Research Software Cost Estimation with Neural Networks and exploration of unused methodologies.

Realization

Bachelor Applied Computer Science

Robin Ramaekers

Acedemic year 2022-2023

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

LID VAN ASSOCIATIE KU LEUVEN

THOMAS MORE

TABLE OF CONTENTS

TABLE OF CONTENTS **3**

ABSTRACT **5**

1 ACKNOWLEDGEMENTS **6**

2 TERMINOLOGY **7**

3 INTRODUCTION **9**

4 PRESENTING TOMAS BATA UNIVERSITY **10**

5 PROJECT SCOPE **12**

5.1 Description **12**
5.2 Deliverables **12**
1. Project scope 12

6 REVIEW PAPER: SOFTWARE COST ESTIMATION AND NEURAL NETWORKS **14**

Abstract **14**
6.1 Introduction **14**
6.2 Related works **15**
6.3 Software Cost Estimation (SCE) **15**
6.3.1 Taxonomies 16
6.3.2 Challenges of existing methods 17
6.4 Data sources **18**
6.4.1 Most common 18
6.4.2 EDA Data sources 18
6.5 Neural Network implementation **22**
6.5.1 Evolution 22
6.5.2 Advantages 23
6.6 Evaluation metrics **23**
6.6.1 MRE (Magnitude of relative error) 23
6.6.2 MMRE (Mean Magnitude of Relative Error) 23
6.6.3 MSE (Mean Squared Error) 23
6.6.4 RMSE (Root Mean Squared Error) 24
6.6.5 MdMRE (Median Magnitude of Relative Error) 24
6.6.6 PRED (Prediction accuracy) 24
6.6.7 MAE (Mean Absolute Error) 24
6.7 Methodologies **24**
6.7.1 Genetic elephant herding optimization (GEHO) based Neuro-fuzzy network (GEHO-based NFN). 25
6.7.2 Optimized Learning-based Cost Estimation (OLCE) 25
6.7.3 ANN with NEAT optimization (Artificial Neural Network with Neuro-Evolution of Augmenting Topology). 26
6.8 Gap in knowledge **27**
6.8.1 Future proposal 28

6.9 Conclusion 28

7 RESEARCH PAPER: SOFTWARE COST ESTIMATION USING SYNTHETIC DATA AND ENSEMBLE MODELING 29

Abstract 29
7.1 Introduction 29
7.2 Related works 30
7.3 Software Cost Estimation 31
7.3.1 Existing methods and key challenges 32
7.4 Methods used 34
7.4.1 Neural network 34
7.5 Methodology 35
7.5.1 Experimental datasets 35
7.5.2 Validation method 38
7.5.3 Handling a small dataset issue 38
7.5.4 Ensemble method 39
7.5.5 Tested model evaluation (work in progress) 39
7.6 Evaluation metrics 40
7.6.1 PRED25 (Prediction accuracy) 40
7.6.2 MSE (Mean Squared Error) 40
7.6.3 MAE (Mean Absolute Error) 41
7.7 Used algorithms formulations 41
7.7.1 Linear regression (LR) 41
7.7.2 K-Nearest Neighbors (KNN) 41
7.7.3 Support Vector Machines (SVM) 42
7.7.4 Least Absolute Shrinkage and Selection Operator (Lasso) 42
7.8 Results 42
7.9 Gap in knowledge and possible solutions 43
7.9.1 Future proposal 43
7.10 Conclusion 44

8 CONCLUSION 45

9 LITERATURE CITED 46

ABSTRACT

This work is written based on the internship of Robin Ramaekers and was made to graduate as a bachelor in the field of study, Applied Computer Science at Thomas More hoge school Geel, Belgium. The subject is conducting a research on the subject Software Cost Estimation (SCE) in correlation with Neural Networks.

The first step in this research is to define the scope of the different parts that should be fitted in the three months work period. This was done by giving a high-level description of all the different tasks and deliverables with the help of a project plan.

Once the project plan was finished, a research was initiated where existing methodologies regarding Software Cost Estimation and Neural Networks will be reviewed. The things found in this research paper will be compiled into a review paper. The ultimate goal of this paper is to find a gap in knowledge which I will try to solve later on.

After the review paper was completed, a new research was initiated to solve the gaps in knowledge and unused methodologies of the review paper. This part mainly consisted of doing a lot of experimentation to achieve a decent accuracy score. All the results are combined into a research paper that tries to convince other researchers to adapt and fine-tune the methodologies.

1 ACKNOWLEDGEMENTS

I would like to thank the FAI of the Tomas Bata University for the opportunity to do my internship at this innovative university and to experience different cultures and live abroad. This internship was the perfect way to finish my three-year bachelor study.

I want to specifically thank Radek Šilhavý, my supervisor throughout this adventure, for guiding me to the end of a successful internship. Your input and expertise in the Software Cost Estimation field made my first steps into this complicated research domain more approachable.

