.6748	0.4380	2373.19	0.22	0.28	0.39
P2							
ANN	0.6081	0.8216	0.3056	1786.68	0.41	0.45	0.54
SWR	1.1787	1.2716	0.4075	2177.96	0.24	0.30	0.44
P3							
ANN	0.6929	0.9452	0.3659	2096.01	0.29	0.35	0.46
SWR	1.5340	1.6240	0.4330	2364.94	0.25	0.30	0.41
P4							
ANN	0.3319	0.3716	0.2063	732.24	0.65	0.71	0.81
SWR	1.1734	1.2640	0.4074	2175.54	0.25	0.31	0.44
P5							
ANN	0.6455	0.7940	0.4156	2152.72	0.31	0.36	0.48
SWR	1.1726	1.0276	0.6099	2310.61	0.27	0.30	0.41
P6							

ANN	0.4628	0.9579	0.2963	1097.54	0.52	0.56	0.72
SWR	0.9092	0.8835	0.4612	2174.54	0.26	0.32	0.44
IFPUG							
IFPUG-PFA	1.7977	1.7989	0.5070	6652.55	0.16	0.18	0.00

5.2 Discussion of the Results

This section focuses on addressing and providing insights into five key RQs. The aim is to examine and compare various estimation methods and predictor sets to determine their accuracy in predicting effort estimation. Each research question serves as a crucial inquiry in understanding the effectiveness of different approaches.

5.2.1 Comparing Predictive Accuracy in SDEE: DLMLP, MLR, RF

This study comprehensively analyses model performance across different predictor groups, focusing on MLR, RF, and DLMLP models. The objective is to compare the performance of these models individually with each predictor group and to answer RQ1: Is the DLMLP more accurate than the MLR, RF? The performance of those models is presented in Table 5-1 and Table 5-2. Figure 5-1 to Figure 5-11 illustrates a detailed comparison of performance metrics for three distinct models: DLMLP, MLR, and RF. Each model is illustrated by a unique colour on the chart, with MLR depicted as a blue bar, RF as orange, and DLMLP as green.

In the P1 predictor group, MLR achieves an MAE of 2173.83, while RF exhibits a slightly improved MAE compared to MLR. However, DLMLP achieves the lowest MAE at 2066.69. Similar trends are observed across metrics such as MMRE, MBRE, MIBRE, Pred(0.25), and SA (see Figure 5-1). These results collectively highlight RF's superior performance over MLR, with DLMLP surpassing both MLR and RF.

Within the P2 predictor group, across multiple metrics, including MMRE, MBRE, MIBRE, and MAE, DLMLP consistently outperforms MLR and RF (see Figure 5-2). For instance, MAE values for MLR, RF, and DLMLP are 2254.00, 2118.20, and 1768.61, respectively. DLMLP's predictive prowess is notably evident through its superior performance across these metrics, with RF outperforming MLR.

Similarly, in the P3 predictor group, competitive performance is observed between MLR and RF, with RF exhibiting advantages across all metrics. Nevertheless, DLMLP consistently outperforms RF and MLR within this predictor group (see Figure 5-3).

Moreover, in the P4 predictor group, DLMLP achieves the lowest MAE with a value of 530.64, outperforming MLR (2018.79) and RF (2012.14). While other metrics, including MMRE, MBRE, MIBRE, Pred(0.25), and SA, favour RF over

MLR, DLMLP's performance exceeds that of both MLR and RF in terms of all metrics (see Figure 5-4). Additionally, P4 predictor group outperforms compared with P1, P2, P3, P5, and P6.

Consistent with the trends observed in the P5 and P6 predictor groups, DLMLP consistently demonstrates superior predictive performance over MLR and RF across multiple metrics, including MNRE, MBRE, MIBRE, and MAE (see Figure 5-5, and Figure 5-6). DLMLP's proficiency in prediction is evident through its consistently lower metric values.