I would also like to give a big thanks to the teachers of Thomas More, and Michiel Verboven for being my internship coach. With the help of your feedback, I always knew if I was on right path.

In my three years of studying, it wasn't always smooth sailing and there where some difficult times. That's why I would like to thank my parents for believing in me.

At last, I would also like to thank my friends that I made over these last three years. It was always fun to make group projects and getting to know like-minded people.

2 TERMINOLOGY

General

- SCE: Software Cost Estimation
- AI: Artificial Intelligence
- ML: Machine Learning
- DL: Deep learning
- CNN: Convelutional Neural Network
- NLP: Natural Language Processing
- EDA: Exploratory Data Analysis
- 80-20 rule: heuristic used by researchers for splitting the data, 80 percent will become training data while 20 percent will become testing data.
- FIA: Faculty of Applied Informatics
- UTB/TBU: University Tomas Bata/Tomas Bata University
- ESN: Erasmus Student Network

Artificial Intelligence algorithms

- RL: Reinforcement Learning
- DRL: Deep Reinforcement Learning
- LR: Linear Regression
- KNN: K-Nearest Neigbhours
- SVM: Support Vector Machines
- LASSO: Least Absolute Shrinkage and Selection Operator
- GEHO: Genetic Elephant Herding Optimization
- EHO: Elephant Herding Optimization
- ANN: Artificial Neural Network
- NEAT: Neuro-Evolution of Augmenting Topology
- OLCE: Optimized Learning-based Cost Estimation
- FNN: Feedforward Neural Network
- NFN: Neural Fuzzy Network
- SEERSEM: Software Evaluation and Estimation of Resources
- AA: Aiming Algorithm

Data Source names

- COCOMO: Constructive Cost Model
- ISBSG: International Software Benchmarking Standards Group
- UCF: the University of Central Florida

Data Source types:

- UCP: Use-Case Points
- FPA: Functional Point Analysis
- LOC: Lines Of Code
- KSLOC: Thousands of Source Lines of Code

Data Source used features:

Funtional Point Analysis:

- EI: External Inputs
- EO: External Outputs
- EQ: External Inquiries
- ILF: Internal Logical Files

- EIF: External Interface Files

Use-Case points analysis:

- UAW: Unadjusted Actor Weight
- UUCW: Unadjusted Use Case Weight
- TCF: Technical Complexity Factor
- ECF: Environmental Complexity Factor

Evaluation Metrics:

- MRE: Magnitude of relative error
- MMRE: Mean Magnitude of relative error
- MdMRE: Median Magnitude of relative error
- MSE: Mean Squared Error
- RMSE: Root Mean Squared Error
- PRED: returns the prediction accuracy in a percentage.
- PRED25: returns the prediction accuracy in a percentage. This Formula takes in that the predicted value is in the 25 percent range of the actual values.
- MAE: Mean Abslolute Error

3 INTRODUCTION

Software Cost Estimation (SCE) is a critical aspect of any new software development project. Its primary objective is to forecast the cost of the development of software projects before the project phases are initiated. This includes determining the resources required for development, the time needed for development, and the estimated effort required to achieve the end goal of the development phase. The success or failure of a project is closely linked to the accuracy of the Software Cost Estimation process.

Unfortunately, Software Cost Estimation is a challenging task due to several factors such as lack of expertise, lack of historical data, human bias, and overconfidence. To address these issues, Artificial Intelligence (AI) models have been developed over the past decade to enhance the accuracy of Software Cost Estimation.

AI is rapidly advancing and has shown great promise in automating and making predictions based on presented data. This progress is attributable to more powerful hardware and publicly available data. While the COCOMO model is the most commonly adopted model in the software industry because of its simplicity, flexibility, historical data, and wide applicability, it also has its strengths and weaknesses.

Consequently, researchers have been experimenting with various methods to improve the accuracy of Software Cost Estimation.

With the availability of more data sources regarding SCE, many researchers have become interested in the topic, paving the way for further exploration and advancement in this field. Many Machine Learning techniques have been applied to do SCE e.g., Lasso, K-Nearest Neighbors, Linear Regression (LR), and Support Vector Machines (SVM), these algorithms can achieve decent accuracy and minimal loss. In this paper, we will explore the possibilities of newer Machine and Deep Learning techniques.

Our proposed research explores the possibilities of synthetic data and ensemble models. We will use synthetic data to eliminate the lack of historical data and improve the overall consistency of the data. This data will be used to train and evaluate the performance of our models.

The best-performing models will be used in an ensemble model using the stacking technique, which will make a final prediction using the strengths of each previously mentioned models. This technique can solve complex problems with a lot of variables and an adaptive environment like that of software engineering.

The application of Artificial Intelligence in Software Cost Estimation is gaining importance, as it has the potential to improve accuracy, reduce costs, and enhance decision-making processes in software development projects, thereby underscoring the need for a broader adoption and standardisation regarding the use of ensemble models trained on synthetic data.

4 PRESENTING TOMAS BATA UNIVERSITY

Tomas Bata University (TBU/UTB) is a modern and dynamic public university located in the Moravian city of Zlín in the Czech Republic. The university consist of six faculties that each have their own dedicated building around the city center, UTB offers courses in technology, economics, humanities, arts, and healthcare. I spend most of my time in the U5 building which is dedicated to the Faculty of Applied Computer Science (FIA) and technology. The FIA building is equipped with a canteen, fitness, shared offices with other students and classrooms.

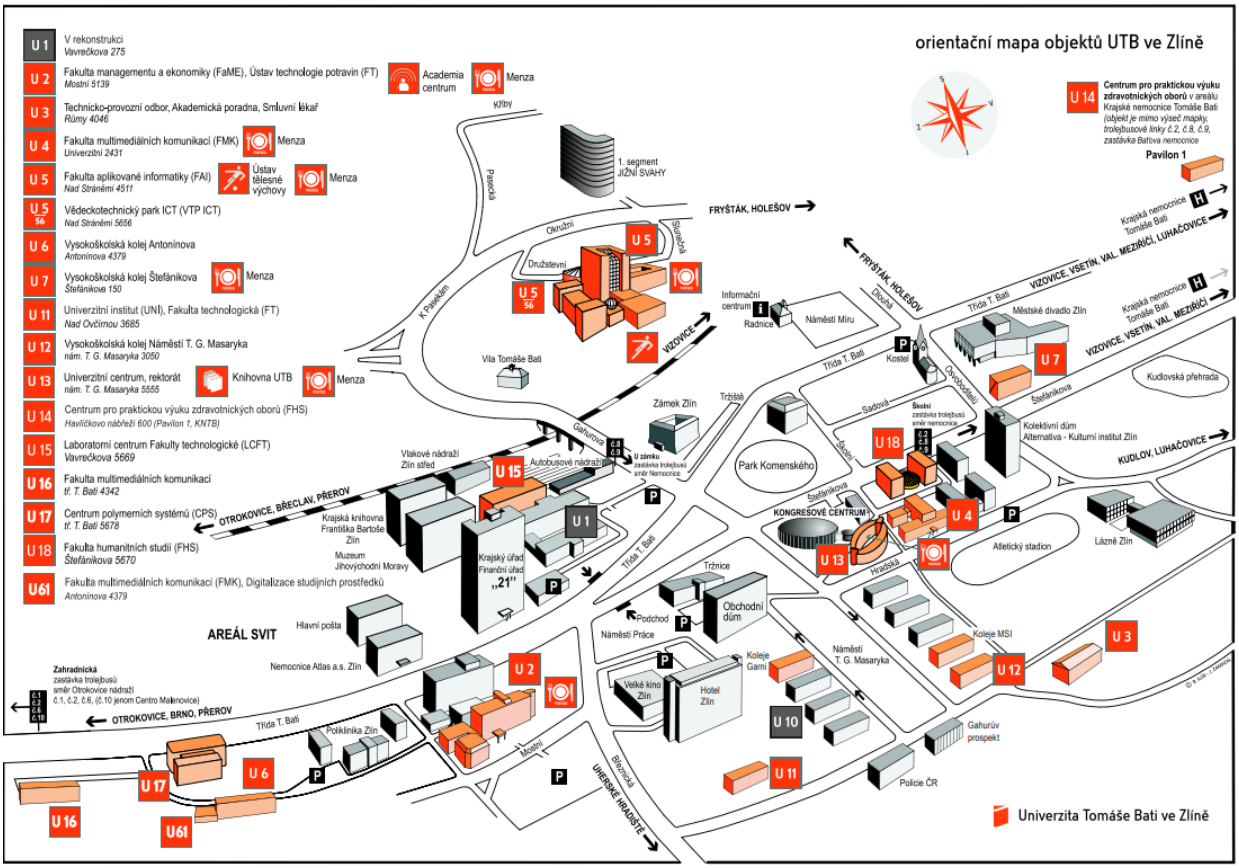


Figure 1: Map with all of the buildings owned by UTB.

Figure 2: Entrance of the U5 (FAI) building.

The school was founded in 2001 and named after the entrepreneur and philanthropist Tomas Bata. He was responsible for the shoe industry in Zlín, Thomas Bata was had a huge impact on Zlín so you can see memorials all over the city. His shoemaking skills where without a doubt ahead of their time.

One of the key strengths of UTB is its innovative approach to education. The university emphasizes project-based learning, giving students the opportunity to get hands on experience and develop practical skills that are valued in the job market.

The university also has a diverse student body, with over 9500 students from more than 70 countries. [1] The cultural differences also gives students the chance to learn more about different countries and this is also emphasized by the ESN network who regularly organize events like parties, trips to other close by countries, country presentations and an international festival.

You also get the opportunity to learn the Czech language for free, this way it is easier to get around and communicate in your daily life. I took this course on Monday and Wednesday and I don’t regret it.