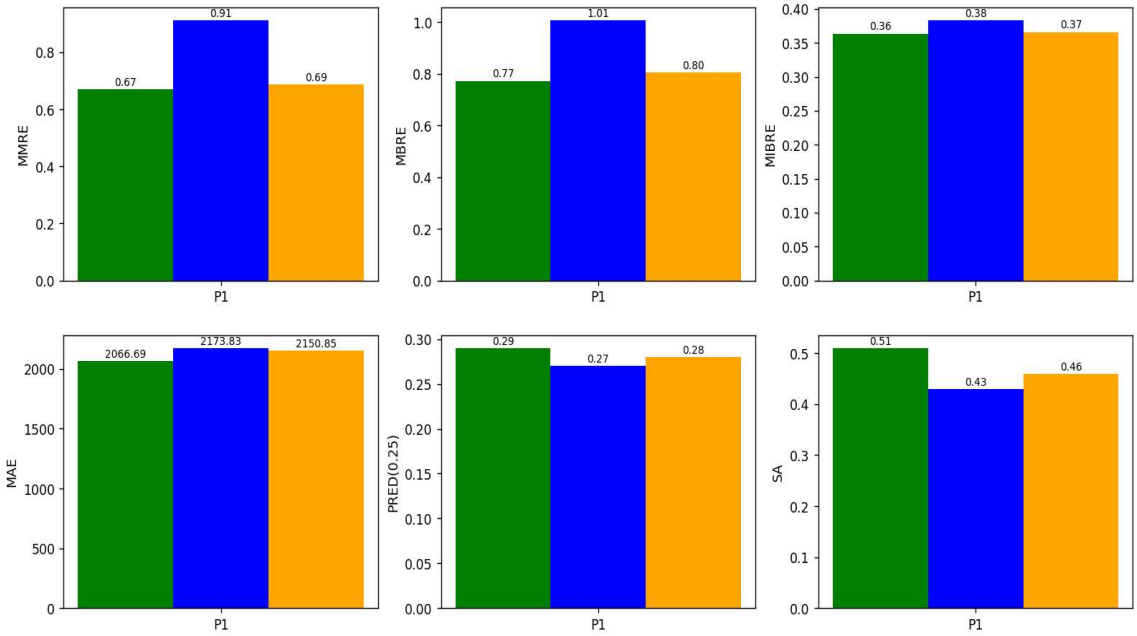


Figure 5-1: The performance of DLMLP compared to MLR, RF based on P1

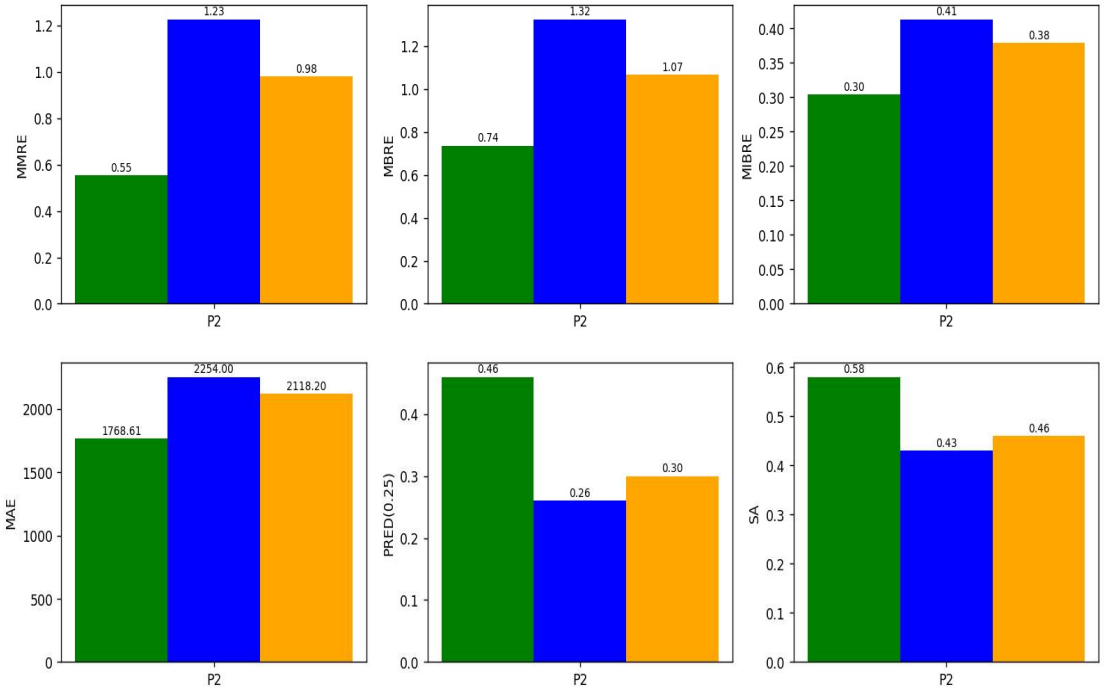


Figure 5-2: The performance of DLMLP compared to MLR, RF based on P2

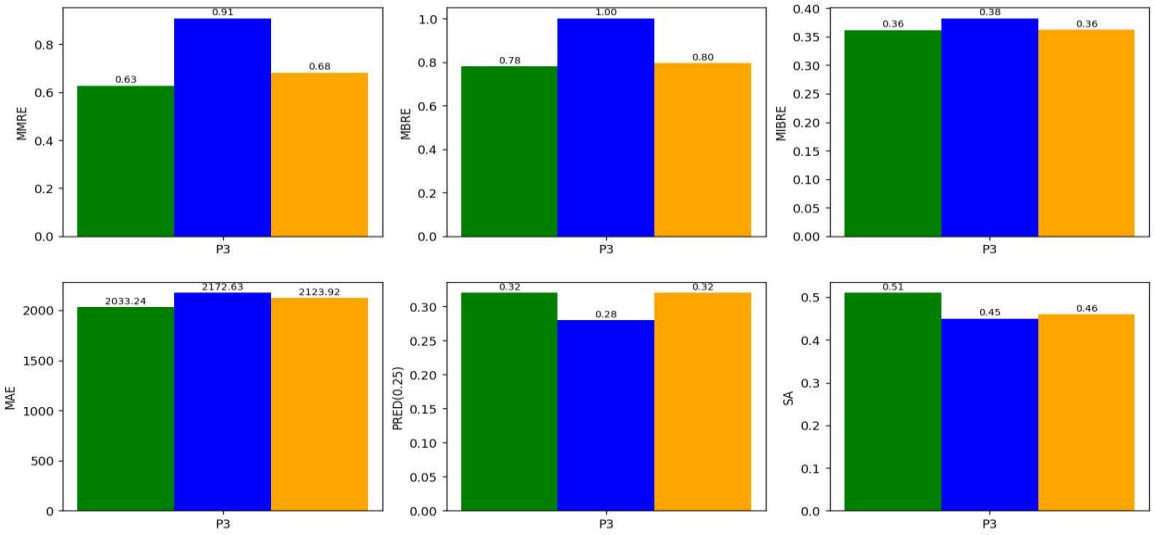


Figure 5-3: The performance of DLMLP compared to MLR and RF based on P3

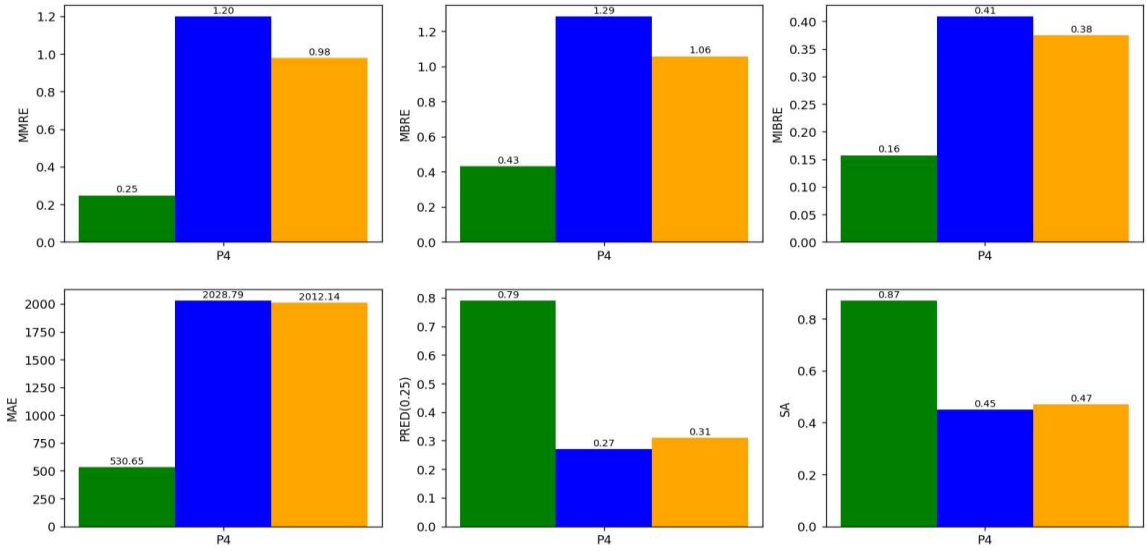


Figure 5-4: The performance of DLMLP compared to MLR RF based on P4

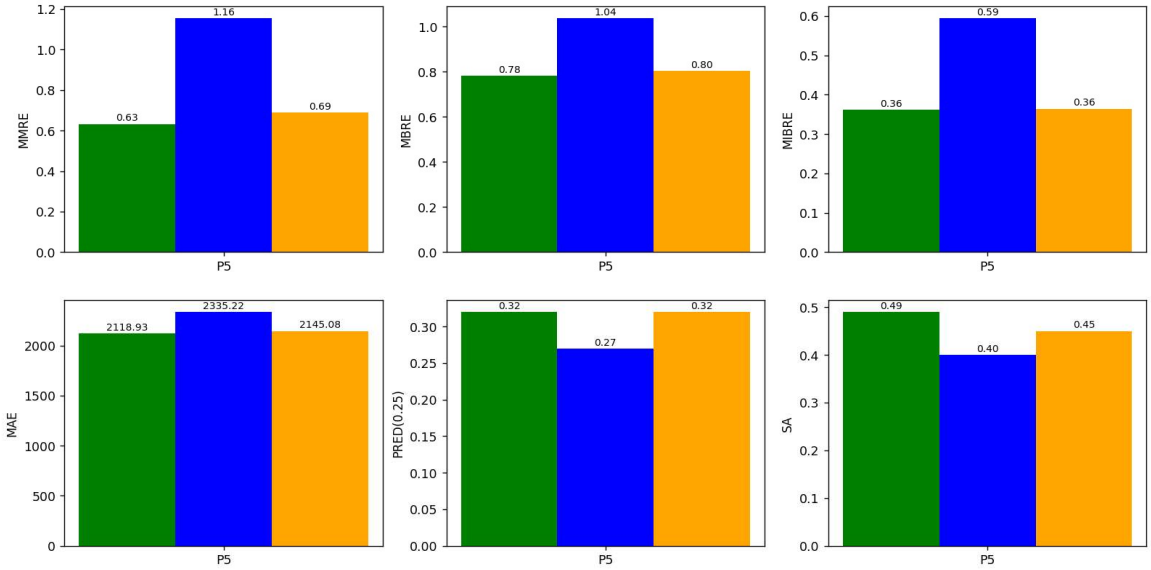


Figure 5-5: The performance of DLMLP compared to MLR RF based on P5

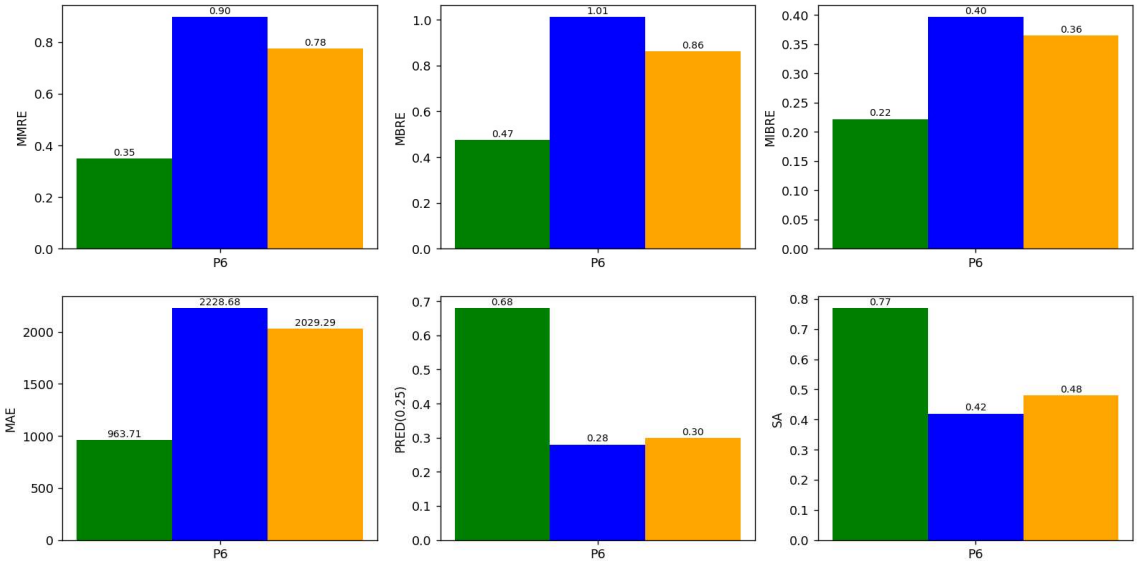


Figure 5-6: The performance of DLMLP compared to MLR, RF based on P6

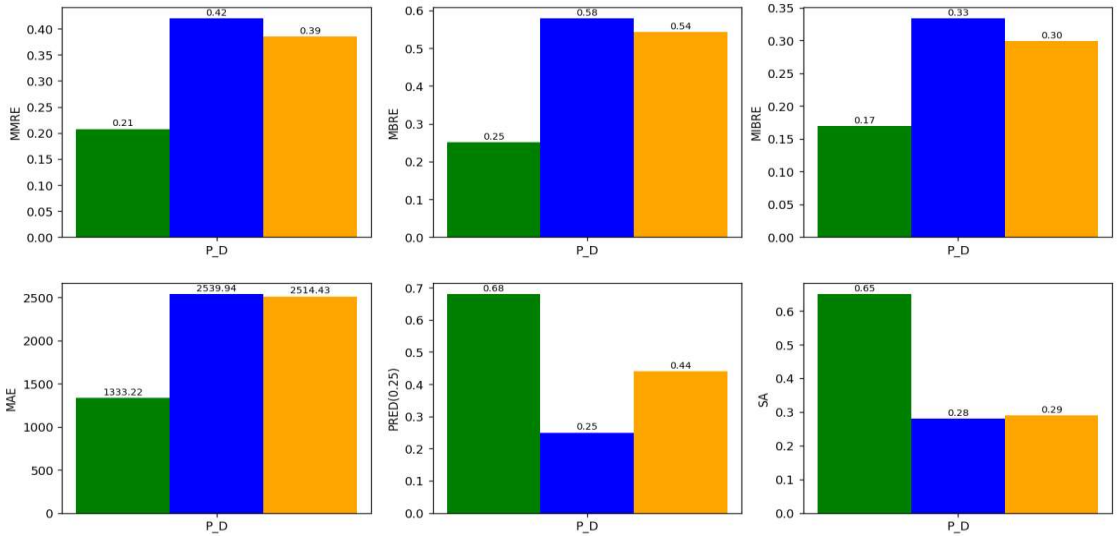


Figure 5-7: The performance of DLMLP compared to MLR RF based on P_D

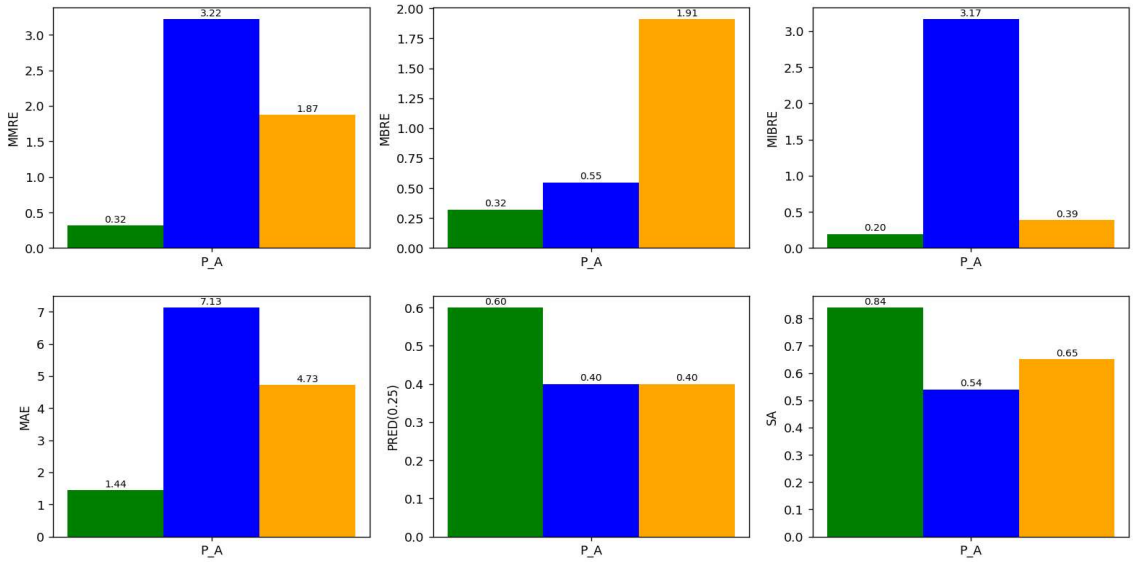


Figure 5-8: The performance of DLMLP compared to MLR, RF based on P_A

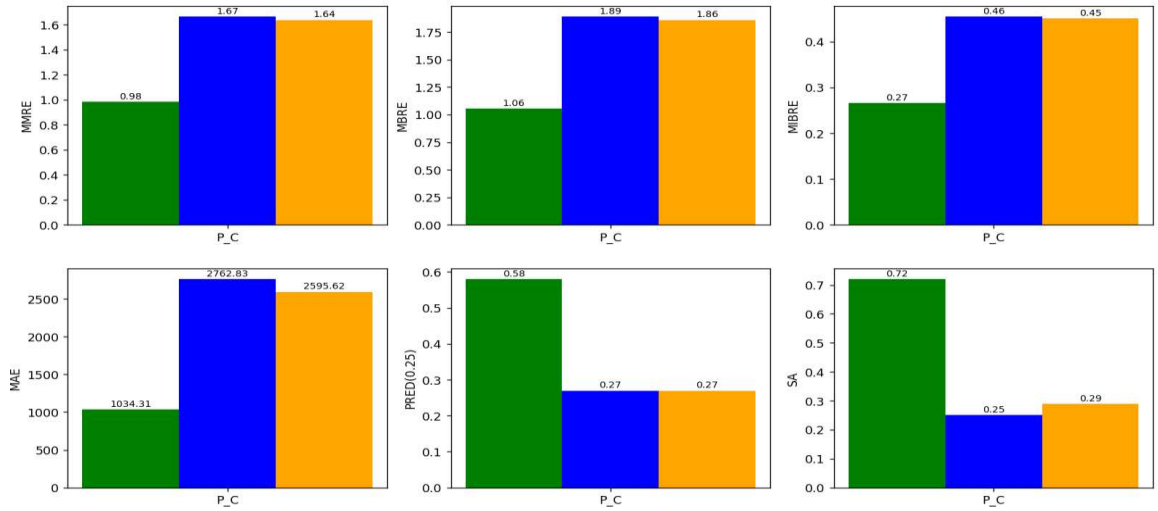


Figure 5-9: The performance of DLMLP compared to MLR, RF based on P_C

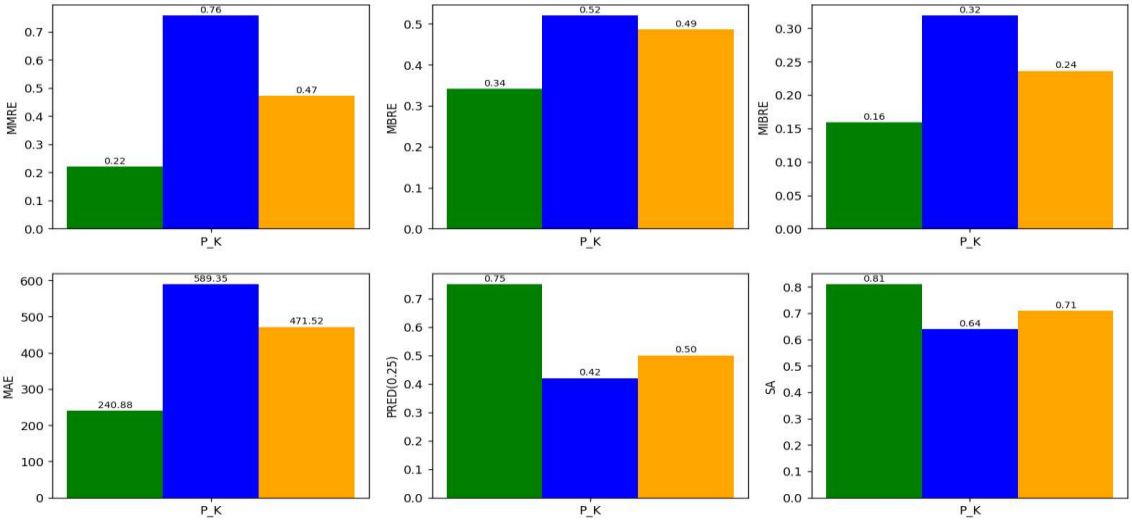


Figure 5-10: The performance of DLMLP compared to MLR RF based on P_K

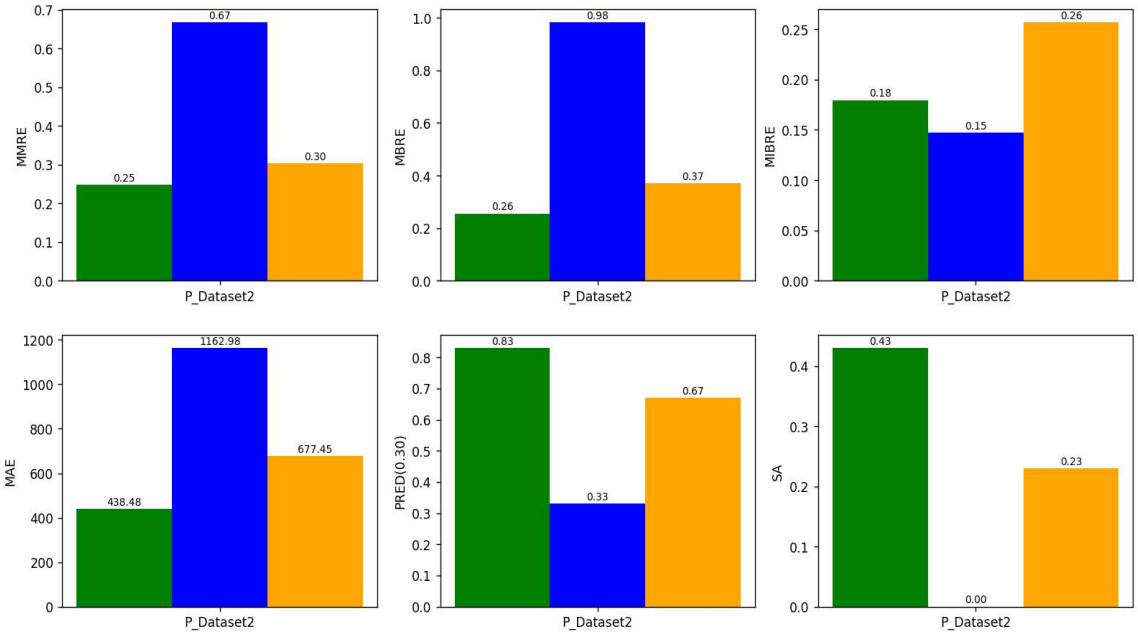


Figure 5-11: The performance of DLMLP compared to MLR, RF based on $P_{Dataset2}$

As previously mentioned, the trends observe in the predictor persist in P_D, P_A, P_K, P_C, and $P_{Dataset2}$ predictors. As presented from Figure 5-7 to Figure 5-11, DLMLP consistently outperforms MLR and RF regarding MMRE, MBRE, MIBRE, MAE, Pred(0.25), and SA, establishing itself as the preferred model within these scenarios. RF demonstrates improved performance over MLR in the majority of cases.

In conclusion, this study provides definitive answers to the research questions. RQ1, which investigates the accuracy of DLMLP compared to MLR and RF models, confirms that DLMLP consistently outperforms both models across all predictive factors. The lower MMRE and MBRE values achieved by DLMLP indicate its ability to capture the magnitude and bias of estimation errors more effectively than MLR and RF models. These results highlight the potential of DL techniques, specifically DLMLP, in improving effort estimation accuracy.

5.2.2 Comparing DLMLP vs. Baseline Models

Comparing the best model to baseline models helps gauge its relative performance. If the complex model, in this case, DLMLP (as discussed in Section 5.2.1), does not significantly outperform baseline models, it raises questions about its complexity and ability to capture crucial data patterns.

The performance of DLMLP compared with baseline models is presented in Figure 5-12 to Figure 5-17, where the green bar stands for DLMLP, and the yellow, red, and black stand for ANN, SWR, and IFPUG-FPA, respectively.

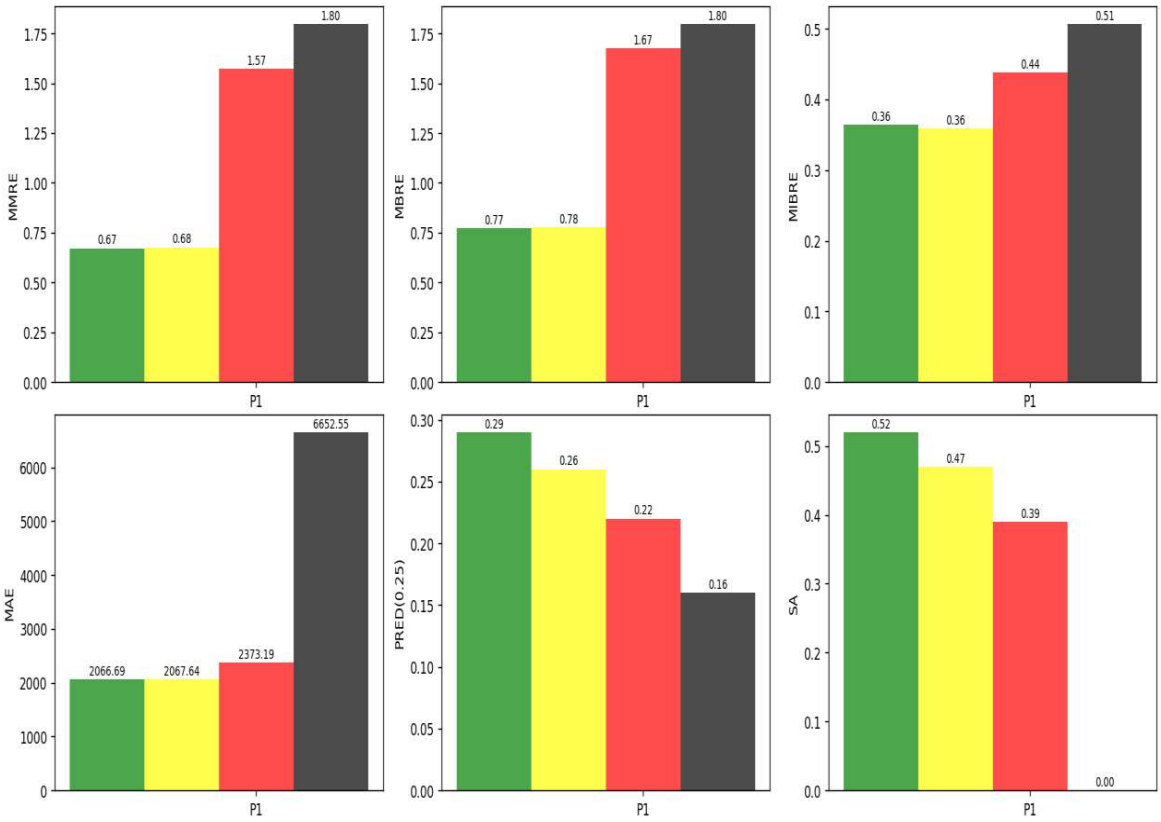


Figure 5-12: The performance of DLMLP compared to baseline models based on P1

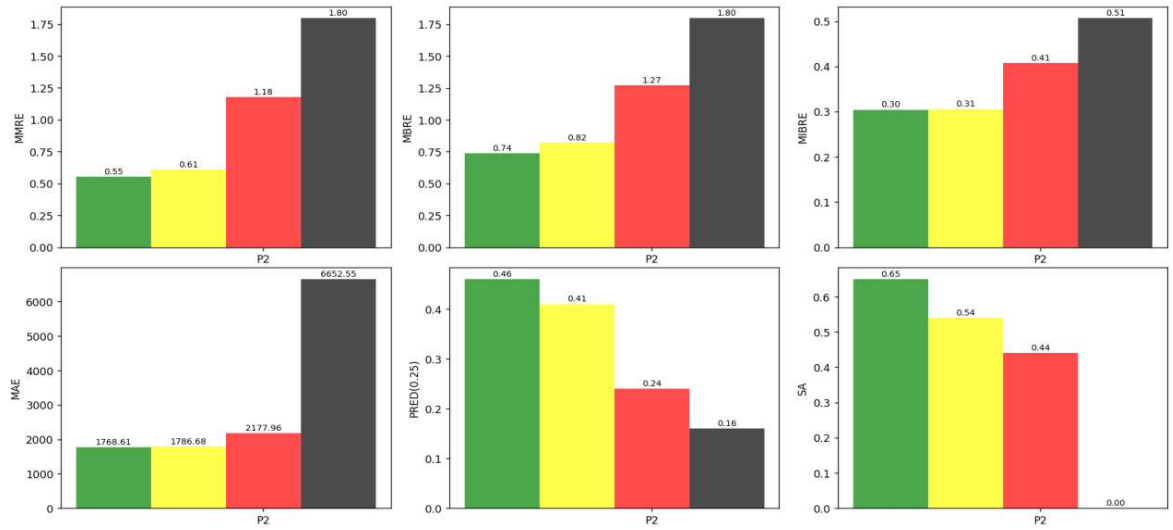


Figure 5-13: The performance of DLMLP compared to baseline models based on P2

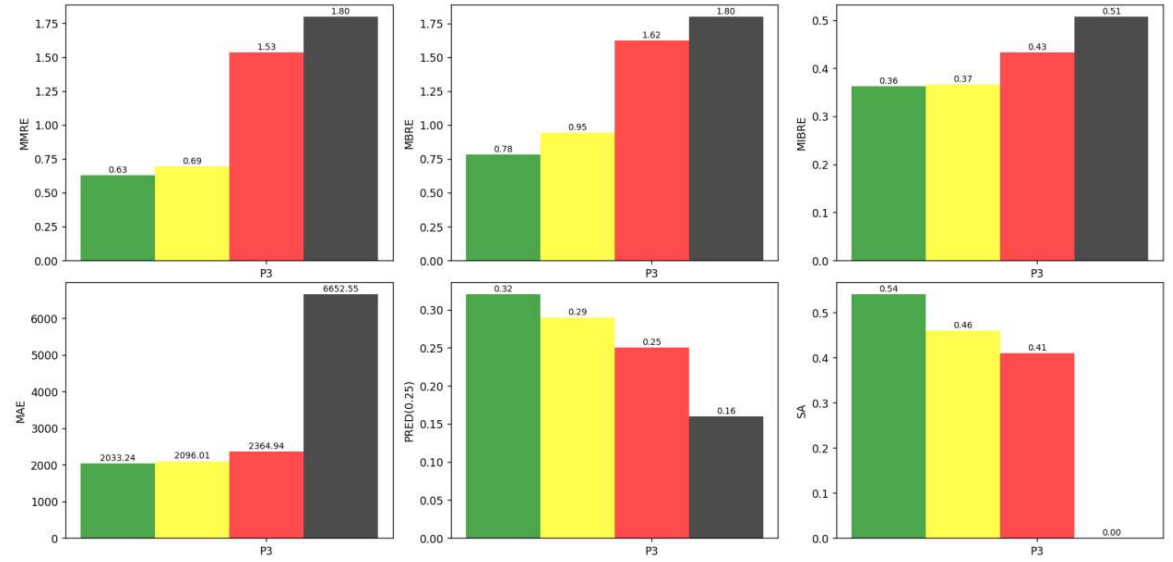


Figure 5-14: The performance of DLMLP compared to baseline models based on P3

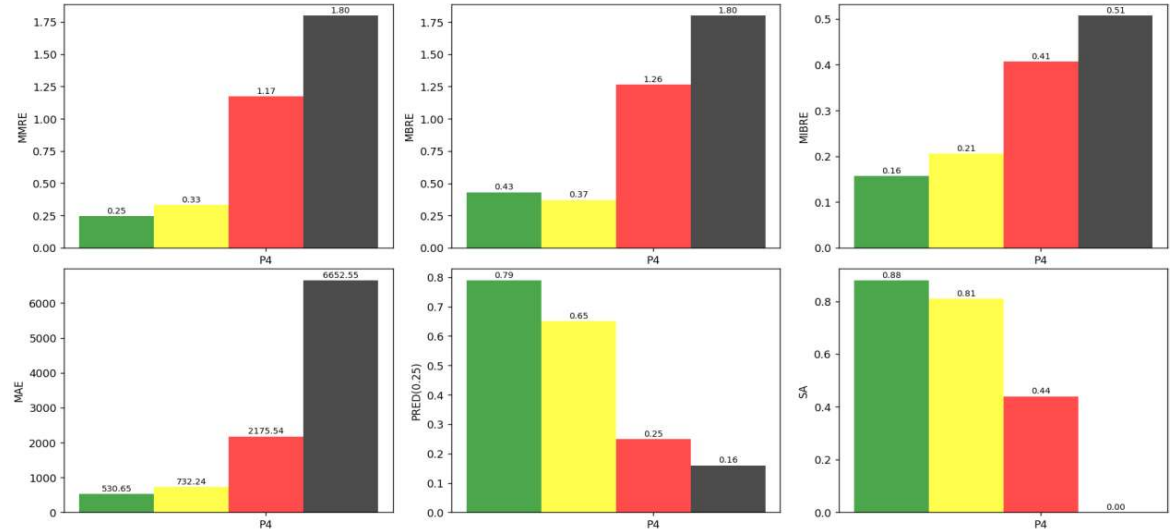


Figure 5-15: The performance of DLMLP compared to baseline models based on P4

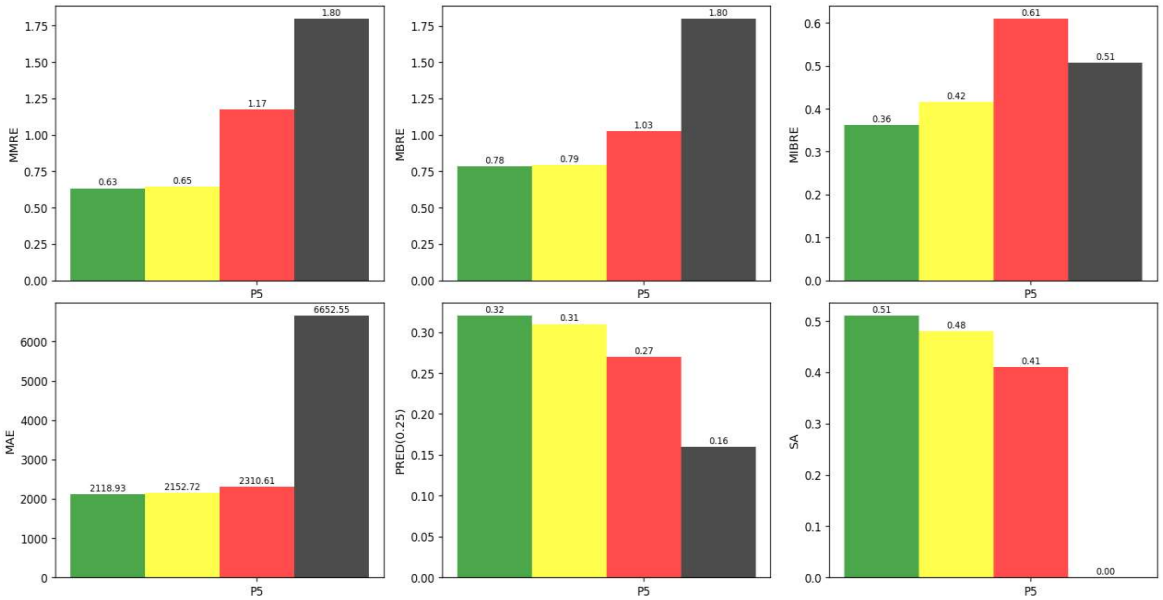


Figure 5-16: The performance of DLMLP compared to baseline models based on P5

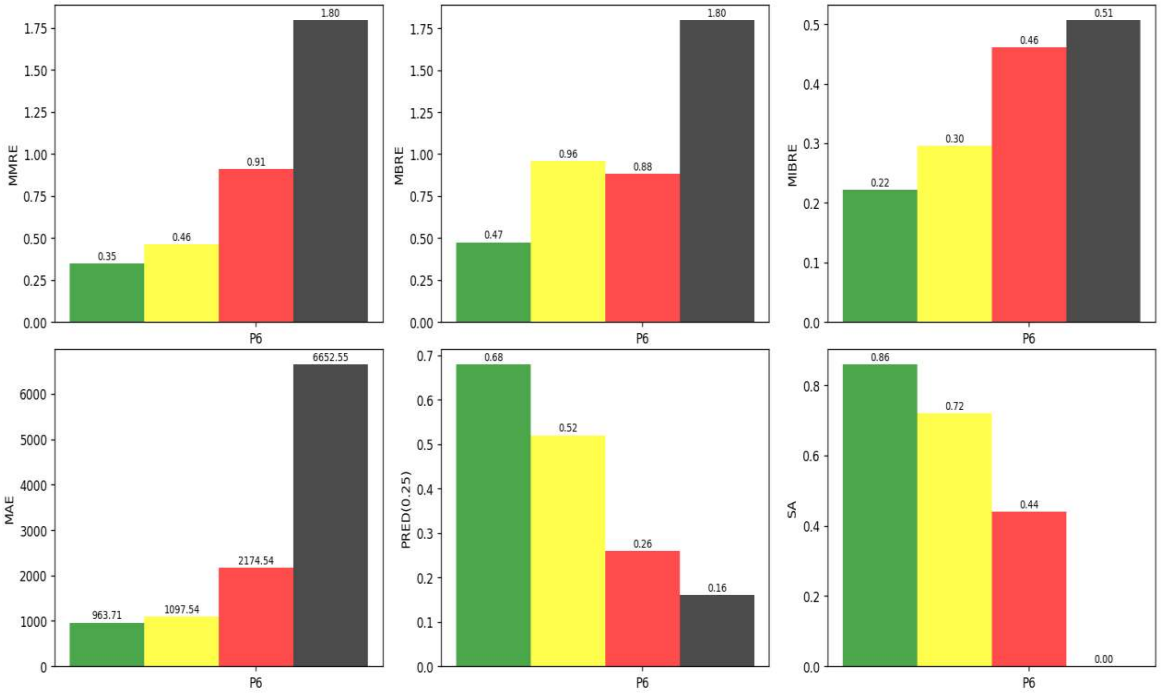


Figure 5-17: The performance of DLMLP compared to baseline models based on P6

It is noticeable that those figures reveal that the DLMLP consistently outperforms the baseline models across diverse datasets (P1 to P6) based on various performance metrics. DLMLP achieves lower values across metrics, including MMRE, MBRE, and MIBRE. These results imply that DLMLP provides more accurate and less biased effort estimations than the alternative models.