UTB also provides accommodation for its students I stayed at the U12 dormitory; this is the main dormitory for international students. The room was equipped with a kitchen and bathroom shared over four students and the bedroom with two students. The dormitory also had useful facilities like washing machines and cleaning equipment.

5 PROJECT SCOPE

5.1 Description

The internship will start on the 1st of march and end on the 25th of May. This time period will consist of three parts.

The first part will define the scope of the project, this will be done with the help of a project plan. This document will be presented during the second meeting on the planned date with my internship supervisor.

In the second part, a research regarding Software Cost Estimation (SCE) and Neural Networks will be instantiated. The primary goal of this research is to find gap in the market that I will be trying to solve by conducting experiments. This will also be useful to get familiar with the topic and see which methodologies are already being researched. All this information will be incorporated into a review paper.

In the third part, a research paper will be made based on the findings that were discovered in the review paper. The core goal of a research paper is to formulate all the finding and experimentation that has been done to solve the gaps of knowledge that were previously discovered.

5.2 Deliverables

1. Project scope

   Consists of:

   - Project title
   - Project justification
   - Project scope
   - List of deliverables

2. Review Paper: Software Cost Estimation and Neural Networks

   Consists of:
   - Exploration topic Software Cost Estimation and standard methodologies
   - Available data sources and their properties
   - The neural network implementation and its evolution
   - Evaluation metrics
   - Explanation of reviewed methodologies
   - Gaps in knowledge

3. Research paper: Software Cost Estimation using synthetic data and ensemble modelling.

   Consists of:
   - Explanation of types of data sets to do SCE
   - Explanation used types and datasets.

- Evaluation metrics
- Used algorithms formulations
- Explanation used methodologies
- Justification used methodologies
- Gaps in knowledge.

6 REVIEW PAPER: SOFTWARE COST ESTIMATION AND NEURAL NETWORKS

Abstract

Software Cost Estimation (SCE) is one of the most vital parts when starting a new software engineering project, it helps with allocating resources, managing risks, making informed decisions, and stands in correlation with the success or the failure of a project. Because Software Cost Estimation (SCE) is prone to human bias, solutions started being researched with the aid of Artificial Intelligence (AI) and Machine Learning (ML). This paper will investigate the importance of Software Cost Estimation (SCE). Further, the existing taxonomies and methodologies regarding the use of neural networks with Software Cost estimation will be compared (COCOMO, GEHO-ANN, OLCE, and ANN-NEAT). This will be done with the use of evaluation metrics such as RMSE, MMRE, PRED, MAE, etc. After, a proposal for further research is made on why using Deep Reinforcement Learning (DRL) could be very beneficial for the development of Software Cost Prediction Models. This technique combines Deep Learning (DL) and Machine Learning (ML) and can solve complex tasks with many variables and a rapidly developing environment.

Keywords: Software Cost Estimation, Artificial Intelligence, Machine Learning, Deep Reinforcement Learning, Neural Networks

6.1 Introduction

The importance of Software Cost Estimation (SCE) cannot be understated in a new software development project. The core goal of Software Cost Estimation (SCE) is to perform a prognostic analysis of the cost of the development of software projects before the project phases are initiated. The term cost refers to the number of resources that will be engaged in the development, time consumed in development, and all approximated effort to achieve the end-state of the development phase. The process of Software Cost Estimation (SCE) stands in correlation with the success or the failure of a project [1]. The core challenge of Software Cost Estimation is lack of expertise, lack of historical data, overconfidence, human bias, etc. This is the reason why Software Cost Estimation (SCE) Models using Artificial intelligence (AI) have been developing over the last decade. Artificial Intelligence (AI) has been rapidly growing and shows great promise in automatization and making predictions based on the data that is presented. This rapid evolution is thanks to more powerful hardware and data that has become publicly available. The most frequently adopted model in the software industry is COCOMO because of its simplicity, flexibility, historical data, and wide applicability but this model has its strengths and weaknesses, this is the reason that researchers have been experimenting with diverse methods. The COCOMO model is based on the COCOMO dataset. Many more data sources regarding SCE are now publicly available, this caused many researchers to take an interest in the topic. Many machine learning techniques have been applied to do SCE e.g., GEHO-based NFN, OLSE, ANN-NEAT, these algorithms can achieve decent accuracy and minimal loss. In this paper, the above-mentioned algorithms are summarized and compared to each other. The research on these techniques where done in the last year (2022-2023) and where found on Google Scholar. Finally, it was noted that while there is a lot of information about the use of machine learning regarding Software Cost Estimation, Deep Learning a more capable form of Machine Learning was rarely mentioned or researched that is why I propose a Deep Reinforcement Learning

approach, the reasoning being that it combines Machine Learning and Deep Learning techniques to solve complex problems with a lot of variables and an adaptive environment similar to that of software engineering. This paper is divided into 10 sections: 1 Introduction, 2 Related works, 3 Software Cost Estimation, 4 Data sources, 5 Neural Network implementation, 6 Evaluation metrics, 7 Methodologies, 8 Future proposal, 9 Conclusion, and 10 Literature cited. The use of Artificial Intelligence in Software Cost Estimation is becoming increasingly important, as it has the potential to improve accuracy, reduce costs, and enhance decision-making processes in software development projects, highlighting the need for greater adoption of Deep Learning approaches in the field.