Its superior performance extends to the MAE, demonstrating its effectiveness in minimizing the absolute difference between predictions and actual effort values. The strength of the DLMLP further manifests in its ability to provide

predictions within a specified tolerance. Higher Pred(0.25) values indicate that DLMLP delivers effort estimation that closely aligns with the actual effort.

5.2.3 Impact of Dataset Balancing on Accuracy of SDEE in DLMLP

The other aim of this study is to comprehensively assess the impact of using a balanced dataset and handling categorical variables in deep learning models in the context of effort estimation. Specifically, the thesis compares model performance with a balanced dataset based on categorical variable handling (DLMLPB) against the model without balancing categorical variables (DLMLP) across predictors P1, P2, P3, P4, P5, and P6. This evaluation answers RQ2 that dataset balancing might enhance the predictive accuracy of DLMLP methods in software effort estimation. Model performance evaluation is based on critical metrics such as MMRE, MBRE, MIBRE, MAE, Pred, and SA.

Figure 5-18 to Figure 5-23 visually illustrate the performance comparison between DLMLP and DLMLPB based on datasets P1 to P6. The green bar stands for DLMLP, and violet stands for DLMLPB. The results reveal that DLMLPB consistently outperformed DLMLP in estimation accuracy across all predictors. This finding provides strong evidence to support the notion that using a balanced dataset and effectively handling categorical variables leads to improved estimation accuracy in deep learning models.

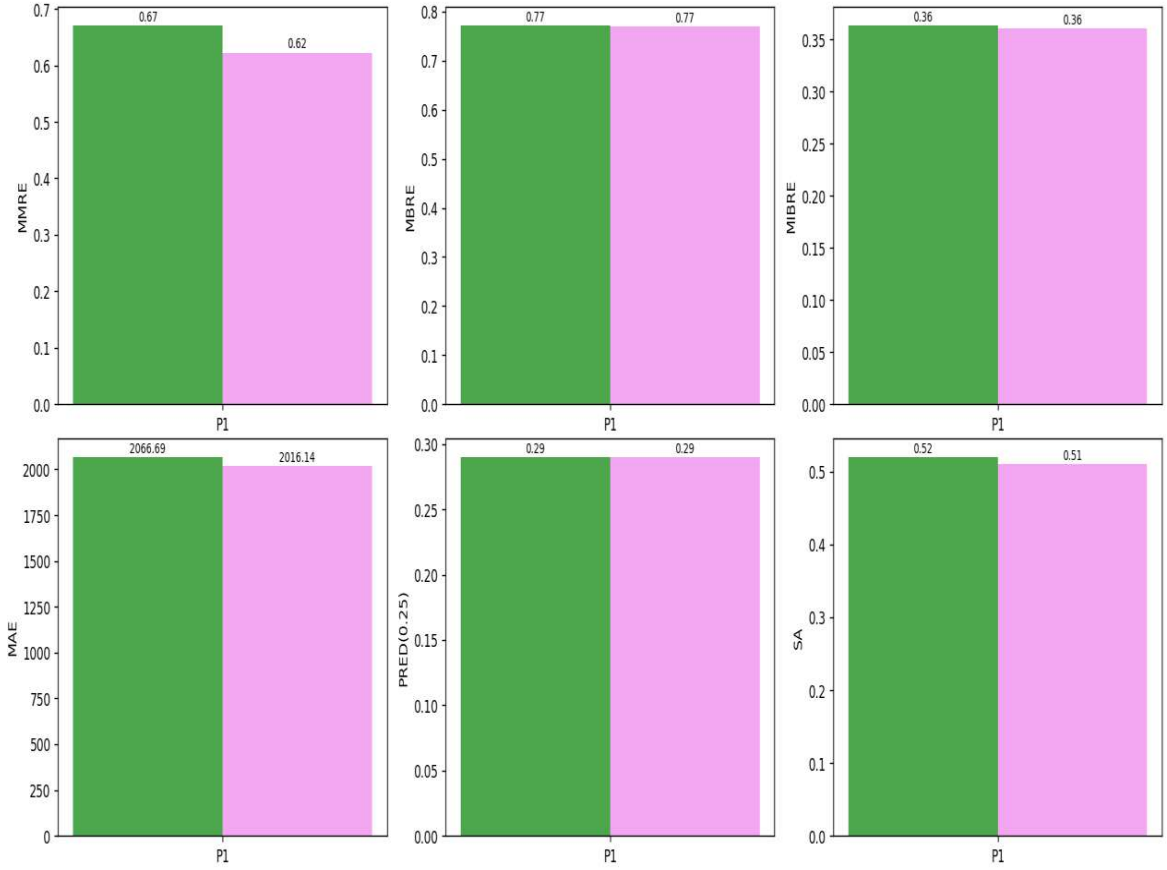


Figure 5-18: The performance of DLMLP compared to DLMLPB based on P1

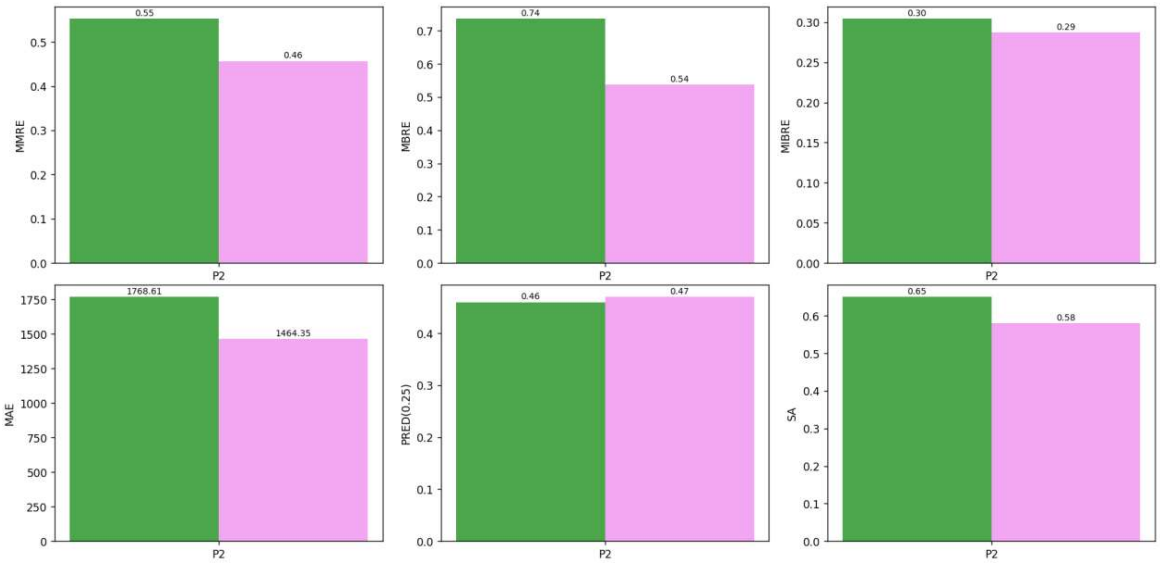


Figure 5-19: The performance of DLMLP compared to DLMLPB based on P2

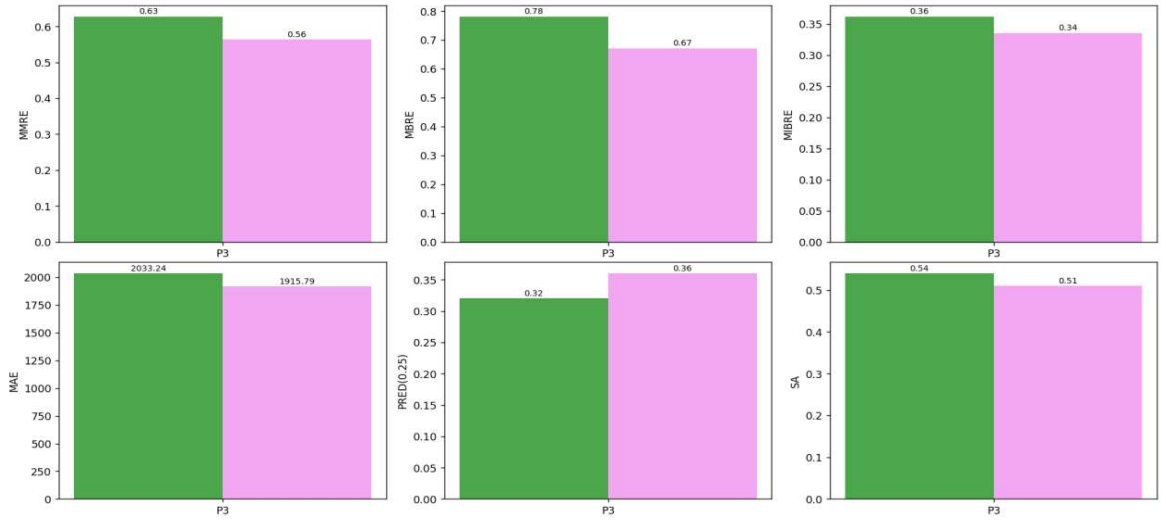


Figure 5-20: The performance of DLMLP compared to DLMLPB based on P3

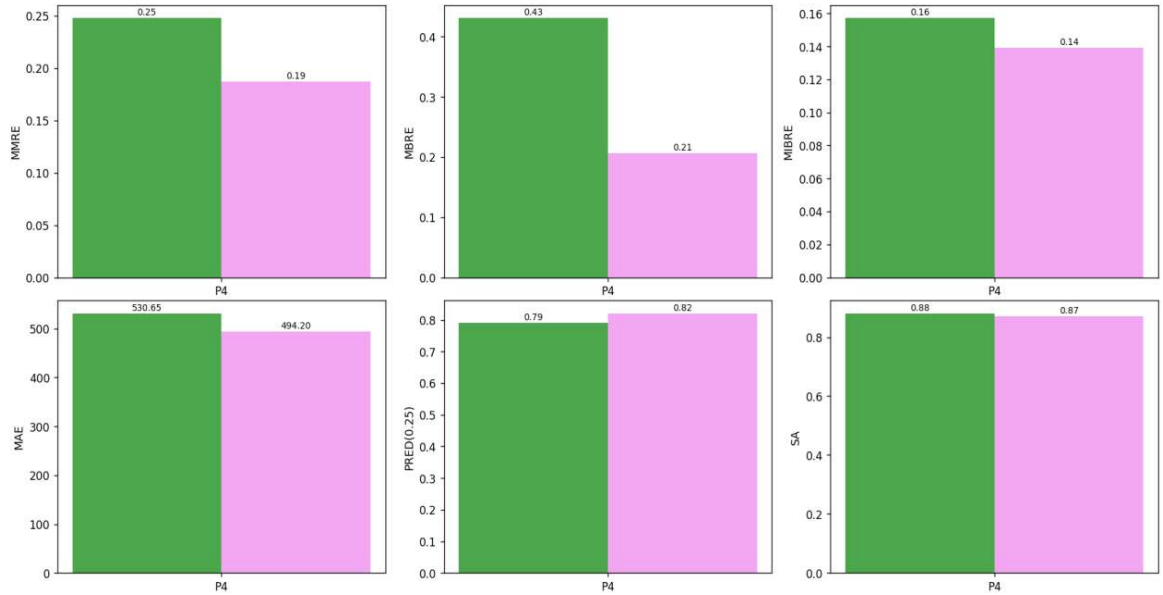


Figure 5-21: The performance of DLMLP compared to DLMLPB based on P4

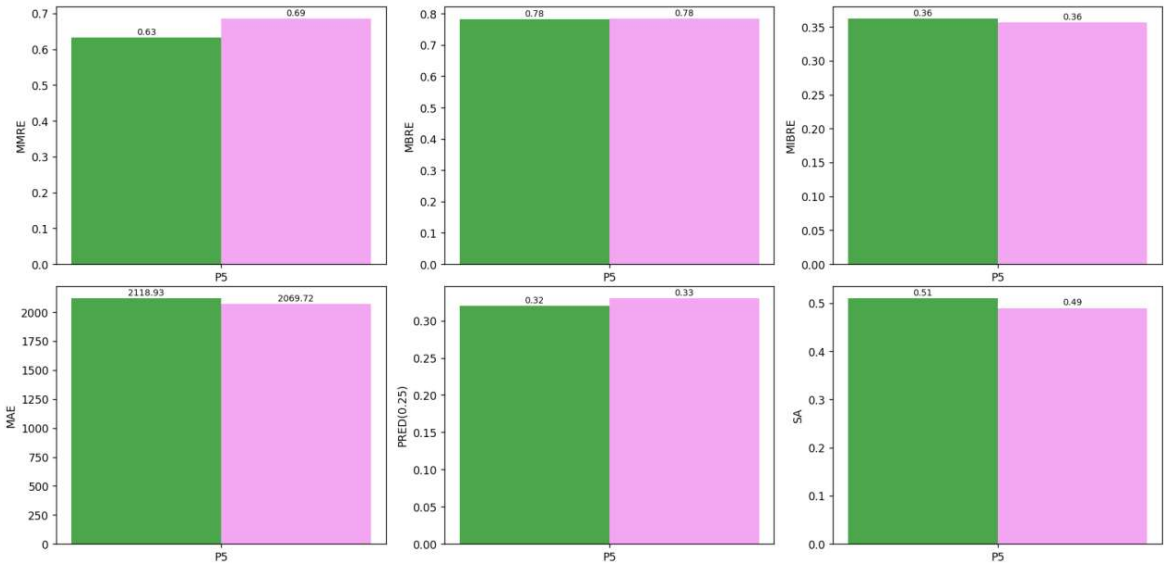


Figure 5-22: The performance of DLMLP compared to DLMLPB based on P5

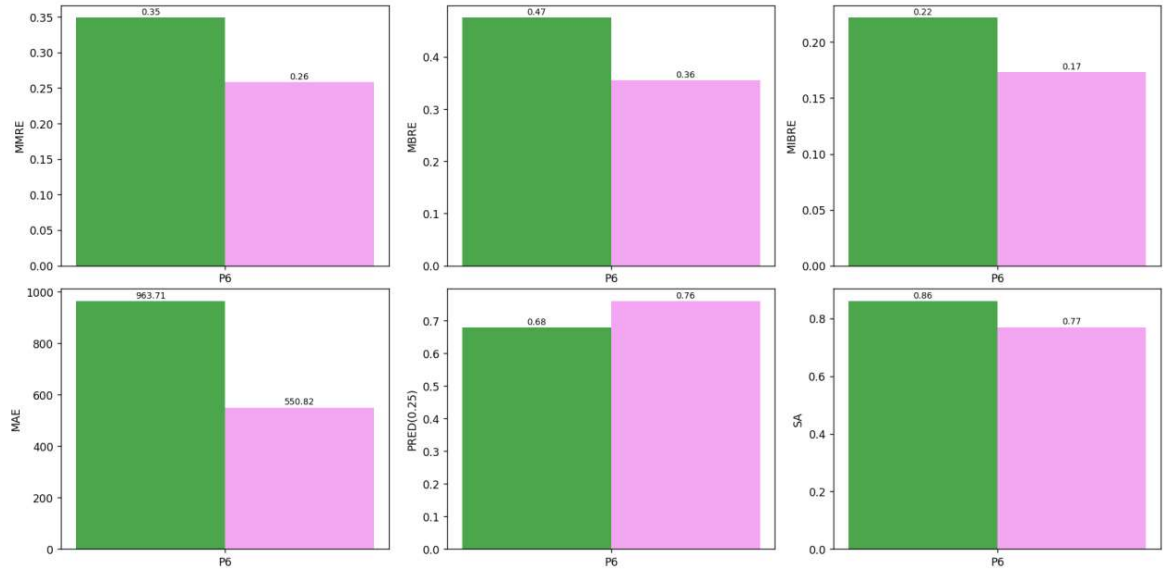


Figure 5-23: The performance of DLMLP compared to DLMLPB based on P6

In a more detailed analysis, DLMLPB consistently outperforms DLMLP across various metrics, including MMRE, MBRE, and MAE. These metrics serve as indicators of superior accuracy in effort estimation. Specifically, in predictors P1 and P5, while some metrics exhibit marginal similarities, the values of MMRE, MIBRE, and MAE obtained from DLMLPB surpass those acquired from DLMLP. In predictors P2, P3, P4, and P6, DLMLPB consistently produces lower MMRE, MBRE, MIBRE, and MAE values than DLMLP. Moreover, the predictive power of DLMLPB, as represented by Pred(0.25), MAE, is notably more significant than that of DLMLP. Additionally, DLMLPB demonstrates enhanced performance in capturing the bias of estimation errors, as evident in its lower MIBRE values. These findings underscore the significance of dataset balancing and effective management of categorical variables within effort estimation.

The findings presented here strongly emphasize the usefulness of employing a balanced dataset when utilizing deep learning models for effort estimation. Researchers and practitioners should consider thoroughly checking and balancing the dataset before training the models to ensure more accurate estimation outcomes. Additionally, adequate handling of categorical variables is critical for achieving reliable estimations. By addressing these aspects, researchers might enhance the accuracy and reliability of effort estimation models, ultimately leading to more informed decision-making and improved project management.

In conclusion, the comparison between DLMLP and DLMLPB has demonstrated that the model with a balanced dataset and effective categorical variable handling consistently yields superior accuracy in effort estimation across various predictors. This finding not only answers the RQ2 that dataset balancing might enhance the predictive accuracy of DLMLP methods in software effort estimation but also emphasizes the critical role of dataset balancing and categorical variable handling in achieving accurate effort estimation. Therefore, researchers are strongly recommended to thoroughly examine and balance their datasets and adopt appropriate techniques for handling categorical variables to improve the accuracy and reliability of their effort estimation models.

5.2.4 Evaluating Ensemble for SDEE: MLR, RF, and DLMLP

The next objective of this research is to compare the performance of MLR, RF, and DLMLP against ensemble models established by incorporating MLR, RF and DLMLP for effort estimation using eleven predictors: P1, P2, P3, P4, P5, P6, P_D, P_A, P_K, P_C, and $P_{Dataset2}$. Figure 5-24 to Figure 5-34 present the performance of MLR, RF, DLMLP, and ensemble models. The blue bar stands for MLR, the orange, the green, and the pink stands for RF, DLMLP, and Ensemble, respectively.

In the context of the P1 predictor (as illustrated in Figure 5-25), the ensemble's performance stands out as it consistently outperforms MLR, RF, and DLMLP. Specifically, the ensemble demonstrates superior accuracy with lower MMRE, MBRE, MIBRE, and MAE values, signifying more precise effort estimation. Furthermore, the ensemble excels in predictive power, as evidenced by higher Pred(0.25) and SA values compared to MLR, RF, and DLMLP. It is worth noting that these trends persist across P2, P3, P5, P_C, P_K, and $P_{Dataset2}$ predictors, as depicted in Figure 5-25, Figure 5-26, Figure 5-28, Figure 5-32, Figure 5-33, and Figure 5-34. Across these diverse predictors, the ensemble consistently maintains its edge in accuracy and predictive prowess, underscoring its effectiveness in effort estimation.

However, when considering predictors such as P4, P6, P_A, and P_D, the ensemble still outperforms MLR and RF regarding MMRE, MBRE, MIBRE, MAE, and SA. Nevertheless, it is worth noting that these values are slightly higher than those obtained from DLMLP, indicating that while the ensemble remains competitive,

DLMLP holds a marginal advantage in these specific cases (such as in P4, P6, P_D, P_C, and P_K).