6.2 Related works

Several studies have investigated the use of Neural Networks in Software Cost Estimation focusing on Machine Learning.

For example, Sudhir Sharma & Shripal Vijayvargiya (2022) have developed several Neural Network models using Machine Learning to do Software Cost Estimation, they benchmarked all the models on different data sources to see which model and technique performed the best.

Meanwhile, K Nitalaksheswara Rao, Jhansi Vazram Bolla, Satyanarayana Mummana, CH. V. Murali Krishna, O. Gandhi & M James Stephen (2022) created an optimized learning-based Cost Estimation model and scheme.

This approach achieved a balance between accurate predictions and efficiency for computational power.

Other studies have explored the use of Artificial Neural Networks (ANN), with Puppala Ramya, M. Sai Mokshith, M. Abdul Rahman & N. Nithin Sai (2023) developing a model that can comprehend complex relationships and patterns found in the data sources for Software Cost Estimation.

6.3 Software Cost Estimation (SCE)

As mentioned in section 1, the importance of Software Cost Estimation (SCE) cannot be understated in a new software development project. The core goal of Software Cost Estimation (SCE) is to perform a prognostic analysis of the cost of the development of software projects before the project phases are initiated. The term cost refers to the number of resources that will be engaged in the development, time consumed in development, and all approximated effort to achieve the end-state of the development phase. In the figure, Figure 1, a typical cost estimation practice is shown. In this case, the project manager is responsible for the target software cost. The process below cycles and the various attributes are modified until the cost of the target software is found justified. [4]

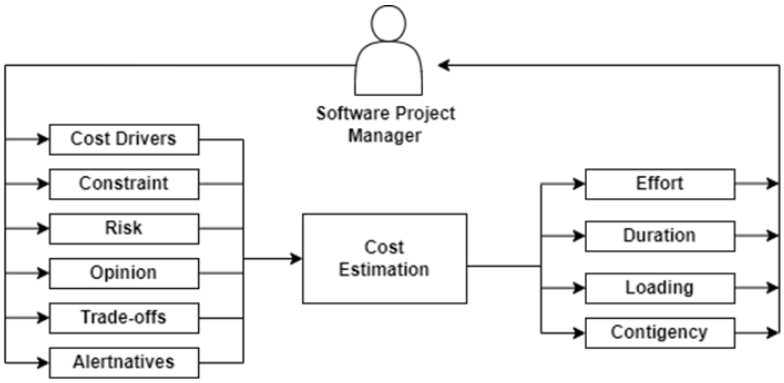


Figure 3: the typical process of Software Cost Estimation

The process shown in Figure 1 is a critical practice to perform analysis and predict all the possible risks and trade-offs in the costing procedure. This process gives the software project manager the possibility to investigate the software project and filter out the evaluated risk possibilities to accomplish the cost of the target software. It`s worth keeping in mind that the project manager and the various department teams work together to realize the final target cost. There are a variety of input variables that are used in cost estimation as shown in Figure 1, cost drivers, constraint, risk, opinion, trade-offs, and alternatives. On the other hand, the outcome of the estimation is evaluated concerning effort, duration, loading, and contingency. In this next section, there will be a focus on the different taxonomies and challenges that are linked with Software Cost Estimation.

6.3.1 Taxonomies

Currently, several cost estimation models are already available for Software Cost Estimation e.g., COCOMO (Constructive Cost Model), COCOMO II, Function Point Analysis (FPA), PERT (Program Evaluation and Review Technique), etc. Each of these models has its strengths and weaknesses, and the suitability of a particular model depends on the project's specific requirements, constraints, and scope. Further detail on the advantages and disadvantages are described in the section below. Out of all these models, the most frequently adopted in the software industry is COCOMO because of its simplicity, flexibility, historical data, and wide applicability. Apart from this Popular SCE model, standard taxonomy cost prediction methods also exist:

- Analogous Estimating: Analogous Estimating is a method used in project management to estimate the cost or duration of a current project by comparing it to a similar completed project. This method is also known as top-down estimating because it involves using data from a previous project to estimate the cost of the current project. Analogous Estimating is often used when limited information about the current project is available. [3]

- Bottom-Up Estimating: Bottom-Up Estimating is a method used in project management to estimate the cost or duration of a project by breaking it down into individual components and estimating the cost or duration of each component. The total cost or duration of the project is then calculated by adding up the costs or durations of each component. This method is more accurate than Analogous Estimating but can be more time-consuming. [4]

- Three-Point Estimating: Three-Point Estimating is a method that involves estimating three scenarios for each project component: an optimistic estimate,

a pessimistic estimate, and a most likely estimate. The final estimate is calculated using a weighted average of the three estimates. [5]

These are just a few examples; the appropriate method depends on the specific use case and characteristics of the project.