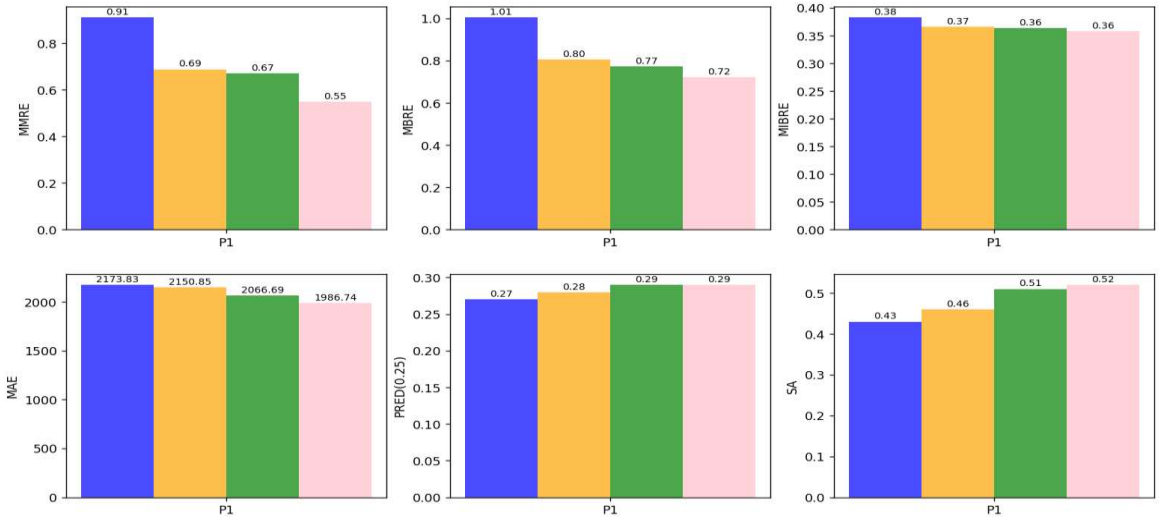


Figure 5-24: The performance of the ensemble model compared to MLR, RF, and DLMLP based on P1

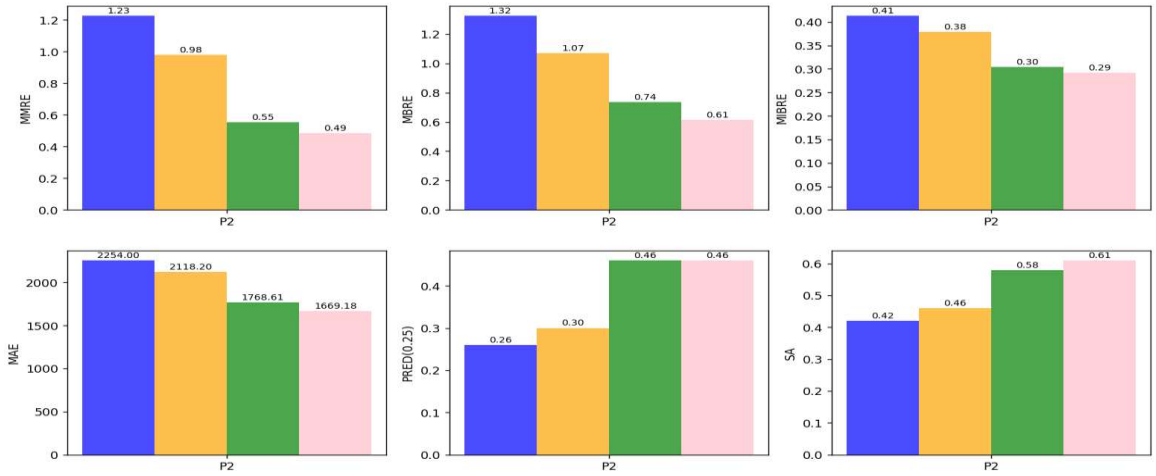


Figure 5-25: The performance of the ensemble model compared to MLR, RF, and DLMLP based on P2

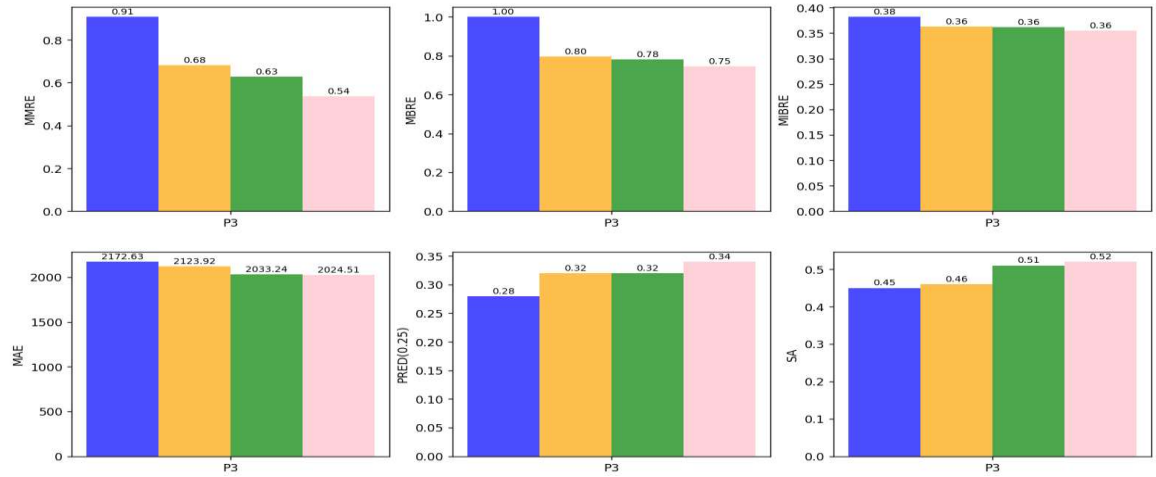


Figure 5-26: The performance of the ensemble model compared to MLR, RF, DLMLP based on P3

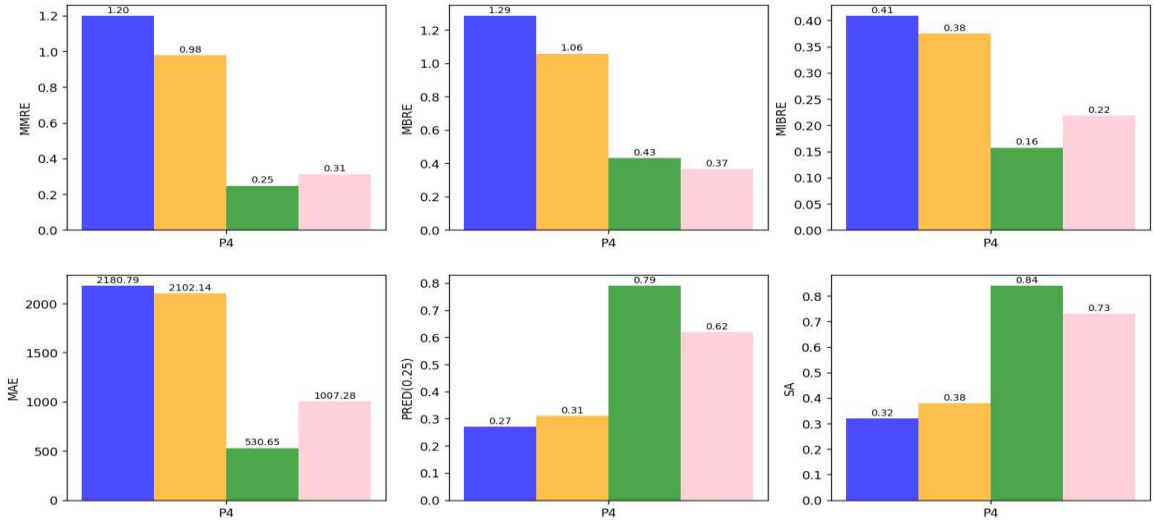


Figure 5-27: The performance of the ensemble model compared to MLR, RF, DLMLP based on P4

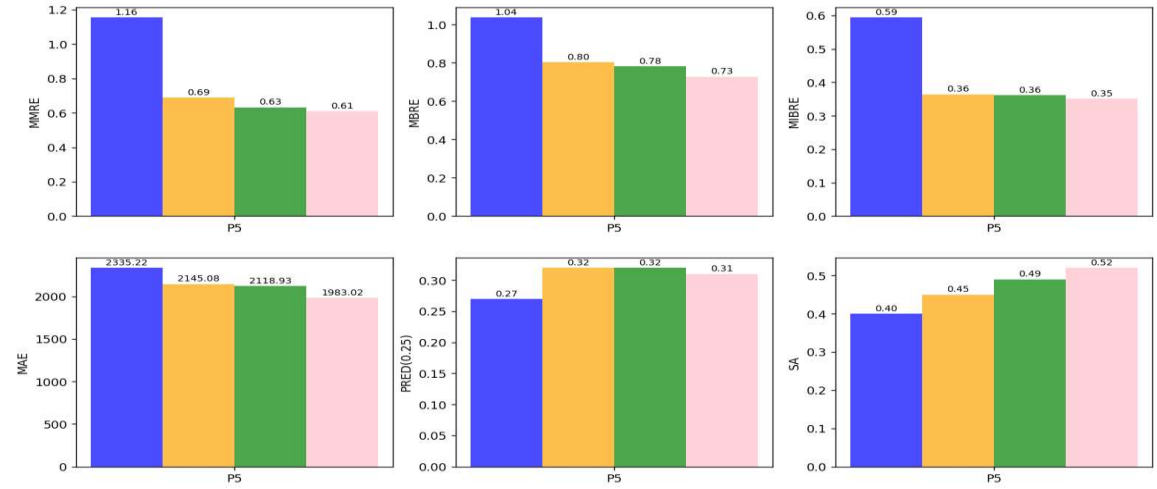


Figure 5-28: The performance of the ensemble model compared to MLR, RF, DLMLP based on P5

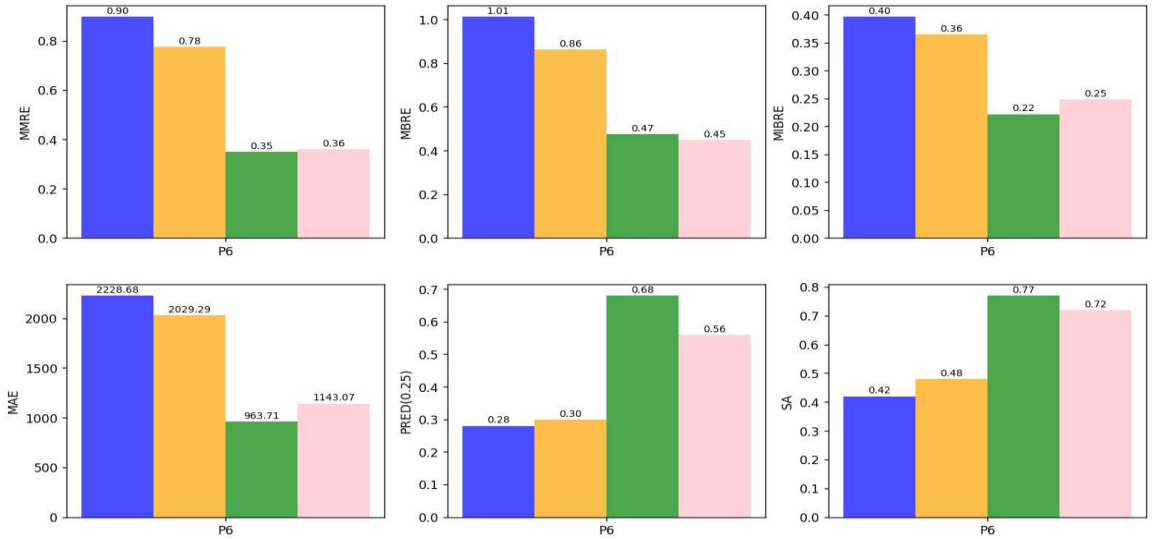


Figure 5-29: The performance of the ensemble model compared to MLR, RF, DLMLP based on P6

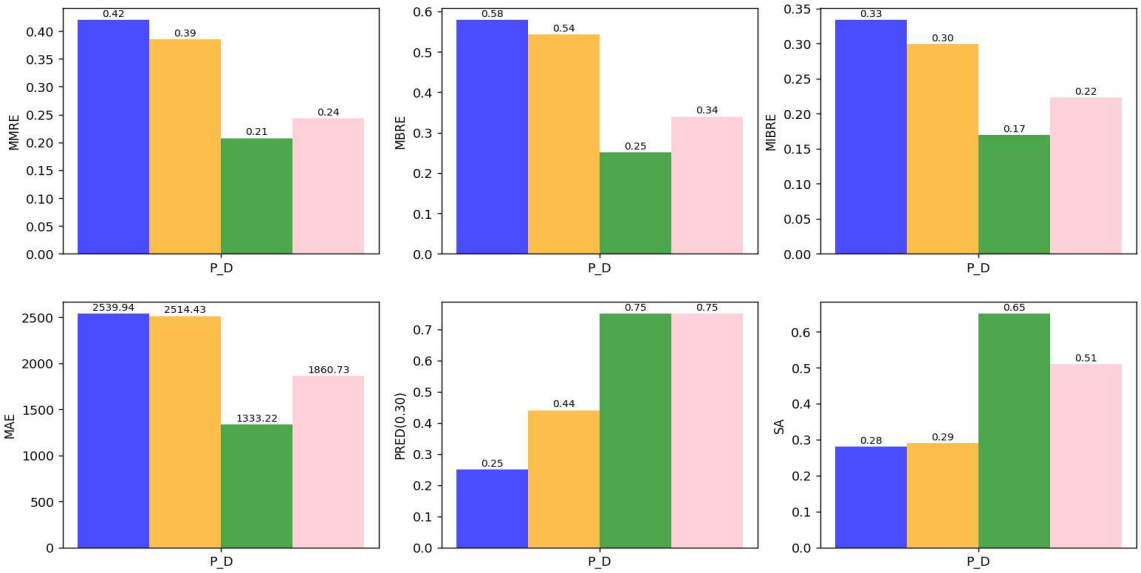


Figure 5-30: The performance of the ensemble model compared to MLR, RF, DLMLP

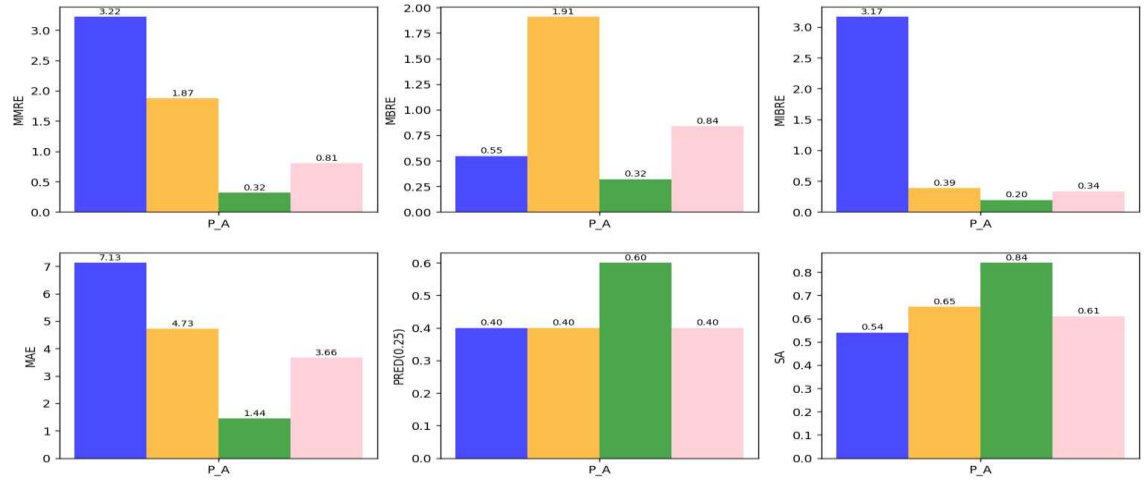


Figure 5-31: The performance of the ensemble model compared to MLR, RF, DLMLP based on P_A

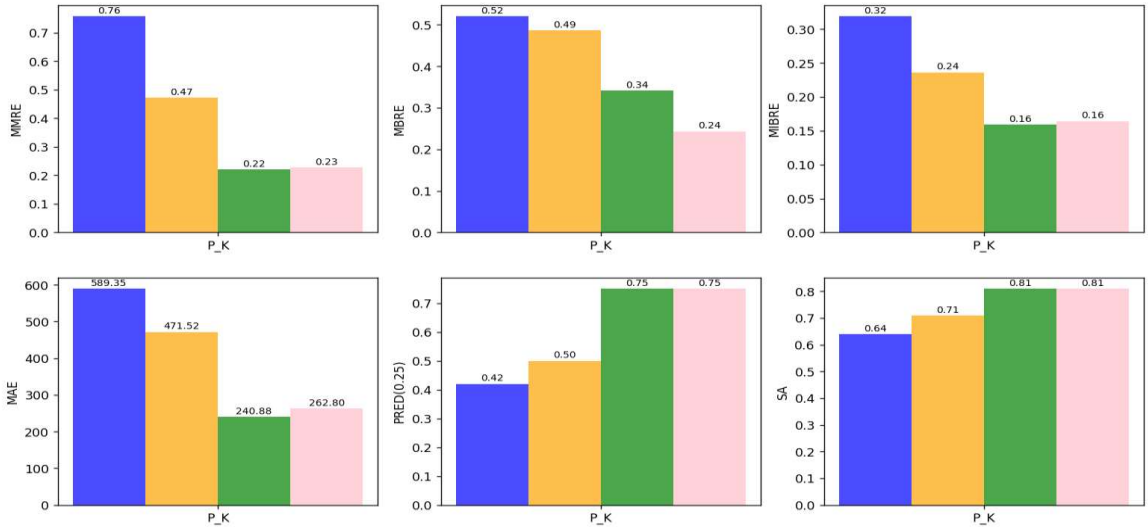


Figure 5-32: The performance of the ensemble model compared to MLR, RF, DLMLP based on P_K

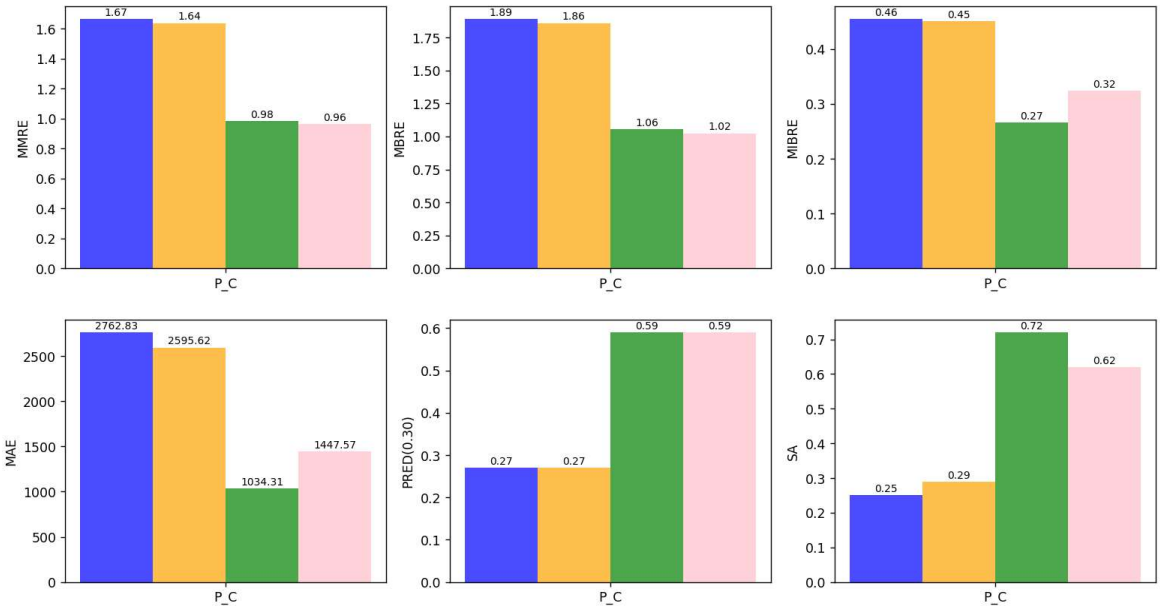


Figure 5-33: The performance of the ensemble model compared to MLR, RF, DLMLP based on P_C

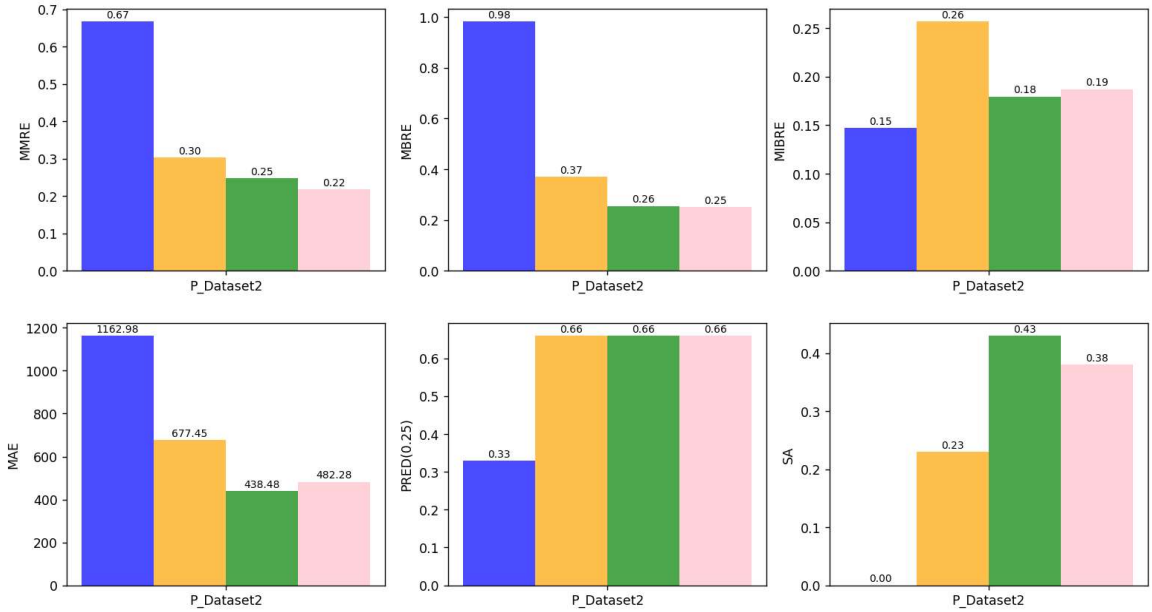


Figure 5-34: The performance of the ensemble model compared to MLR, RF, and DLMLP based on $P_{Dataset2}$

In conclusion, the extensive analysis conducted on various effort estimation scenarios presents valuable insights into the choice between ensemble and individual models. As demonstrated across multiple predictors, ensemble models consistently exhibit superior accuracy and predictive power compared to individual models, including MLR and RF. This finding highlights their potential to enhance effort estimation outcomes significantly. The recommendation to consider ensemble models becomes especially compelling when accuracy and precision are essential in estimating effort. Ensemble models might effectively leverage the strengths of different individual models to provide more accurate and reliable estimations.

However, it is essential to exercise caution when opting for ensemble models. While they generally outperform individual models, DLMLP retains a marginal advantage in specific cases, such as P4, P6, P_A, and P_D predictors. The recommendation is to prioritize ensemble models for effort estimation, particularly when seeking superior accuracy and predictive capabilities across various predictors. Nevertheless, careful consideration should be given to the nature of the task and the individual predictor profiles. When precision is critical, ensemble models offer a robust solution, but DLMLP remains a viable alternative, particularly in cases where its slight advantage aligns with the specific requirements of the effort estimation.

5.2.5 *A Comparative Analysis of Transfer Learning and DLMLP*

The other objective of this study is to compare the accuracy of the transferred model with the DLMLP-based model trained on the new datasets (Albrecht, China, and Dataset 2). This comparison addresses RQ4: "Does DLMLP-based transfer learning offer accuracy over conventional DLMLP?". The study also introduces a pre-trained model based on the ISBSG dataset, providing a comprehensive and reliable foundation for effort estimations.

As mentioned in Section 3.4.5, three transfer learning scenarios, namely TL-Case1, TL-Case2, and TL-Case3, are investigated to assess the effectiveness of transfer learning in effort estimation. In TL-Case1, DLMLP-based models trained on Dataset 1 (the old dataset) are employed to evaluate effort estimation performance on Dataset 2. TL-Case2 involves training DLMLP-based models on 80% of the Albrecht, China, and Dataset 2 datasets and evaluating their performance on the remaining 20%. Finally, TL-Case3 employs DLMLP-based models initially trained on Dataset 1 (pre-trained model) and continues their training on 80% of the new datasets, with an evaluation conducted on the remaining 20% of new datasets.

Figure 5-35, Figure 5-36, and Figure 5-37 illustrate the performance comparison among TL-Case1, TL-Case2, and TL-Case3 across the studied datasets (Albrecht, China, and Dataset 2). The cyan bars represent TL-Case1, while TL-Case2 and TL-Case3 are illustrated in green and grey, respectively. The results obtained from these three cases provide insights into the efficacy of transfer learning. TL-Case 1, obtained from Dataset 2, reveals that the DLMLP model trained on Dataset 1 does not outperform TL-Case2 and TL-Case3. On the other hand, TL-Case3 truly showcases its potential. By combining the strengths of the pre-trained model with further training on the combined datasets, TL-Case3 achieves the lowest MMRE, MBRE, MIBRE, MAE, Pred, and SA values, suggesting superior performance in estimating software effort. These findings collectively emphasize the significance of transfer learning and its ability to enhance the accuracy of effort estimation models in software development projects.

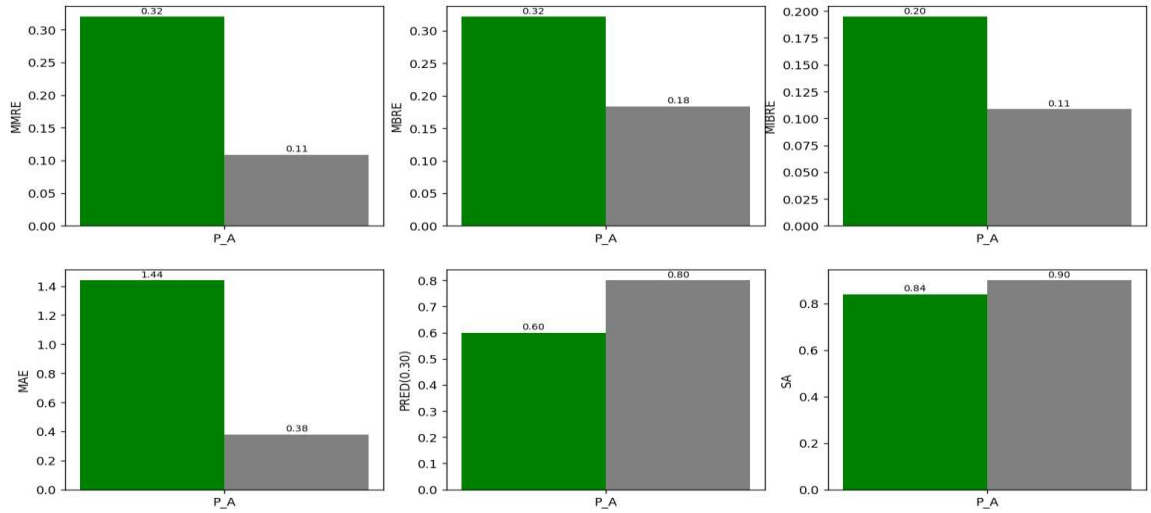


Figure 5-35: The performance of TL Case 2 compared to TL Case 3 based on P_A

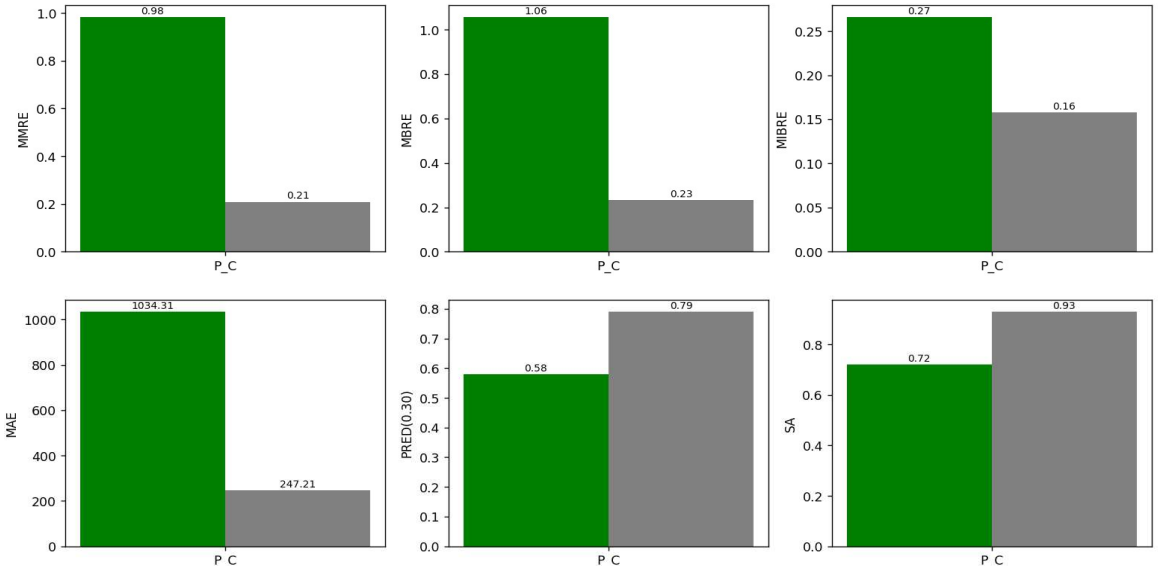


Figure 5-36: The performance of TL_Case 2 compared to TL_Case 3 based on P_C

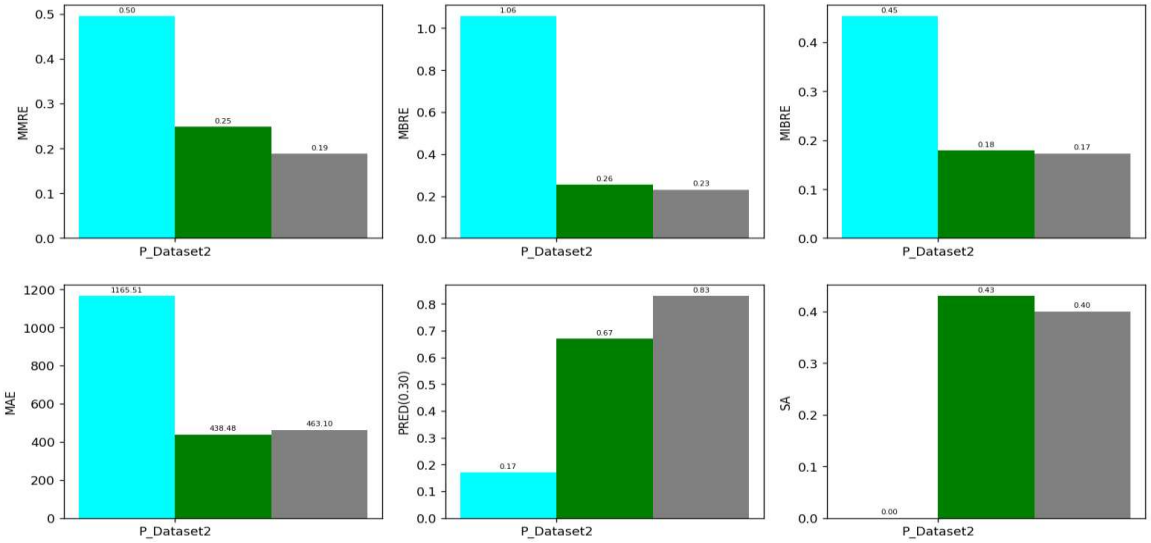


Figure 5-37: The performance of TL_Case 2 compared to TL_Case 3 based on $P_{Dataset2}$

Furthermore, this thesis introduces a pre-trained model constructed upon the ISBSGModel, a neural network architecture derived from the nn.Module class of the PyTorch library. The ISBSGModel is specifically designed to process an input of size input_size and produce a single output value. The model comprises four fully connected linear layers, with the ReLU activation function applied after each layer except for the output layer. The structure of the ISBSGModel is defined in its initialization method, where the four fully connected layers are instantiated (see Figure 3-1). The first layer connects input_size neurons to a hidden layer of 32 neurons. The ReLU activation function is subsequently applied to the output of each layer, introducing non-linearity and facilitating feature extraction. The second layer comprises 64 neurons and connects to the preceding layer through linear transformations. This pattern is repeated in the third layer, which includes 32 neurons. Finally, the output of the third layer is fed into a fourth layer containing a single neuron, representing the final output of the model.

```
class ISBSGModel(nn.Module):
    def __init__(self,input_size):
        super(ISBSGModel, self).__init__()
        self.fc1 = nn.Linear(input_size, 32)
        self.fc2 = nn.Linear(32, 64)
        self.fc3 = nn.Linear(64, 32)
        self.fc4 = nn.Linear(32, 1)
        self.relu = nn.ReLU()
    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.relu(self.fc3(x))
        x = self.fc4(x)
        return x
```

Figure 5-38: The ISBSGModel

In conclusion, transfer learning offers significant advantages in effort estimation by leveraging prior knowledge and improving the accuracy of predictions. Examining three transfer learning scenarios (TL-Case1, TL-Case2, and TL-Case3) has provided valuable insights into the effectiveness of transfer learning techniques within this domain. Notably, TL-Case3, which utilized pre-trained models adjusted on a combined dataset, emerged as the most effective strategy, highlighting the potential of transfer learning to improve effort

estimation accuracy significantly. These findings contribute to the expanding body of research on transfer learning and underscore its relevance in enhancing the performance of machine learning-based models for software effort estimation. Additionally, this thesis presents a pre-trained model based on the ISBSGModel architecture, showcasing the effectiveness of transfer learning for effort estimation. A dedicated library for this pre-trained model offers a convenient and accessible resource for integration into Python projects, ultimately enhancing the accuracy and efficiency of effort estimation in software engineering.

5.2.6 *Exploring the Influence of IS and RS on SDEE*

The influence of IS and RS on software effort estimation using deep learning methods is a crucial aspect to consider. This study aims to analyse the impact of IS and RS on effort estimation accuracy by comparing predictor sets without categorical variables (P1, P2) and those with categorical variables (P3, P4, P5, P6) using DLMLP and DLMLPB models.

Figure 5-39 and Figure 5-40 present the performance of DLMLP and DLMLPB among predictors from P1 to P6. The experimental results presented in these figures provide insights into the influence of IS and RS on effort estimation. When comparing the predictor sets like P1 (AFP), P3 (AFP, IS), and P5 (AFP, IS, RS) for effort estimation, it is essential to analyze the effects of the IS and RS predictors on the accuracy of estimation. Including the IS predictor in P3 leads to significant improvements in accuracy compared to P1. P3 consistently achieves lower MMRE, MBRE, MIBRE, and MAE values, indicating the significant contribution of IS in enhancing estimation accuracy. Moreover, P5, which includes both IS and RS predictors, performs similarly to P3, suggesting that adding RS provides an extra but relatively minor improvement when AFP and IS are already present. These findings underscore the importance of considering IS in effort estimation models, as it captures valuable information related to the inherent complexities and intricacies of the software development process.

Furthermore, when comparing P1 (AFP) against P2 (EI, EO, EQ, EIF, ILF), it becomes evident that P2 consistently outperforms P1 regarding accuracy metrics. Including complexity-related predictors in P2, such as EI, EO, EQ, EIF, and ILF, enhances estimation accuracy. However, it is essential to note that including IS and RS in P4 and P6 further improves estimation accuracy beyond the AFP-based model of P1. These findings underscore the critical role of IS and RS in capturing the complex factors that significantly impact the effort required for software development projects.

Upon examining the experimental results, a difference in performance between predictors P4 and P6 becomes evident. Predictor P4 represents the inclusion of IS as a predictor, while predictor P6 incorporates both IS and RS as predictors. Surprisingly, including RS in predictor P6 results in lower performance than P4 without RS. This result suggests that RS might have a detrimental effect on the model's overall performance.

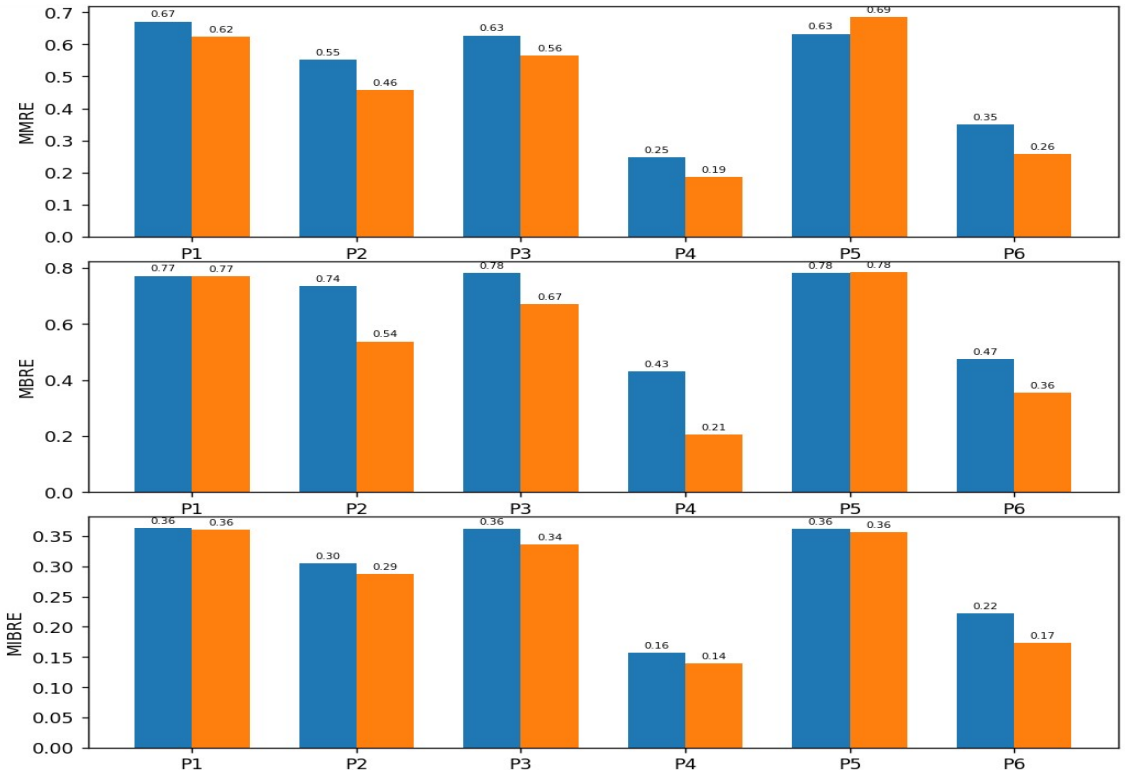


Figure 5-39: The MMRE, MBRE, and MIBRE obtained from DLMLP, DLMLPB among six predictors (P1, P2, P3, P4, P5, P6)

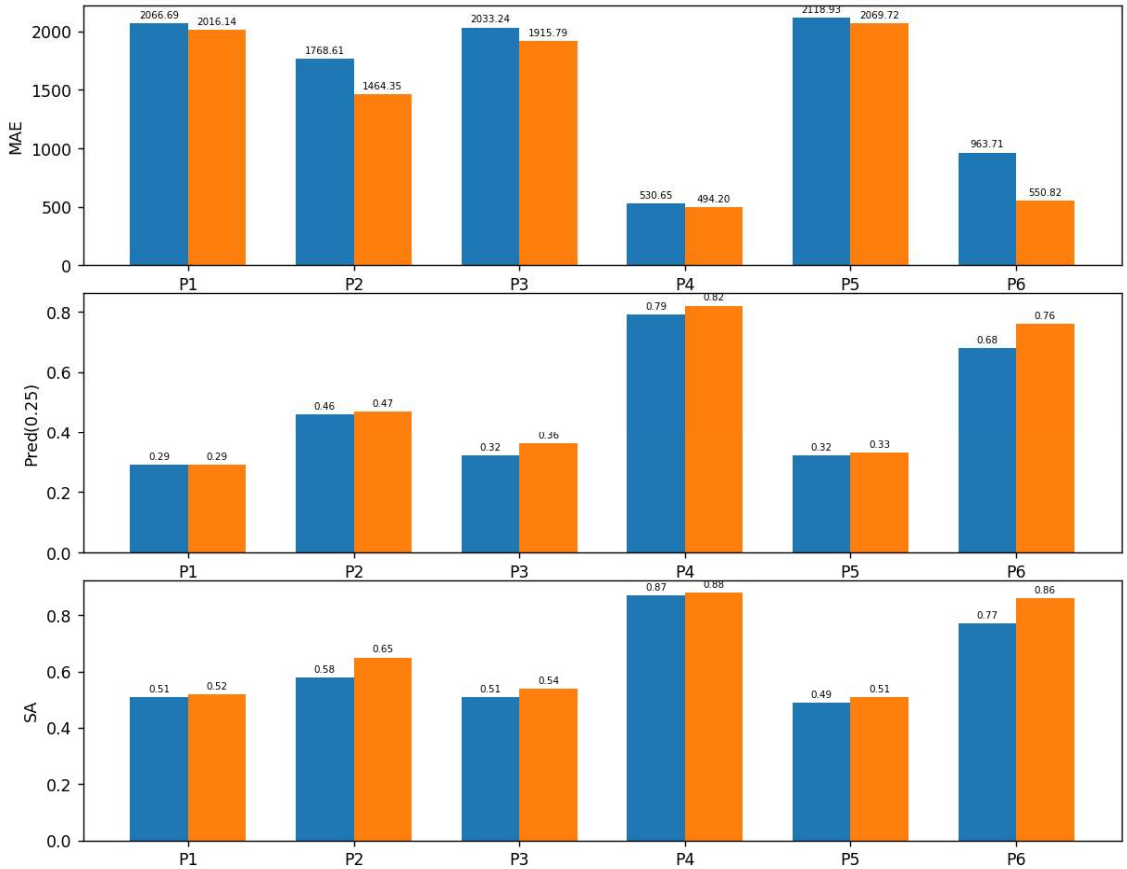


Figure 5-40: MAE, Pred(0.25), and SA obtained from DLMLP, DLMLPB among six predictors (P1, P2, P3, P4, P5, P6)

Interestingly, the experimental results reveal a surprising outcome: the inclusion of RS in predictor P6 leads to lower performance than P4, which does not include RS. This unexpected result suggests a potential negative impact of RS on the overall model performance. Further research is needed to understand the underlying factors contributing to this phenomenon and to explore potential approaches to mitigating the adverse effects of RS on effort estimation accuracy.

In conclusion, including IS and RS predictors consistently enhances the accuracy of effort estimation models. Predictor sets incorporating IS and RS, such as P3 and P5, demonstrate superior performance compared to models solely relying on AFP or complexity factors. These findings highlight the importance of considering IS and RS predictors to capture the intricate nature of software development projects and achieve more precise and reliable effort estimation.

5.3 Evaluation against Hypotheses

Table 5-4 and Table 5-5 present the results of the Mann-Whitney U-tests, which are conducted to examine potential significant differences in mean among various machine learning methodologies: DLMLP, MLR, RF, the ensemble, transfer learning, and DLMLPB. The primary aim of these tests is to ascertain whether statistically significant variations in performance among these methodologies exist. The null hypothesis (H0) stipulates significantly less or equal mean accuracy, while the alternative hypothesis (H1) posits the contrary.

Table 5-4: The Mann-Whitney hypothesis test between DLMLP, MLR, RF, the ensemble and DLMLPB models based on P1, P2, P3, P4, P5, P6

No	Model 1	Model 2	P-value					
			P1	P2	P3	P4	P5	P6
0	DLMLP	MLR	0.00	0.01	0.04	0.00	0.04	0.02
1	DLMLP	RF	0.03	0.04	0.00	0.00	0.04	0.00
2	DLMLP	DLMLPB	0.00	0.02	0.04	0.00	0.00	0.02
3	DLMLP	Ensemble	0.01	0.04	0.01	0.60	0.01	0.70
4	MLR	RF	0.02	0.01	0.04	0.00	0.04	0.04
5	MLR	DLMLPB	0.01	0.00	0.03	0.00	0.03	0.01
6	MLR	Ensemble	0.00	0.00	0.00	0.00	0.00	0.00
7	RF	DLMLPB	0.00	0.02	0.04	0.01	0.04	0.00
8	RF	Ensemble	0.00	0.01	0.00	0.00	0.00	0.03
9	DLMLPB	Ensemble	0.01	0.02	0.01	0.04	0.01	0.01

- **DLMLP vs. MLR, and RF**:

As shown in Table 5-4, the p-values resulting from the comparison of DLMLP with MLR and RF, using predictors from P1 to P6, are consistently below the significance threshold of 0.05. Furthermore, when the study extends this comparison to include other predictors (P_A, P_D, P_C, $P_{Dataset2}$), as presented in Table 5-5, the findings that the p-values remain below 0.05, these findings collectively indicate that DLMLP exhibits substantial variations in mean performance compared to MLR and RF. Consequently, the null hypothesis ($\mu DLMLP \leq \mu MLR \; or \; \mu DLMLP \leq \mu RF$) is rejected, highlighting that the mean accuracy obtained from DLMLP is greater than MLR and RF in software effort estimation.