6.3.2 Challenges of existing methods

As previously stated, there are still several challenges that must be tackled when utilizing these methods. Some of the key challenges linked to the current approaches are:

6.3.2.1 Analogous Estimating [3]

- Finding a suitable reference project: The accuracy of Analogous Estimating is dependent on the similarity between the current project and the reference project used for estimation. Finding a suitable reference project can be challenging, especially if the current project is unique or complex.

- Historical data: Analogous Estimating relies on historical data from a finished project to estimate the cost of the current project. However, there may be limited historical data available, making it difficult to accurately estimate the cost of the current project.

- Project details: Analogous Estimating relies on the assumption that the details of the current project are similar to the reference project used for estimation. However, if there are significant differences in the details of the two projects, the estimate may not be accurate.

- Assumptions and biases: Analogous Estimating involves making assumptions about the current project based on the reference project used for estimation. These assumptions can be subject to biases, such as optimism bias, which can lead to inaccurate estimates.

- Lack of transparency: Analogous Estimating can be less transparent than other cost estimation methods, as it relies on the expertise of the estimator to produce accurate estimates. This can make it difficult for stakeholders to understand and validate the estimates.

6.3.2.2 Bottom-Up Estimating [4]

- Time-consuming: Bottom-Up Estimating can be a time-consuming process, as it involves breaking down the project into smaller tasks and estimating the cost of each task individually. This can be a significant challenge for large and complex software development projects, as it can require a lot of time and effort to produce accurate estimates.

- Lack of expertise: Bottom-Up Estimating requires a deep understanding of software development processes and practices. If the project team lacks expertise in cost estimation, it can be challenging to produce accurate estimates.

- Changing requirements: Software development projects often have changing requirements, which can make it challenging to estimate costs

accurately. As requirements change, estimates may need to be revised, which can add time and cost to the project.

- Historical data: See the explanation in the section above.
- Human bias: See the explanation in the section above.

6.3.2.3 Three-Point Estimating [5]

- Subjectivity: Three-Point Estimating relies on the judgment of the estimator to determine the optimistic and pessimistic estimates of a task. This subjectivity can lead to inaccurate estimates if the estimator is biased or lacks expertise.
- Overconfidence: Estimators may be overconfident in their ability to estimate the cost of a task, leading to overly optimistic estimates. This can lead to cost overruns and project delays later in development.
- Lack of transparency: See explanation in section 'Analogous Estimating'.
- Historical data: See explanation in section 'Analogous Estimating'.
- Changing requirements: See the explanation section above.

6.4 Data sources

Several data sources regarding Software Cost Estimation are already available, they are used to train artificial intelligence models e.g., the cocomo81, cocomonasa1, cocomonasa v2, Desharnais, China, etc. datasets. For my further research, I propose to work on a variety of data sources to evaluate the model's performance in different scenarios because this is the foremost representation of the real world.

6.4.1 Most common

The most used data sources are cocomo'81, NASA, and Desharnais. In 2019 the number of projects using these data sources looks like this:

Data sources	# of projects	Methodologies
Desharnais	Unk	GEHO-NF
NASA + industrial	Ninety-nine projects	FNN-SEERSEM, SEERSEM
COMOMO`81	Sixty-nine projects	FNN,COCOMO

Table 1: Comparison data sources with their number of projects and methodologies.

6.4.2 EDA Data sources

In this section, a clarification regarding the contents of the listed data sources will be available. Understanding the data in a dataset is an essential first step to creating an Artificial Intelligence model. This also allows for a thorough Exploratory Data Analysis (EDA) of the provided data.

6.4.2.1 Desharnais

The Desharnais data source is a cost estimation dataset that contains eighty records. The first 5 records in the dataset are shown in the Figure (Figure x) below.

	Project	TeamExp	ManagerExp	YearEnd	Length	Effort	Transactions	Entities	PointsAdjust	Envergure	PointsNonAjust	Language
0	1.0	1.0	4.0	85.0	12.0	5152.0	253.0	52.0	305.0	34.0	302.0	b'1'
1	2.0	0.0	0.0	86.0	4.0	5635.0	197.0	124.0	321.0	33.0	315.0	b'1'
2	3.0	4.0	4.0	85.0	1.0	805.0	40.0	60.0	100.0	18.0	83.0	b'1'
3	4.0	0.0	0.0	86.0	5.0	3829.0	200.0	119.0	319.0	30.0	303.0	b'1'
4	5.0	0.0	0.0	86.0	4.0	2149.0	140.0	94.0	234.0	24.0	208.0	b'1'

Figure 4: First 5 records in the data source.

The table below will explain the meaning of the "effort multipliers".

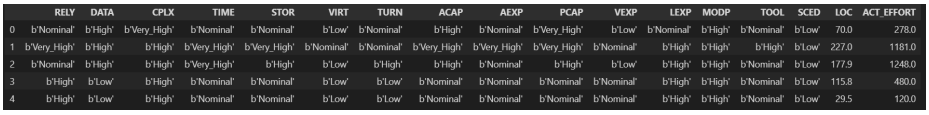
Project	This column indicates the **required software reliability**.
TeamExp	This column indicates the **team experience (in years)**.
ManagerExp	This column indicates the **manager's experience (in years)**.
YearEnd	This column indicates the **year that a project was finished**.
Length	This column indicates the **project length**.
Effort	This column indicates the **Actual effort (in person-hours)**.
Transactions	This column indicates the **count of basic logical transactions in the system**.
Entities	This column indicates the **number of entities in the systems data model**.
PointsAdjust	This column indicates the **application experience**.
Envergure	This column indicates the **size of the project**.
PointsNonAdjust	This column indicates the **number of function points in a project**.
Language	This column indicates the **programming languages**.

Table 2: columns and their explanations.

6.4.2.2 NASA

The COCOMO NASA data source is based on the COCOMO`81 but instead of classifying the "effort multipliers" in 'low', 'nominal', 'high', and 'very_high' instead of numeric values. Note these classifications are based on the standard numeric values that are shown in Figure 3 in the section below.

The first five records in the dataset are shown in the Figure (Figure 3) below.



	RELY	DATA	CPLX	TIME	STOR	VIRT	TURN	ACAP	AEXP	PCAP	VEXP	LEXP	MODP	TOOL	SCED	LOC	ACT_EFFORT
0	b'Nominal'	b'High'	b'Very_High'	b'Nominal'	b'Nominal'	b'Low'	b'Nominal'	b'High'	b'Nominal'	b'Very_High'	b'Low'	b'Nominal'	b'High'	b'Nominal'	b'Low'	70.0	278.0
1	b'Very_High'	b'High'	b'High'	b'Very_High'	b'Very_High'	b'Nominal'	b'Nominal'	b'Very_High'	b'Very_High'	b'Very_High'	b'Nominal'	b'High'	b'High'	b'High'	b'Low'	227.0	1181.0
2	b'Nominal'	b'High'	b'High'	b'Very_High'	b'High'	b'Low'	b'High'	b'High'	b'Nominal'	b'High'	b'Low'	b'High'	b'High'	b'Nominal'	b'Low'	177.9	1248.0
3	b'High'	b'Low'	b'High'	b'Nominal'	b'Nominal'	b'Low'	b'Low'	b'Nominal'	b'Nominal'	b'Nominal'	b'Nominal'	b'High'	b'High'	b'Nominal'	b'Low'	115.8	480.0
4	b'High'	b'Low'	b'High'	b'Nominal'	b'Nominal'	b'Low'	b'Low'	b'Nominal'	b'Nominal'	b'Nominal'	b'Nominal'	b'High'	b'High'	b'Nominal'	b'Low'	29.5	120.0

figure 5: First 5 records in the data source.

The table below will explain the meaning of the "effort multipliers."

rely	This column indicates the **required software reliability**.
data	This column indicates the **database size**.
cplx	This column indicates the **process complexity**.
time	This column indicates the **time constraint for the CPU**.
stor	This column indicates the **main memory constrain**t.
virt	This column indicates the **machine volatility**.
turn	This column indicates the **turnaround time**.
acap	This column indicates the **analyst's capability**.
aexp	This column indicates the **application experience**.
pcap	This column indicates the **programmer's capability**.
vexp	This column indicates the **virtual machine experience**.
lexp	This column indicates the **language experience**.
modp	This column indicates **modern programming practices**.
tool	This column indicates the **use of software tools**.
sced	This column indicates the **schedule constraint**.
loc	This column indicates the **Lines of code**.
act	This column indicates the **actual value**.

6.4.2.3 COCOMO`81

The COCOMO software cost model measures effort in calendar months of 152 hours (and includes development and management hours). COCOMO assumes that the effort grows more than linearly on software size. ("Comparative Study of the Performance of M5-Rules Algorithm with ...") Following the formula:

$$months = a \times (KSLOC^{b}) \times (EM1 \times EM2 \times EM3 \times \dots)$$

"a" and "b" are domain-specific parameters; "KSLOC" is estimated directly or computed from a function point analysis; and "c" is the product of over a dozen "effort multipliers."

The first 5 records in the dataset are shown in the Figure (Figure 6) below.

	rely	data	cplx	time	stor	virt	turn	acap	aexp	pcap	vexp	lexp	modp	tool	sced	loc	actual
0	0.88	1.16	0.70	1.0	1.06	1.15	1.07	1.19	1.13	1.17	1.1	1.00	1.24	1.10	1.04	113.0	2040.0
1	0.88	1.16	0.85	1.0	1.06	1.00	1.07	1.00	0.91	1.00	0.9	0.95	1.10	1.00	1.00	293.0	1600.0
2	1.00	1.16	0.85	1.0	1.00	0.87	0.94	0.86	0.82	0.86	0.9	0.95	0.91	0.91	1.00	132.0	243.0
3	0.75	1.16	0.70	1.0	1.00	0.87	1.00	1.19	0.91	1.42	1.0	0.95	1.24	1.00	1.04	60.0	240.0
4	0.88	0.94	1.00	1.0	1.00	0.87	1.00	1.00	1.00	0.86	0.9	0.95	1.24	1.00	1.00	16.0	33.0

Figure 6: First 5 records in the data source.