Table 5-5: The Mann-Whitney hypothesis test between TL-Case2 (DLMLP), MLR, RF, the ensemble and TL-Case3 models based on P_A, P_D, P_C, $P_{Dataset2}$.

No	Model 1	Model 2	P-value				
			P_A	P_D	P_C	P_K	$P_{Dataset2}$
0	TL-Case2	MLR	0.00	0.00	0.00	0.02	0.00
1	TL-Case2	RF	0.02	0.00	0.00	0.01	0.03
3	TL-Case2	Ensemble	0.25	0.57	0.08	0.08	0.00
4	MLR	RF	0.00	0.00	0.02	0.02	0.04
6	MLR	Ensemble	0.02	0.00	0.00	0.00	0.00
8	RF	Ensemble	0.02	0.00	0.00	0.03	0.00
10	TL-Case3	Ensemble	0.00	#	0.01	#	0.00
11	TL-Case3	RF	0.00	#	0.00	#	0.00
12	TL-Case3	MLR	0.00	#	0.00	#	0.00
13	TL-Case3	TL-Case2	0.03	#	0.01	#	0.01

- **DLMLP vs. DLMLPB**:

Balancing the dataset in DLMLPB yields notable improvements, as evidenced by relatively low p-values: 0.00 for P1, 0.02 for P2, 0.04 for P3, 0.00 for P4, 0.00 for P5, and 0.02 for P6 (see Table 5-4). As a result, the alternative hypothesis ($\mu DLMLPB > \mu DLMLP$) is retained in this case, highlighting that the mean accuracy obtained from DLMLPB is greater than DLMLP.

- **The Ensemble vs. DLMLP, MLR, and RF**:

In the comparative analysis of mean performance metrics across the ensemble, MLR, and RF concerning predictors P1 to P6, P_A, P_D, P_C, P_K, and $P_{Dataset2}$, as presented in Table 5-4 and Table 5-5, it is evident that the derived p-values for these comparisons consistently fall below the 0.05 significance threshold. This outcome leads to rejecting the null hypothesis ($\mu ENS \leq \mu MLR, or \; \mu ENS \leq \mu RF$), thereby establishing statistically significant mean differences between the ensemble and MLR and RF, confirming that the mean accuracy attained from the

ensemble is better than those obtained from MLR and RF. Moreover, when examining the mean accuracy achieved from the ensemble approach in comparison to DLMLP, it is observed that predictors P1, P2, P3, P5, P_A, $P_{Dataset2}$ exhibit p-values fall below 0.05 except P4, P6, P_D, P_C, and P_K. Consequently, the null hypothesis ($\mu ENS \leq \mu DLMLP$) might be rejected.

In conclusion, this observation suggests that, in general, the mean accuracy obtained from the ensemble is more significant than that obtained from DLMLP, MLR, and RF in software effort estimation.

- **Transfer Learning (TL-Case3) vs. DLMLP**:

The transfer learning model **(TL-Case3)** shows significant mean performance disparities compared to the DLMLP, as the p-values obtained from those models are less than 0.05 in P_A, P_C, and $P_{Dataset2}$. The null hypothesis ($\mu TL \leq \mu DLMLP$) is rejected, indicating that the mean accuracy obtained from the transfer learning model **(TL-Case3)** is significantly greater than DLMLP.

- **Influence of IS and RS in the accuracy of effort estimation**:

Analyzing the results in Table 5-6 provides insights into RQ5, which aims to determine whether the categorical variables IS and RS significantly influence effort estimation accuracy. These values indicate that both. βIS and γRS for each predictor are not equal to 0. Consequently, we reject the null hypothesis ($\beta IS = \gamma RS = 0$) and accept the alternative hypothesis, suggesting that the categorical variables (IS and RS) influence the accuracy of effort estimation.

Table 5-6: The Regression coefficient for IS and RS obtained from MLR.

Predictors	βIS	γRS	Description
P5	-2.11	-358.41	$\beta IS \neq \gamma RS \neq 0$
P6	24.94	-804.60	$\beta IS \neq \gamma RS \neq 0$

Upon examining the coefficients presented in this table, it is observed that the magnitude of the regression coefficients for RS (-358.41 and -804.60) is notably smaller than those for IS (-2.11 and 24.94). This phenomenon indicates that the variable RS might have a weaker impact on effort estimation accuracy than IS. This observation, in turn, suggests that the IS variable may exert a more substantial influence on the accuracy of effort estimation models than the RS variable. The differences in the magnitudes of these coefficients provide valuable insights into the importance of these categorical variables in influencing effort estimation accuracy. This information contributes to a deeper understanding of the role of Industry Sector and Relative Size in the accuracy of effort estimation models and informs decision-making regarding model development and feature selection.

The Mann-Whitney U-test results suggest that the DLMLP significantly outperforms MLR and RF regarding mean performance. Balancing the dataset in

DLMLPB does yield significant improvements in mean performance compared to DLMLP. The transfer learning model shows significant differences in mean performance compared to DLMLP. However, the ensemble approach demonstrates comparable mean performances to DLMLP. Notably, the results also shed light on the influence of categorical variables, as evidenced by the regression coefficients obtained for IS and RS from the MLR model. In this regard, while the regression coefficient of IS demonstrates a moderate effect on the accuracy, the regression coefficient for RS indicates a relatively minor influence. These insights collectively provide crucial guidance for practitioners in selecting suitable machine learning methodologies, underlining the significance of both model performance and the influence of categorical variables for well-informed decision-making in practical scenarios.

5.4 Model Explainability Findings - Analysis of Predictor Contributions

5.4.1 LIME

This study explores the interpretation of predicted effort values generated by a specific model, DLMLP-P6 (DLMLP trained on P6). This model contains input features EO, EIF, ILF, EQ, EI, IS, and RS, where EO, EIF, ILF, EQ, and EI are measured in function points, and IS, RS are categorical variables, a predicted effort is measured in person-hours.

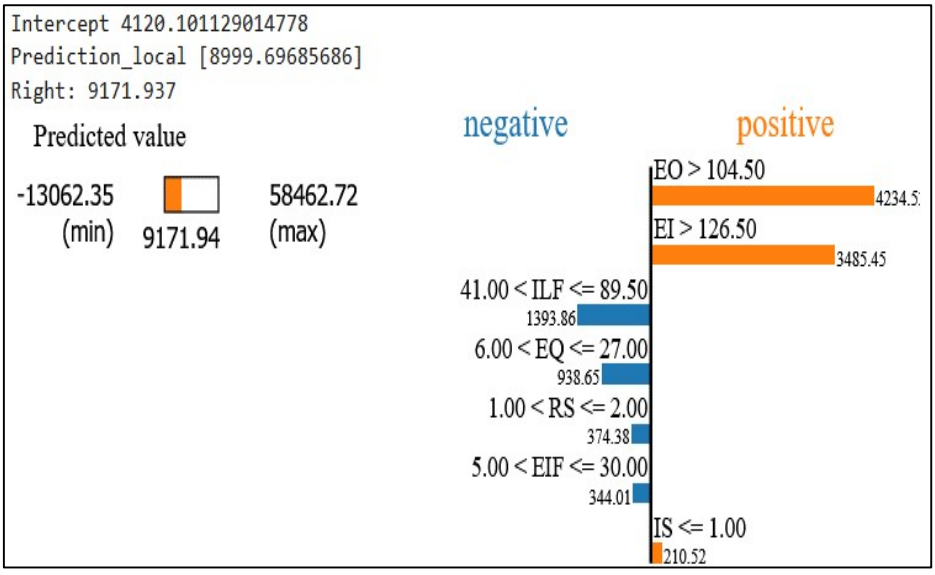


Figure 5-41: Interpreting the predicted effort values obtained from DLMLP-P6

This method employed for interpretation is LIME, as illustrated in Figure 5-41. Through applying LIME, the objective is to elucidate the roles of IS and RS within the effort estimation process across these models. Subsequently, an exhaustive analysis of the outcomes derived from LIME's interpretations is presented below:

- LIME predicts approximately 8999.69 (person-hours) for a specific instance, while the actual prediction is 9171.94 (person-hours). The range of predicted effort (person-hours) spans from -13062.35 (minimum) to 58462.72 (maximum), indicating a wide variance in predictions.
- The feature contributions in this analysis offer valuable insights into the factors influencing the predicted effort. Notably:
    - ✓ ILF: When ILF falls within the range of 41 to 89.5, it negatively affects the predicted effort by contributing to -1393.86 person-hours. This result suggests that an increase in ILF within this range correlates with reducing the predicted effort.
    - ✓ EQ: Falling within the range of 6 to 27, EQ negatively affects the predicted effort, contributing -938.65 person-hours. This result implies that a moderate number of EQ might decrease the predicted effort compared to extreme values.
    - ✓ RS: With values between 1 (M1) and 2 (M2), RS negatively influences the predicted effort by contributing -374.38 person-hours. This result indicates that a specific range of RS values tends to decrease the predicted effort.
    - ✓ EIF: When EIF falls between 5 and 30, it negatively impacts the predicted effort, contributing to -344.01 person-hours. This finding suggests that an increase in EIFs is associated with a decrease in predicted effort within this range.
    - ✓ EO: When EO exceeds 104.50, it positively influences the predicted effort with a contribution of 4234.5 person-hours. This finding indicates that more external outputs in the project increase the predicted effort.
    - ✓ EI: An EI value more excellent than 126.50 positively contributes 3485.45 person-hours, signifying that an elevated count of external inputs increases predicted effort.
    - ✓ IS: When IS is 0.0 (Banking) or 1.0 (Communication), it positively contributes 210.52 person-hours, suggesting that a smaller interface size is associated with higher predicted effort.

The LIME results suggest that EI, EO, and IS are the key features impacting the predicted effort. These variations in LIME's interpretations emphasize the importance of comparing results. LIME's visualizations further aid in understanding these nuanced interpretations, enabling a more comprehensive analysis of feature influence.

In conclusion, this study underscores the complexity of predictive models in software effort estimation and the need for interpretability methods like LIME. The differences in feature contributions across models highlight the necessity of thorough evaluation and comparison. This analysis might guide practitioners in selecting the most suitable model for their projects, considering the strengths and weaknesses of each. Moreover, exploring model interpretability methods like

LIME enhances understanding of complex predictive models and their practical applications in software effort estimation.

5.4.2 SHAP

Through applying the SHAP technique to DLMLP-P6, the thesis aims to gain comprehensive insights into the contributions of individual features (EI, EO, EQ, EIF, ILF, IS, and RS) within these predictive models for effort estimation. SHAP values provide a valuable means to assess the significance of each feature in predicting the effort required for diverse software development projects. By examining these SHAP values, we might better understand the relative impact of each feature on the model's predictions, facilitating the identification of critical attributes that influence the effort estimation process.

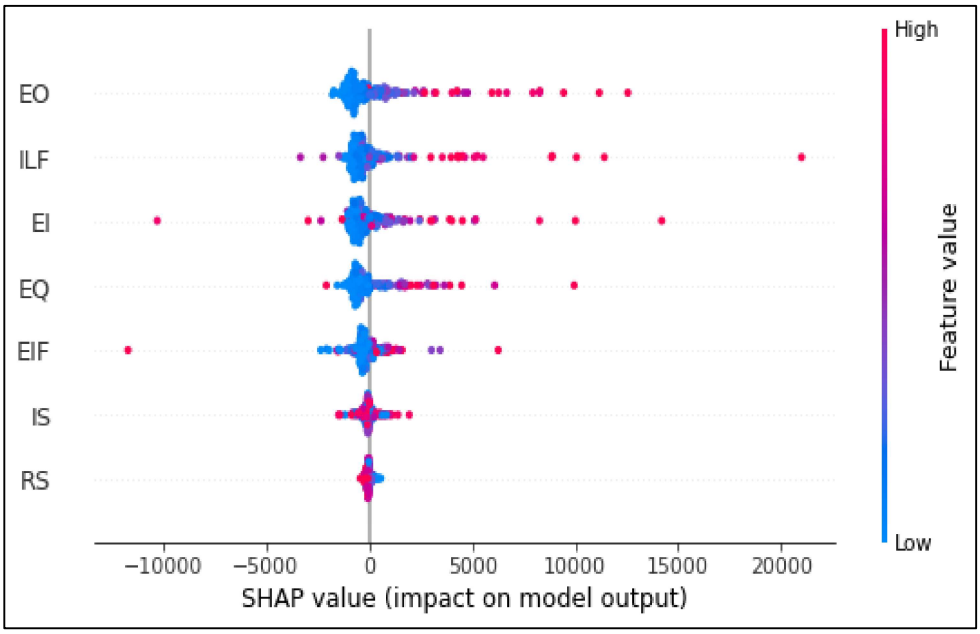


Figure 5-42: The contributions of each feature in DLMLP-P6

Figure 5-42 presents the feature contributions obtained from Dataset 1 of the testing dataset for DLMLP-P6. These contributions reveal intriguing patterns. EI, EO, EQ, ILF, and EIF feature contributions exhibit consistent trends across all three models. These features exhibit significantly positive contributions when their values are high, while they display low negative contributions when their values are low. This consistency highlights their crucial roles in effort estimation within these models.

- The EI, EO, EQ, ILF, and EIF features demonstrate relatively consistent trends, with a positive contribution associated with higher values and a minor negative contribution linked to lower values across all three models, while IS has a slight positive contribution.
- Notably, the feature RS appears to have no significant contribution to predictions, irrespective of its value.

In summary, the analysis of feature contributions using SHAP values offers valuable insights into the impact of individual features on the predictions made by DLMLP-P6. The consistency in the contributions of EI, EO, EQ, ILF, and EIF suggests their critical roles in effort estimation across these models. However, the negligible contribution of RS merits further exploration to comprehend its influence on the model's predictive performance. These findings contribute a deeper understanding of interpretability and feature importance in software effort estimation using deep learning models.

6. CONTRIBUTIONS

This section summarises contributions and the implications for practice and research.

6.1 Summary of Contributions

This study seeks to provide specific contributions to the domain of SDEE by addressing several key research areas:

- **Comparative Analysis of Predictive Models:**

This research extensively evaluates predictive models across distinct predictors, specifically MLR, RF, and DLMLP. The objective is to determine the superior model for SDEE. Findings reveal that DLMLP consistently surpasses MLR and RF across multiple performances, including MMRE, MBRE, MIBRE, MAE, Pred(0.25), and SA. Consequently, DLMLP emerges as the preferred predictive model for SDEE.

- **Impact of Dataset Balancing on Accuracy:**

This study examines the influence of dataset-balancing techniques and the handling of categorical variables in deep learning models by comparing DLMLP (unbalanced dataset) to DLMLPB (balanced dataset). The outcomes indicate that DLMLPB consistently outperforms DLMLP across all predictor sets, underscoring the significance of dataset balancing and effective categorical variable management in enhancing estimation accuracy.

- **Ensemble Models for SDEE:**

The research evaluates ensemble models that combine MLR, RF, and DLMLP to assess their effectiveness in SDEE across various predictor sets. The findings demonstrate that ensemble models, mainly when precision is pivotal, exhibit superior performance compared to individual models. Nonetheless, it is noteworthy that DLMLP retains a slight advantage in specific scenarios, suggesting that the choice between ensemble models and DLMLP should hinge on the specific requirements of the effort estimation.

- **Transfer Learning for Enhanced Accuracy:**

The study investigates the efficacy of transfer learning in the context of SDEE by comparing DLMLP-based models trained on different datasets. The results

emphasize the potential of transfer learning, particularly when employing a pre-trained model and fine-tuning it on the new dataset. This approach consistently outperforms models exclusively trained on new data. Using a pre-trained model is recommended based on the findings of outcomes. This model leverages prior knowledge and extensive training on large datasets, which might significantly enhance the predictive capabilities. By starting with a pre-trained model as a foundation, researchers and practitioners might save valuable time and resources that would otherwise be required for extensive model training.

- **Influence of Categorical Variables:**

This research examines the impact of IS and RS predictors on effort estimation accuracy across diverse predictor sets. The findings underscore the imperative nature of incorporating IS predictors into effort estimation models, as they encapsulate crucial information regarding the intricacies of the software development process. While RS predictors exhibit some influence on accuracy, further investigation is warranted to comprehend their nuanced impact.

In summation, this study contributes significantly to the domain of SDEE by providing precise insights and recommendations. It delivers clear guidance regarding predictive model selection, dataset balancing, ensemble models' utilisation, transfer learning's advantages, and the pivotal role of IS predictors in improving estimation accuracy. These findings empower software development teams to conduct more precise estimations, enhancing project planning and management within the software industry.

6.2 Implications for Practice

The practical implications of this thesis carry considerable significance for the software industry. Predictive models might improve effort estimation accuracy by balancing datasets based on categorical variables or applying transfer learning based on pre-trained models. The study also applied LIME and SHAP to deeply analyse insights into the black-box of predictive models. The findings reveal that EI, EO, and IS positively impact SDEE among EI, EO, EQ, EIF, ILF, IS, and RS.

In other words, the research findings strongly suggest that adopting the pre-trained model and integrating deep learning methods with balancing categorical variables might significantly improve effort estimation performance in practice. By incorporating these innovative approaches into practical settings, project management processes might be streamlined, resource utilization might be optimized, and higher-quality software products might be delivered to customers.

The implications of this thesis extend beyond theoretical constructs, offering actionable solutions that directly impact software development. These novel approaches empower practitioners with enhanced estimation capabilities and may revolutionize project outcomes, ensuring greater efficiency and effectiveness in the software development process. In doing so, the software industry benefits

from improved project success rates, reduced resource wastage, and heightened customer satisfaction, reinforcing the research's value and practicality.

6.3 Implications for Research

The research presented in this thesis opens avenues for future investigation in SDEE. The effectiveness of pre-trained models in software engineering is highlighted, encouraging further exploration of ensemble and transfer learning techniques to improve effort estimation accuracy. Moreover, LIME and SHAP are known and commonly used for prediction model interpretation. The study's use of them to interpret and understand the contribution of different features within proposed effort estimation models offers valuable insights into the underlying decision-making process.

In the coming years, prospective research endeavours may concentrate on balancing datasets, adopting transfer learning using the pre-trained model, or applying an ensemble approach by integrating several models such as MLR, RF, and DLMLP. Moreover, the integration of LIME and SHAP with other interpretability techniques to attain comprehensive insights into model predictions could be explored.

Overall, the findings of this thesis lay the foundation for future research endeavours in the domain of software development effort estimation, advancing the field towards more accurate and reliable prediction models.

7. THREAT AND VALIDATION

In this study, the assessment of the proposed method's validity encompasses both internal and external aspects. Internal validity, crucial for concluding experimental research, is addressed by implementing the k-fold cross-validation method. This approach ensures a rigorous validation of the statistical sample, enhancing the accuracy of the evaluation process.

External validity, which focuses on the applicability of the results in different settings, is evaluated to ascertain the prediction ability of the proposed method. The ISBSG repository August 2020 R1 dataset is utilized to achieve this research. This dataset comprises various software projects from various organizations worldwide, each characterized by distinct features, fields, and sizes. Additionally, the proposed model is assessed using other datasets such as Albrecht, Deshairnais, Kitchenham, and China, further enhancing the robustness of the evaluation.

Various evaluation criteria are employed to verify the performance of effort estimation obtained from the proposed methods, including MMRE, MBRE, MIBRE, MAE, Pred(0.25)/Pred(0.30), and SA. These evaluation criteria have been recognized in previous publications [149], [150] as unbiased and reliable for accuracy assessment. Consequently, the experimental results of this study are

highly generalizable, reflecting the method's performance across diverse datasets and real-world scenarios.

In conclusion, the study adopts a comprehensive approach to validate the proposed method's effectiveness. Using k-fold cross-validation ensures internal validity while incorporating multiple datasets, including the ISBSG repository and others, enhances external validity. The study provides reliable and generalizable results by employing established and unbiased evaluation criteria, making it a valuable contribution to software project prediction.

8. CONCLUSION

This section summarises the thesis and presents the future directions for research.

8.1 Summary of the Thesis

The thesis assesses effort estimation performance using three distinct methodologies: MLR, RF, and DLMLP. The experimental procedures encompass a diverse array of datasets, with the primary dataset originating from the ISBSG dataset release of 2020, complemented by supplementary datasets obtained from the other datasets. The study considers eleven predictors: six predictor combinations denoted as P1 through P6 obtained from ISBSG, and individual predictors P_A, P_D, P_C, P_K, and $P_{Dataset2}$. The research outcomes provide conclusive responses to RQ1; the findings indicate that DLMLP consistently delivers superior accuracy in SDEE compared to MLR and RF. Additionally, this study introduces and evaluates three baseline models: ANN-based models, SWR-based models, and IFPUG-FPA methodology. The comparative analysis between DLMLP and these baseline models reveals that the DLMLP model consistently outperforms them across various performance metrics, including MMRE, MBRE, MIBRE, MAE, Pred(0.25), and SA.

Additionally, this study explores the impact of two categorical variables from the ISBSG dataset, the industry sector and relative size factor, alongside factors from the FPA as input features. These variables are carefully selected to understand their influence on the DLMLP, MLR, and RF models. As illustrated in Table 4-4, the industry sector includes various categories, such as banking and government, each characterized by differing data distributions. The research poses RQ2 and endeavours to rectify dataset imbalance using the class-weight approach. The ensuing examination contrasts the performance of DLMLP on the original dataset against that on the balanced dataset (DLMLPB). The results may offer insights into whether the dataset-balancing approach, as pursued in this study, outperforms its unbalanced dataset.

The thesis also investigates ensemble techniques that combine the previously examined models, namely MLR, RF, and DLMLP. Specifically, a stacking

ensemble approach is applied to MLR and RF, with XGBoost being chosen as the final estimator. Subsequently, the results of this ensemble process are further combined with DLMLP using a voting (average) method for regression. These experiments encompass eleven predictor variables denoted as P1 to P6, P_A, P_D, P_C, P_K, and $P_{Dataset2}$. In general, the ensemble approach demonstrates potential superiority over each model in most instances. These findings have the potential to provide insights into addressing RQ3 by suggesting that the ensemble approach may outperform individual models.

The study examines three scenarios involving the pre-trained model. In the first scenario, the model is applied for predictions on a new test dataset called TL-Case1. The second scenario involves using the architecture of DLMLP to construct a model on a new training dataset, TL-Case2. Notably, the prediction results on the new test dataset are superior in TL-Case2 compared to TL-Case1. The final scenario uses the pre-trained model and further trains it on a new training dataset called TL-Case3, commonly called the transfer learning model. When comparing the prediction outcomes on the new dataset from TL-Case3 with those of TL-Case1 and TL-Case2, TL-Case3 demonstrates the best results. This observation might address RQ4 by suggesting that transfer learning enhances prediction model accuracy. Based on these findings, the research has also constructed a library (https://github.com/huynhhoc/effort-estimation-by-using-pre-trained-model). While not necessarily the ultimate version, researchers might utilize or upgrade it to enhance the accuracy of the pre-trained model.

Last but not least, this thesis looks at how IS and RS affect effort estimation accuracy. As discussed in Section 5.2.6, IS demonstrates a slight effect on the accuracy, while RS has a relatively small effect. This observation is supported by analyses using LIME and SHAP, which might answer for RQ5 that IS has a positive effect on effort estimation, while RS has a negative one. The findings obtained from LIME and SHAP also reveal that EI and EO positively impact effort estimation compared with EQ, EIF and ILF.

In summary, this thesis presents a comprehensive analysis of predictive models, dataset balancing techniques, ensemble methods, transfer learning, and the influence of categorical variables in software development effort estimation. The findings provide valuable insights for practitioners and open doors to future research in this field.

8.2 Future Directions for Research

In future research, it would be valuable to explore additional categorical variables within the ISBSG dataset and analyze the distribution patterns of these variables. Furthermore, addressing the challenge of imbalanced data, particularly in the context of categorical factors, is crucial for deep learning models. Investigating techniques to mitigate data imbalance, such as oversampling, undersampling, data augmentation, or using advanced algorithms tailored for

imbalanced datasets, might enhance effort estimation accuracy and ensure the reliability of predictions across various industry domains.

Furthermore, the current pre-trained model is based on a multilayer perceptron architecture. Future developments could involve upgrading the pre-trained model by selecting the most suitable architecture from various options, including multilayer perceptrons, convolutional neural networks, and recurrent neural networks. Moreover, for future developments, extending the scope of the pre-trained model to incorporate a broader array of categorical variables from the updated version of the ISBSG dataset could yield valuable insights for practitioners and researchers.

Moreover, as presented in the preceding sections, ensemble methods consistently outperform individual models in predictive accuracy. Nonetheless, there are instances where a few predictors are not clear about this trend yet. In the future, combining highly effective individual models across diverse datasets may be beneficial to verify the optimal ensemble approach for effort estimation predictions.

Last but not least, it is essential to underscore the significance of employing LIME and SHAP techniques for comprehensively evaluating the contributions of relevant attributes within a model. By applying LIME and SHAP, practitioners and researchers might thoroughly understand how various features impact the proposed effort estimation models. In future research endeavours involving predictive models, practitioners and researchers are strongly encouraged to systematically apply LIME and SHAP methodologies to assess model attributes' contributions comprehensively. This method will provide a deeper insight into the factors influencing model outcomes and guide feature selection and model refinement.

9. REFERENCES

[1] A. J. Albrecht, “Measuring application development productivity,” in *Proc. Joint Share, Guide, and IBM Application Development Symposium, 1979*, 1979.

[2] N. Agarwal, A. Sondhi, K. Chopra, and G. Singh, “Transfer Learning: Survey and Classification,” in *Smart Innovations in Communication and Computational Sciences*, S. Tiwari, M. C. Trivedi, K. K. Mishra, A. K. Misra, K. K. Kumar, and E. Suryani, Eds., Singapore: Springer Singapore, 2021, pp. 145–155.

[3] E. Kocaguneli, T. Menzies, and E. Mendes, “Transfer learning in effort estimation,” *Empir Softw Eng*, vol. 20, no. 3, pp. 813–843, 2015, doi: 10.1007/s10664-014-9300-5.

[4] L. L. Minku, “Multi-stream online transfer learning for software effort estimation: Is it necessary?,” in *Proceedings of the 17th International*

Conference on Predictive Models and Data Analytics in Software Engineering, 2021, pp. 11–20.

[5] L. L. Minku and X. Yao, "Can cross-company data improve performance in software effort estimation?," in *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*, 2012, pp. 69–78.

[6] V. Van Hai, H. Le Thi Kim Nhung, and H. T. Hoc, "A review of software effort estimation by using functional points analysis," in *Proceedings of the Computational Methods in Systems and Software*, Springer, 2019, pp. 408–422.

[7] ISBSG, "ISBSG," *International Software Benchmarking Standards Group, Release R1*, 2020.

[8] C. López-Martín, A. Chavoya, and M. E. Meda-Campaña, "Use of a feedforward neural network for predicting the development duration of software projects," in *2013 12th International Conference on Machine Learning and Applications*, IEEE, 2013, pp. 156–159.

[9] C. López-Martín, A. Chavoya, and M. E. Meda-Campaña, "Use of a feedforward neural network for predicting the development duration of software projects," in *2013 12th International Conference on Machine Learning and Applications*, IEEE, 2013, pp. 156–159.

[10] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, "From Function Points to COSMIC - A Transfer Learning Approach for Effort Estimation," in *Product-Focused Software Process Improvement*, P. Abrahamsson, L. Corral, M. Oivo, and B. Russo, Eds., Cham: Springer International Publishing, 2015, pp. 251–267.

[11] E. Kocaguneli, T. Menzies, and E. Mendes, "Transfer learning in effort estimation," *Empir Softw Eng*, vol. 20, no. 3, pp. 813–843, 2015, doi: 10.1007/s10664-014-9300-5.

[12] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods," *Journal of Software: evolution and Process*, vol. 31, no. 10, p. e2211, 2019.

[13] S. Shukla and S. Kumar, "Applicability of neural network based models for software effort estimation," in *2019 IEEE World Congress on Services (SERVICES)*, IEEE, 2019, pp. 339–342.

[14] S. Goyal and P. K. Bhatia, "A non-linear technique for effective software effort estimation using multi-layer perceptrons," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, IEEE, 2019, pp. 1–4.

[15] K. Srinivasan and D. Fisher, “Machine learning approaches to estimating software development effort,” *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 126–137, 1995.

[16] O. Hidmi and B. E. Sakar, “Software development effort estimation using ensemble machine learning,” *Int. J. Comput. Commun. Instrum. Eng*, vol. 4, no. 1, pp. 143–147, 2017.

[17] A. G. Priya Varshini, K. Anitha Kumari, D. Janani, and S. Soundariya, “Comparative analysis of Machine learning and Deep learning algorithms for Software Effort Estimation,” *J Phys Conf Ser*, vol. 1767, no. 1, p. 012019, 2021, doi: 10.1088/1742-6596/1767/1/012019.

[18] W. Amaral, L. Rivero, G. B. Junior, and D. Viana, “Using Machine Learning Technique for Effort Estimation in Software Development,” in *Proceedings of the XVIII Brazilian Symposium on Software Quality*, in SBQS’19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 240–245. doi: 10.1145/3364641.3364670.

[19] M. Hammad and A. Alqaddoumi, “Features-level software effort estimation using machine learning algorithms,” in *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, IEEE, 2018, pp. 1–3.

[20] M. T. Ribeiro, S. Singh, and C. Guestrin, “‘ Why should i trust you?’ Explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[21] L. Antwarg, R. M. Miller, B. Shapira, and L. Rokach, “Explaining anomalies detected by autoencoders using SHAP,” *arXiv preprint arXiv:1903.02407*, 2019.

[22] A. Trendowicz and R. Jeffery, “Software project effort estimation,” *Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO pags*, vol. 12, pp. 277–293, 2014.

[23] Y.-S. Seo, D.-H. Bae, and R. Jeffery, “AREION: Software effort estimation based on multiple regressions with adaptive recursive data partitioning,” *Inf Softw Technol*, vol. 55, no. 10, pp. 1710–1725, 2013.

[24] R. N. Charette, “Why software fails,” *IEEE Spectr*, vol. 42, no. 9, p. 36, 2005.

[25] The Standish Group, “CHAOS Chronicles. Technical report. The Standish Group International,” *The Standish Group*, 2018.

[26] A. Minkiewicz, “Use case sizing,” in *19th International Forum on COCOMO and Software Cost Modeling, Los Angeles, CA (USA)*, 2004.

[27] H. T. Hoc, V. van Hai, and H. le Thi Kim Nhung, "A review of the regression models applicable to software project effort estimation," in *Proceedings of the Computational Methods in Systems and Software*, Springer, 2019, pp. 399–407.

[28] S. W. Munialo and G. M. Muketha, "A review of agile software effort estimation methods," 2016.

[29] O. Fedotova, L. Teixeira, and H. Alvelos, "Software Effort Estimation with Multiple Linear Regression: Review and Practical Application.," *J. Inf. Sci. Eng.*, vol. 29, no. 5, pp. 925–945, 2013.

[30] H. L. T. K. Nhung, H. T. Hoc, and V. van Hai, "A review of use case-based development effort estimation methods in the system development context," in *Proceedings of the Computational Methods in Systems and Software*, Springer, 2019, pp. 484–499.

[31] D. NESMA, "Counting Guidelines for the Application of Function Point Analysis." Version, 1997.

[32] P. Faria and E. Miranda, "Expert Judgment in Software Estimation During the Bid Phase of a Project--An Exploratory Survey," in *2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement*, IEEE, 2012, pp. 126–131.

[33] S.-W. Lin and V. M. Bier, "A study of expert overconfidence," *Reliab Eng Syst Saf*, vol. 93, no. 5, pp. 711–721, 2008.

[34] M. Shepperd, C. Schofield, and B. Kitchenham, "Effort estimation using analogy," in *Proceedings of IEEE 18th International Conference on Software Engineering*, 1996, pp. 170–178. doi: 10.1109/ICSE.1996.493413.

[35] V. Mahajan, "The Delphi method: Techniques and Applications," *JMR, Journal of Marketing Research (Pre-1986)*, vol. 13, no. 000003, p. 317, 1976.

[36] N. Dalkey and O. Helmer, "An experimental application of the Delphi method to the use of experts," *Manage Sci*, vol. 9, no. 3, pp. 458–467, 1963.

[37] G. Rowe and G. Wright, "The Delphi technique as a forecasting tool: issues and analysis," *Int J Forecast*, vol. 15, no. 4, pp. 353–375, 1999.

[38] B. Barry, "Software engineering economics," *New York*, vol. 197, 1981.

[39] A. Sharma and N. Chaudhary, "Linear regression model for agile software development effort estimation," in *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, IEEE, 2020, pp. 1–4.

[40] V. Van Hai, H. L. T. K. Nhung, and H. T. Hoc, "A Productivity Optimising Model for Improving Software Effort Estimation," in *Software Engineering Perspectives in Intelligent Systems*, R. Silhavy, P. Silhavy, and Z. Prokopova, Eds., Cham: Springer International Publishing, 2020, pp. 735–746.

[41] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, "A comparison between decision trees and decision tree forest models for software development effort estimation," in *2013 Third International Conference on Communications and Information Technology (ICCIT)*, 2013, pp. 220–224. doi: 10.1109/ICCITechnology.2013.6579553.

[42] Z. Prokopova, P. Šilhavý, and R. Šilhavý, "VAF factor influence on the accuracy of the effort estimation provided by modified function points methods," in *Annals of DAAAM and Proceedings of the International DAAAM Symposium*, Danube Adria Association for Automation and Manufacturing, DAAAM, 2018.

[43] D. D. Lewis and M. Ringuette, "A comparison of two learning algorithms for text categorization," in *Third Annual Symposium on document analysis and information retrieval*, 1994, pp. 81–93.

[44] G. H. John, *Enhancements to the data mining process*. Stanford University, 1997.

[45] W. Zhang, Y. Yang, and Q. Wang, "Using Bayesian regression and EM algorithm with missing handling for software effort prediction," *Inf Softw Technol*, vol. 58, pp. 58–70, 2015.

[46] S. Amasaki and T. Yokogawa, "The effects of variable selection methods on linear regression-based effort estimation models," in *2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement*, IEEE, 2013, pp. 98–103.

[47] P. Silhavy, R. Silhavy, and Z. Prokopova, "Categorical variable segmentation model for software development effort estimation," *IEEE Access*, vol. 7, pp. 9618–9626, 2019.

[48] M. A. Ramessur and S. D. Nagowah, "A predictive model to estimate effort in a sprint using machine learning techniques," *International Journal of Information Technology*, vol. 13, no. 3, pp. 1101–1110, 2021, doi: 10.1007/s41870-021-00669-z.

[49] S. Shukla and S. Kumar, "Applicability of neural network based models for software effort estimation," in *2019 IEEE World Congress on Services (SERVICES)*, IEEE, 2019, pp. 339–342.

[50] M. Ochodek, S. Kopczyńska, and M. Staron, "Deep learning model for end-to-end approximation of COSMIC functional size based on use-case names," *Inf Softw Technol*, vol. 123, p. 106310, 2020, doi: https://doi.org/10.1016/j.infsof.2020.106310.

[51] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, "Neural network models for software development effort estimation: a comparative study," *Neural Comput Appl*, vol. 27, no. 8, pp. 2369–2381, 2016.

[52] M. Madheswaran and D. Sivakumar, "Enhancement of prediction accuracy in COCOMO model for software project using neural network," in *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, Ieee, 2014, pp. 1–5.

[53] S. Mukherjee and R. K. Malu, "Optimization of project effort estimate using neural network," in *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, IEEE, 2014, pp. 406–410.

[54] R. Kneuper, *CMMI: improving software and systems development processes using capability maturity model integration*. Rocky Nook, 2008.

[55] D. R. Pai, K. S. McFall, and G. H. Subramanian, "Software effort estimation using a neural network ensemble," *Journal of Computer Information Systems*, vol. 53, no. 4, pp. 49–58, 2013.

[56] C.-L. Liu and Y.-H. Chang, "Learning From Imbalanced Data With Deep Density Hybrid Sampling," *IEEE Trans Syst Man Cybern Syst*, vol. 52, no. 11, pp. 7065–7077, 2022, doi: 10.1109/TSMC.2022.3151394.

[57] A. Islam, S. B. Belhaouari, A. U. Rehman, and H. Bensmail, "KNNOR: An oversampling technique for imbalanced datasets," *Appl Soft Comput*, vol. 115, p. 108288, 2022, doi: https://doi.org/10.1016/j.asoc.2021.108288.

[58] T. Wongvorachan, S. He, and O. Bulut, "A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining," *Information*, vol. 14, no. 1, p. 54, 2023.

[59] C.-L. Liu and Y.-H. Chang, "Learning From Imbalanced Data With Deep Density Hybrid Sampling," *IEEE Trans Syst Man Cybern Syst*, vol. 52, no. 11, pp. 7065–7077, 2022, doi: 10.1109/TSMC.2022.3151394.

[60] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," *GESTS international transactions on computer science and engineering*, vol. 30, no. 1, pp. 25–36, 2006.

[61] H. T. Hoc, V. Van Hai, H. L. T. K. Nhung, and R. Jasek, "Improving the Performance of Effort Estimation in Terms of Function Point Analysis by Balancing Datasets," in *Software Engineering Application in Systems*

Design, R. Silhavy, P. Silhavy, and Z. Prokopova, Eds., Cham: Springer International Publishing, 2023, pp. 705–714.

[62] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[63] M. Zhu *et al.*, "Class Weights Random Forest Algorithm for Processing Class Imbalanced Medical Data," *IEEE Access*, vol. 6, pp. 4641–4652, 2018, doi: 10.1109/ACCESS.2018.2789428.

[64] A. Islam, S. B. Belhaouari, A. U. Rehman, and H. Bensmail, "KNNOR: An oversampling technique for imbalanced datasets," *Appl Soft Comput*, vol. 115, p. 108288, 2022, doi: https://doi.org/10.1016/j.asoc.2021.108288.

[65] Z. Ren *et al.*, "Adaptive cost-sensitive learning: Improving the convergence of intelligent diagnosis models under imbalanced data," *Knowl Based Syst*, vol. 241, p. 108296, 2022, doi: https://doi.org/10.1016/j.knosys.2022.108296.

[66] V. Ganganwar, "An overview of classification algorithms for imbalanced datasets," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 4, pp. 42–47, 2012.

[67] S. Abdellatif, M. A. Ben Hassine, S. Ben Yahia, and A. Bouzeghoub, "ARCID: A New Approach to Deal with Imbalanced Datasets Classification," in *SOFSEM 2018: Theory and Practice of Computer Science*, A. M. Tjoa, L. Bellatreche, S. Biffl, J. van Leeuwen, and J. Wiedermann, Eds., Cham: Springer International Publishing, 2018, pp. 569–580.

[68] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans Pattern Anal Mach Intell*, vol. 12, no. 10, pp. 993–1001, 1990.

[69] S. Shukla and S. Kumar, "Towards ensemble-based use case point prediction," *Software Quality Journal*, 2023, doi: 10.1007/s11219-022-09612-2.

[70] K. K. Beesetti, S. Bilgaiyan, and B. S. P. Mishra, "Software Effort Estimation through Ensembling of Base Models in Machine Learning using a Voting Estimator," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 2, 2023.

[71] S. Goyal, "Effective Software Effort Estimation using Heterogenous Stacked Ensemble," in *2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, 2022, pp. 584–588. doi: 10.1109/SPICES52834.2022.9774231.

[72] P. Suresh Kumar, H. S. Behera, J. Nayak, and B. Naik, "A pragmatic ensemble learning approach for effective software effort estimation," *Innov Syst Softw Eng*, vol. 18, no. 2, pp. 283–299, 2022, doi: 10.1007/s11334-020-00379-y.

[73] M. Hosni, A. Idri, A. B. Nassif, and A. Abran, "Heterogeneous Ensembles for Software Development Effort Estimation," in *2016 3rd International Conference on Soft Computing & Machine Intelligence (ISCMI)*, 2016, pp. 174–178. doi: 10.1109/ISCMI.2016.15.

[74] P. K. M. Passakorn, "Model-Based software effort estimation – A robust comparison of 14 algorithms widely used in the data science community," *International Journal of Innovative Computing, Information and Control*, vol. 15, no. 2, Apr. 2019.

[75] S. K. Palaniswamy and R. Venkatesan, "Hyperparameters tuning of ensemble model for software effort estimation," *J Ambient Intell Humaniz Comput*, vol. 12, no. 6, pp. 6579–6589, 2021, doi: 10.1007/s12652-020-02277-4.

[76] P. V. AG and V. Varadarajan, "Estimating software development efforts using a random forest-based stacked ensemble approach," *Electronics (Basel)*, vol. 10, no. 10, p. 1195, 2021.

[77] E. Kocaguneli, T. Menzies, and E. Mendes, "Transfer learning in effort estimation," *Empir Softw Eng*, vol. 20, no. 3, pp. 813–843, 2015, doi: 10.1007/s10664-014-9300-5.

[78] L. L. Minku and X. Yao, "How to make best use of cross-company data in software effort estimation?," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 446–456.

[79] IFPUG, "http://www.ifpug.org/," *International Function Point Users Group*.

[80] I. F. P. U. Group, "Function Point Counting Practices Manual." Princeton Junction New Jersey, 2010.

[81] J. Hihn, L. Juster, J. Johnson, T. Menzies, and G. Michael, "Improving and expanding NASA software cost estimation methods," in *2016 IEEE Aerospace Conference*, IEEE, 2016, pp. 1–12.

[82] F. Ahmed, S. Bouktif, A. Serhani, and I. Khalil, "Integrating function point project information for improving the accuracy of effort estimation," in *2008 The Second International Conference on Advanced Engineering Computing and Applications in Sciences*, IEEE, 2008, pp. 193–198.

[83] L. Huang, J. Zhang, and Y. Liu, "Antecedents of student MOOC revisit intention: Moderation effect of course difficulty," *Int J Inf Manage*, vol. 37, no. 2, pp. 84–91, 2017.

[84] K. Meridji, K. T. Al-Sarayreh, M. Abu-Arqoub, and W. M. Hadi, "Exploration of development projects of renewable energy applications in the ISBSG dataset: Empirical study," in *2017 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes & Systems (IT-DREPS)*, IEEE, 2017, pp. 1–6.

[85] S. P. Pillai, S. D. Madhukumar, and T. Radharamanan, "Consolidating evidence-based studies in software cost/effort estimation—A tertiary study," in *TENCON 2017-2017 IEEE Region 10 Conference*, IEEE, 2017, pp. 833–838.

[86] M. Fernández-Diego and F. González-Ladrón-de-Guevara, "Application of mutual information-based sequential feature selection to ISBSG mixed data," *Software Quality Journal*, vol. 26, pp. 1299–1325, 2018.

[87] J. Liu, Q. Du, and J. Xu, "A learning-based adjustment model with genetic algorithm of function point estimation," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, IEEE, 2018, pp. 51–58.

[88] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," *Journal of Systems and Software*, vol. 137, pp. 184–196, 2018.

[89] L. Song, L. L. Minku, and X. Yao, "Software effort interval prediction via Bayesian inference and synthetic bootstrap resampling," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 28, no. 1, pp. 1–46, 2019.

[90] Y. Li, L. Shi, J. Hu, Q. Wang, and J. Zhai, "An empirical study to revisit productivity across different programming languages," in *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, 2017, pp. 526–533.

[91] K. Kaewbanjong and S. Intakosum, "Statistical analysis with prediction models of user satisfaction in software project factors," in *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, IEEE, 2020, pp. 637–643.

[92] V. Van Hai, H. L. T. K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, "A new approach to calibrating functional complexity weight in software development effort estimation," *Computers*, vol. 11, no. 2, p. 15, 2022.

[93] F. González-Ladrón-de-Guevara, M. Fernández-Diego, and C. Lokan, "The usage of ISBSG data fields in software effort estimation: A systematic

mapping study," *Journal of Systems and Software*, vol. 113, pp. 188–215, 2016.

[94] Z. Prokopova, P. Silhavy, and R. Silhavy, "Influence analysis of selected factors in the function point work effort estimation," in *Proceedings of the Computational Methods in Systems and Software*, Springer, 2018, pp. 112–124.

[95] V. Van Hai, H. L. T. K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, "Toward Improving the Efficiency of Software Development Effort Estimation via Clustering Analysis," *IEEE Access*, vol. 10, pp. 83249–83264, 2022, doi: 10.1109/ACCESS.2022.3185393.

[96] J. I. S. Martínez, F. V. Souto, and M. R. Monje, "Analysis of automated estimation models using machine learning," in *2020 8th International Conference in Software Engineering Research and Innovation (CONISOFT)*, IEEE, 2020, pp. 110–116.

[97] J. Huang, Y.-F. Li, J. W. Keung, Y. T. Yu, and W. K. Chan, "An empirical analysis of three-stage data-preprocessing for analogy-based software effort estimation on the ISBSG data," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, 2017, pp. 442–449.

[98] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Adv Neural Inf Process Syst*, vol. 30, 2017.

[99] A. and M. A. Najm Assia and Zakrani, "Efficient Shapely Explanation of Support Vector Regression for Agile and Non-agile Software Effort Estimation," in *Intelligent Sustainable Systems*, D. and M. D. K. and J. A. Nagar Atulya K. and Singh Jat, Ed., Singapore: Springer Nature Singapore, 2023, pp. 711–729.

[100] L. A. de Lima, J. M. Abe, C. Z. Kirilo, J. P. da Silva, and K. Nakamatsu, "Using Logic Concepts in Software Measurement," *Procedia Comput Sci*, vol. 131, pp. 600–607, 2018.

[101] G. C. Low and D. R. Jeffery, "Function points in the estimation and evaluation of the software process," *IEEE transactions on Software Engineering*, vol. 16, no. 1, pp. 64–71, 1990.

[102] A. J. Albrecht and J. E. Gaffney, "Software function, source lines of code, and development effort prediction: a software science validation," *IEEE transactions on software engineering*, no. 6, pp. 639–648, 1983.

[103] N. Rankovic, D. Rankovic, M. Ivanovic, and L. Lazic, "A new approach to software effort estimation using different artificial neural network architectures and Taguchi orthogonal arrays," *IEEE Access*, vol. 9, pp. 26926–26936, 2021.

[104] J. Sayyad Shirabad, J. and Menzies, and T.J., "The PROMISE Repository of Software Engineering Databases.," *2005*, 2005.

[105] J. M. Desharnais, "Analyse statistique de la productivitie des projets informatique a partie de la technique des point des function," *Masters Thesis, University of Montreal*, 1989.

[106] B. Kitchenham, S. L. Pfleeger, B. McColl, and S. Eagan, "An empirical study of maintenance and development estimation accuracy," *Journal of systems and software*, vol. 64, no. 1, pp. 57–77, 2002.

[107] F. H. Yun, "China: Effort Estimation Dataset," Apr. 2010, doi: 10.5281/ZENODO.268446.

[108] H. He, B. Yang, E. A. Garcia, and S. A. Li, "Adaptive synthetic sampling approach for imbalanced learning. Proeedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence); June 2008; Hong Kong, China." ChinaIEEE.

[109] L. Taylor and G. Nitschke, "Improving Deep Learning with Generic Data Augmentation," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018, pp. 1542–1547. doi: 10.1109/SSCI.2018.8628742.

[110] A. Arora, N. Shoeibi, V. Sati, A. González-Briones, P. Chamoso, and E. Corchado, "Data Augmentation Using Gaussian Mixture Model on CSV Files," in *Distributed Computing and Artificial Intelligence, 17th International Conference*, Y. Dong, E. Herrera-Viedma, K. Matsui, S. Omatsu, A. González Briones, and S. Rodríguez González, Eds., Cham: Springer International Publishing, 2021, pp. 258–265.

[111] D. A. Reynolds, "Gaussian mixture models.," *Encyclopedia of biometrics*, vol. 741, no. 659–663, 2009.

[112] A. S. Hadi and S. Chatterjee, *Regression analysis by example*. John Wiley & Sons, 2015.

[113] K. Baker, "Singular value decomposition tutorial," *The Ohio State University*, vol. 24, 2005.

[114] L. Breiman, "Arcing the edge," Technical Report 486, Statistics Department, University of California at …, 1997.

[115] "http://www.stat.berkeley.edu/~breiman/RandomForests/cc_h ome.htm," *#prox Symposium, volume 1*, Jul. 01, 2005.

[116] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random forests and decision trees," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 272, 2012.

[117] Z. abdelali, H. Mustapha, and N. Abdelwahed, "Investigating the use of random forest in software effort estimation," *Procedia Comput Sci*, vol. 148, pp. 343–352, 2019, doi: https://doi.org/10.1016/j.procs.2019.01.042.

[118] A. Liaw and M. Wiener, "Classification and Regression by RandomForest," *Forest*, vol. 23, Nov. 2001.

[119] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J Comput Syst Sci*, vol. 55, no. 1, pp. 119–139, 1997.

[120] J. Friedman, "Greedy boosting approximation: a gradient boosting machine," *The Annals of*.

[121] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[122] H. Aljamaan and A. Alazba, "Software defect prediction using tree-based ensembles," in *Proceedings of the 16th ACM international conference on predictive models and data analytics in software engineering*, 2020, pp. 1–10.

[123] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[124] Aurélien Géron, "Ensemble Learning and Random Forests," in *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed.O'reilly, 2019, pp. 189–212.

[125] J. Brownlee, *XGBoost With python: Gradient boosted trees with XGBoost and scikit-learn*. Machine Learning Mastery, 2016.

[126] A. Guryanov, "Histogram-based algorithm for building gradient boosting ensembles of piecewise linear decision trees," in *Analysis of Images, Social Networks and Texts: 8th International Conference, AIST 2019, Kazan, Russia, July 17–19, 2019, Revised Selected Papers 8*, Springer, 2019, pp. 39–50.

[127] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[128] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[129] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer (Long Beach Calif)*, vol. 29, no. 3, pp. 31–44, 1996.

[130] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

[131] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[132] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans Knowl Data Eng*, vol. 22, no. 10, pp. 1345–1359, 2010.

[133] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, IGI Global, 2010, pp. 242–264.

[134] A. Arnold, R. Nallapati, and W. W. Cohen, "A comparative study of methods for transductive transfer learning," in *Seventh IEEE international conference on data mining workshops (ICDMW 2007)*, IEEE, 2007, pp. 77–82.

[135] A. Arnold, R. Nallapati, and W. W. Cohen, "A Comparative Study of Methods for Transductive Transfer Learning," in *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, 2007, pp. 77–82. doi: 10.1109/ICDMW.2007.109.

[136] P. Baldi, "Autoencoders, Unsupervised Learning, and Deep Architectures," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, Eds., in Proceedings of Machine Learning Research, vol. 27. Bellevue, Washington, USA: PMLR, May 2012, pp. 37–49. [Online]. Available: https://proceedings.mlr.press/v27/baldi12a.html

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

[138] I. Myrtveit, E. Stensrud, and M. Shepperd, "Reliability and validity in comparative studies of software prediction models," *IEEE Transactions on Software Engineering*, vol. 31, no. 5, pp. 380–391, 2005.

[139] Y. Mahmood, N. Kama, A. Azmi, A. S. Khan, and M. Ali, "Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation," *Softw Pract Exp*, vol. 52, no. 1, pp. 39–65, 2022.

[140] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Inf Softw Technol*, vol. 54, no. 8, pp. 820–827, 2012.

[141] T. Xia, R. Shu, X. Shen, and T. Menzies, "Sequential model optimization for software effort estimation," *IEEE Transactions on Software Engineering*, 2020.

[142] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation.," *Encyclopedia of database systems*, vol. 5, pp. 532–538, 2009.

[143] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Ijcai*, Montreal, Canada, 1995, pp. 1137–1145.

[144] X. Ma, Y. Zhang, and Y. Wang, “Performance evaluation of kernel functions based on grid search for support vector regression,” in *2015 IEEE 7th international conference on Cybernetics and intelligent systems (CIS) and IEEE Conference on Robotics, automation and mechatronics (RAM)*, IEEE, 2015, pp. 283–288.

[145] J. Brownlee, “A gentle introduction to the rectified linear unit (ReLU),” *Machine learning mastery*, vol. 6, 2019.

[146] E. Okewu, S. Misra, and F.-S. Lius, “Parameter tuning using adaptive moment estimation in deep learning neural networks,” in *International Conference on Computational Science and Its Applications*, Springer, 2020, pp. 261–272.

[147] V. N. Gudivada, M. T. Irfan, E. Fathi, and D. L. Rao, “Chapter 5 - Cognitive Analytics: Going Beyond Big Data Analytics and Machine Learning,” in *Handbook of Statistics*, V. N. Gudivada, V. V Raghavan, V. Govindaraju, and C. R. Rao, Eds., Elsevier, 2016, pp. 169–205. doi: https://doi.org/10.1016/bs.host.2016.07.010.

[148] R. Silhavy, P. Silhavy, and Z. Prokopova, “Analysis and selection of a regression model for the use case points method using a stepwise approach,” *Journal of Systems and Software*, vol. 125, pp. 1–14, 2017.

[149] M. Azzeh, A. Bou Nassif, and I. B. Attili, “Predicting software effort from use case points: A systematic review,” *Sci Comput Program*, vol. 204, p. 102596, 2021, doi: https://doi.org/10.1016/j.scico.2020.102596.

[150] A. De Myttenaere, B. Golden, B. Le Grand, and F. Rossi, “Mean absolute percentage error for regression models,” *Neurocomputing*, vol. 192, pp. 38–48, 2016.

LIST OF PUBLICATIONS

Journals:

1. **Hoc, H. T.**, R. Silhavy, Z. Prokopova and P. Silhavy, “Comparing Multiple Linear Regression, Deep Learning and Multiple Perceptron for Functional Points Estimation,” in *IEEE Access*, vol. 10, pp. 112187-112198, 2022, doi: 10.1109/ACCESS.2022.3215987.
2. **Hoc, H. T.**, R. Silhavy, Z. Prokopova and P. Silhavy, "Comparing Stacking Ensemble and Deep Learning for Software Project Effort Estimation," in IEEE

Access, vol. 11, pp. 60590-60604, 2023, doi: 10.1109/ACCESS.2023.3286372.

3. **Hoc, H. T.**, Silhavy P., Fajkus M., Prokopova Z, Silhavy R. Propose-Specific Information Related to Prediction Level at x and Mean Magnitude of Relative Error: A Case Study of Software Effort Estimation. *Mathematics*. 2022; 10(24):4649. https://doi.org/10.3390/math10244649.
4. **Hoc, H. T.**, Silhavy P., Dey SK, Hoang SD, Prokopova Z, Silhavy R. Analysing Public Opinions Regarding Virtual Tourism in the Context of COVID-19: Unidirectional vs. 360-Degree Videos. *Information*. 2023; 14(1):11. https://doi.org/10.3390/info14010011.
5. DEY, Sandeep Kumar, Duc Sinh HOANG, **Hoc, H. T.**, Quynh Giao Ngoc PHAM. Engaging virtual reality technology to determine pro-environmental behaviour: The Indian context. Geojournal of Tourism and Geosites [online]. 2022, vol. 41, iss. 2, s. 464-471. [cit. 2023-03-14]. ISSN 2065-0817.
6. Kondamudi, B. R., Hoang, S. D., Tuckova, Z., Dey, S. K., **Hoc, H. T.**, & Kumar, B. R. (2023). Tourists' Perception and Influence Factors in Virtual Tourism Using Bayesian Sentimental Analysis Model in Vietnam Based on e WOM for Sustainable Development. Journal of Law and Sustainable Development, 11(3), e338. https://doi.org/10.55908/sdgs.v11i3.338.
7. Pham P.T., **Hoc, H. T.**, B.Popesko, Sinh D.H., Tri B.T, "Impact of Fintech's Development on Bank Performance: An Empirical Study from Vietnam.", accept submission by GamaIJB, Volume 26 No.1, 2023.

Conferences:

8. **Hoc, H. T**., Van Hai, V., Nhung, H. L. T. K., & Jasek, R. (2023). Improving the Performance of Effort Estimation in Terms of Function Point Analysis by Balancing Datasets. In *Software Engineering Application in Systems Design: Proceedings of 6th Computational Methods in Systems and Software 2022, Volume 1* (pp. 705-714). Cham: Springer International Publishing.
9. **Hoc, H. T**., Van Hai, V., & Le Thi Kim Nhung, H. (2020). AdamOptimizer for the optimisation of use case points estimation. In *Software Engineering Perspectives in Intelligent Systems: Proceedings of 4th Computational Methods in Systems and Software 2020, Vol. 1 4* (pp. 747-756). Springer International Publishing.
10. **Hoc, H. T**., Van Hai, V., & Nhung, H. L. T. K. (2021). An approach to adjust effort estimation of function point analysis. In *Software Engineering and Algorithms: Proceedings of 10th Computer Science On-line Conference 2021, Vol. 1* (pp. 522-537). Springer International Publishing
11. **Hoc, H. T.**, Van Hai, V., & Le Thi Kim Nhung, H. (2019). A review of the regression models applicable to software project effort estimation. *Computational Statistics and Mathematical Modeling Methods in Intelligent Systems: Proceedings of 3rd Computational Methods in Systems and Software 2019, Vol. 2 3*, 399-407.

12. Van Hai, V., Le Thi Kim Nhung, H., & **Hoc, H. T.** (2021). Empirical Evidence in Early Stage Software Effort Estimation Using Data Flow Diagram. In *Software Engineering and Algorithms: Proceedings of 10th Computer Science On-line Conference 2021, Vol. 1* (pp. 632-644). Springer International Publishing.

13. Nhung, H. L. T. K., Van Hai, V., **Hoc, H. T.** Analyzing Correlation of the Relationship between Technical Complexity Factors and Environmental Complexity Factors for Software Development Effort Estimation.

14. Hai, V. V., Nhung, H. L. T. K., & **Hoc, H. T.** (2021). Calibrating Function Complexity in Enhancement Project for Improving Function Points Analysis Estimation. In *Software Engineering Application in Informatics: Proceedings of 5th Computational Methods in Systems and Software 2021, Vol. 1* (pp. 857-869). Springer International Publishing.

15. Le Thi Kim Nhung, H., **Hoc, H. T.**, & Van Hai, V. (2020). An evaluation of technical and environmental complexity factors for improving use case points estimation. In *Software Engineering Perspectives in Intelligent Systems: Proceedings of 4th Computational Methods in Systems and Software 2020, Vol. 1 4* (pp. 757-768). Springer International Publishing.

16. Hai, V. V., Nhung, H. L. T. K., & **Hoc, H. T.** (2020). A Productivity optimising model for improving software effort estimation. In *Software Engineering Perspectives in Intelligent Systems: Proceedings of 4th Computational Methods in Systems and Software 2020, Vol. 1 4* (pp. 735-746). Springer International Publishing.

17. Van Hai, V., Le Thi Kim Nhung, H., & **Hoc, H. T.** (2019). A review of software effort estimation by using functional points analysis. *Computational Statistics and Mathematical Modeling Methods in Intelligent Systems: Proceedings of 3rd Computational Methods in Systems and Software 2019, Vol. 2 3*, 408-422.

18. Nhung, H. L. T. K., **Hoc, H. T.**, & Hai, V. V. (2019). A review of use case-based development effort estimation methods in the system development context. *Intelligent Systems Applications in* Software Engineering: Proceedings of 3rd Computational Methods in Systems and Software 2019, Vol. 1 3, 484-499.

19. Dey, S.K., Hoang, D.S, **Hoc, H.T.**, & Pham, Q.G.N. (2022). ENGAGING VIRTUAL REALITY TECHNOLOGY TODETERMINE PRO-ENVIRONMENTAL BEHAVIOUR: THE INDIAN CONTEXT X. GeoJournal of Tourism and Geosites,41(2), 464–471. https://doi.org/10.30892/gtg.41217-851.

20. Dey, S.K., Hung, V.V., **Hoc, H.T.**, Pham, Q.G.N. (2022). AVR Technologies in Sustainable Tourism: A Bibliometric Review. In: Bashir, A.K., Fortino, G., Khanna, A., Gupta, D. (eds) Proceedings of International Conference on Computing and Communication Networks. Lecture Notes in Networks and

Systems, vol 394. Springer, Singapore. https://doi.org/10.1007/978-981-19-0604-6_52.

21. Nguyen T.T.N., Cartocci A., **Hoc H. T.**, Tong L. T., Mozafari M., Dang T.K., Nguyen T.Z., AI-aided automatic severity scoring system for Hidradenitis Suppurativa, 12th Hybrid Conference of the EHSF 2023 Hidradenitis Suppurativa / Acne Inversa-Tagung 2023.

Book Editor:

22. Zuzana Tučková, Sandeep Kumar Dey, **Hoc H. T.**, Sinh Duc Hoang, Impact of Industry 4.0 on Sustainable Tourism: Perspectives, Challenges and Future, published by Emerald, October, 2023.

CURRICULUM VITAE

Personal Information

Full name: Huynh Thai Hoc

Address: 30/2C Trung My Tan Xuan, Hoc Mon, Ho Chi Minh City, Vietnam

Nationality: Vietnamese

Orcid ID: 0000-0003-3845-8466

Scholar ID: xoesuc8AAAAJ

Email: huynh_thai@utb.cz; hoc.ht@vlu.edu.vn; huynhhoc@gmail.com

Work Experiences

- July 2023 – September 2023: Lead researcher for Internal Geospatial Data Science Bootcamp at Valhko company, France.
- March 2022 – January 2023: Internship at Torus Actions, Toulouse, France.
- 2018 – ongoing: Lecturer at the Faculty of Information Technology, School of Engineering and Technology, Van Lang University, HCMC, Vietnam.
- 2011 – 2018: Lecture at Faculty of Information Technology, University of Industry (UIH), Ho Chi Minh City, Vietnam.
- 2014 – January 2019: Developer at Capgemini Vietnam, HCMC, Vietnam.
- 2007 – 2014: Lecture at Faculty of Information Technology, University of Natural Resources and Environment (HCMUNRE), HCMC, Vietnam.
- 2002 – 2007: GIS developer at DITAGIS, HCMC, Vietnam.

Education

- 2019 – 10/2023: PhD scholar at the Faculty of Applied Informatics, Tomas Bata University, Zlin, the Czech Republic.
- 2004 – 2007: master's degree in Geographic Information Systems, University of Technology (HCMUT), Ho Chi Minh City, Vietnam.
- 1998 – 2002: bachelor's degree in mathematics and computer Science, University of Science (HCMUS), Ho Chi Minh City, Vietnam.

Programming Languages

- R programming

- Python
- C/C++; Core Java; .NET

Data scientist skills

- Pytorch, Tensorflow
- Random Forest, XGBoost, SVM
- Regression models, Ensemble
- LIME/SHAP
- Generative Models
- Scikit learns

Technical skills

- SQL Server, PostgreSQL/PostGIS, MongoDB
- .NET
- Design patterns (MVC)
- Web service (Restful, SOAP)
- HTML, CSS, JavaScript
- Git and Agile methodologies

Research Interests

Data Scientist, GIS, Developer.

Research Activities at Tomas Bata University, Zlin

- IGA projects:
    - IGA/CebiaTech/2019/002
    - RO30196021025, RO30196021025
    - IGA/CebiaTech/2020/001, RVO/FAI/2020/002
    - IGA/CebiaTech/2021/001
    - IGA/CebiaTech/2022/001
- Competition projects:
    - OP RDE Project Junior grants of TBU in Zlin, reg. No. CZ.02.2.69/0.0/19_073/0016941.
    - IGA-K-TRINITY/004.