The table below will explain the meaning of the "effort multipliers."

rely	This column indicates the **required software reliability**.
data	This column indicates the **database size**.
cplx	This column indicates the **process complexity**.
time	This column indicates the **time constraint for the CPU**.
stor	This column indicates the **main memory constrain**t.
virt	This column indicates the **machine volatility**.
turn	This column indicates the **turnaround time**.
acap	This column indicates the **analyst's capability**.
aexp	This column indicates the **application experience**.
pcap	This column indicates the **programmer's capability**.
vexp	This column indicates the **virtual machine experience**.
lexp	This column indicates the **language experience**.
modp	This column indicates **modern programming practices**.
tool	This column indicates the **use of software tools**.
sced	This column indicates the **schedule constraint**.
loc	This column indicates the **Lines of code**.
act	This column indicates the **actual value**.

Note: For Columns **acap**, **pcap**, **Aexp**, **modp**, **tool** and **vexp** a higher value indicates decreased effort as shown in Figure 5.

The standard numeric values of the effort multipliers are:

	very low	low	nominal	very high	extra high	productivity range
acap	1.46	1.19	1.00	0.86	0.71	2.06
pcap	1.42.	1.17	1.00	0.86	0.70	1.67
aexp	1.29	1.13	1.00	0.91	0.82	1.57
modp	1.24.	1.10	1.00	0.91	0.82	1.34
tool	1.24	1.10	1.00	0.91	0.83	1.49
vexp	1.21	1.10	1.00	0.90		1.34
lexp	1.14	1.07	1.00	0.95		1.20
sced	1.23	1.08	1.00	1.04	1.10	e
stor			1.00	1.06	1.21	1.56	-1.21
data		0.94	1.00	1.08	1.16		-1.23
time			1.00	1.11	1.30	1.66	-1.30
turn		0.87	1.00	1.07	1.15		-1.32
virt		0.87	1.00	1.15	1.30		-1.49
cplx	0.70	0.85	1.00	1.15	1.30	1.65	-1.86
rely	0.75	0.88	1.00	1.15	1.40		-1.87

Figure 7: Value classifications.

6.5 Neural Network implementation

The methodologies as stated in section "3.2 Challenges of existing methods" have some critical challenges to overcome from whom most of which are human error and lack of experience. A solution to this problem is by using the power of Artificial Intelligence (AI). The AI field is one of the fastest-growing sectors in IT at the moment. some examples of recent advancements in this field are ChatGPT and DALL-E both of which use Natural language Processing (NLP) techniques.

6.5.1 Evolution

The use of Artificial Intelligence in Software Cost Estimation has been a topic of research for several decades, but its practical implementation started gaining traction in the early 2000s.

One of the earliest examples of the use of AI in Software Cost Estimation was the COCOMO II model developed by Barry Boehm in 2000. COCOMO II is a constant software cost estimation model that uses a set of equations and algorithms to estimate the effort, time, and cost required to develop a software project. The model uses Machine Learning techniques to learn from historical data and refine its estimates over time.

Since then, many researchers and practitioners have explored the use of different AI techniques such as Neural Networks, decision trees, fuzzy logic, and genetic algorithms for Software Cost Estimation. These techniques have been applied to various software development processes, including agile development, traditional waterfall development, and iterative development.

Today, many Software Cost Estimation tools and platforms use AI and Machine Learning algorithms to provide accurate and reliable estimates. These tools leverage large datasets of historical project data to identify patterns and correlations and make predictions based on those patterns.

6.5.2 Advantages

There are several advantages of using AI in Software Cost Estimation, some examples are listed below [6]:

- Improved accuracy: AI can analyze large datasets and identify patterns and correlations that may not be obvious to humans. This allows for more accurate cost estimations based on historical data.

- Increased efficiency: AI can automate many of the tasks involved in software cost estimation, such as data analysis and modeling. This reduces the time and effort required to generate estimations and allows teams to focus on other critical tasks.

- Scalability: Using AI it is possible to scale for handling large and complex software projects, which may be challenging for humans to estimate accurately. This allows organizations to estimate costs for projects of any size and complexity.

- Consistency: AI algorithms can provide consistent estimates based on pre-defined rules and parameters, which can reduce the variability that may occur when humans estimate costs.

- Continuous learning: AI algorithms can learn from new data and adjust their estimates over time, which can improve the accuracy of cost estimates as more data becomes available.

Overall, the use of AI in software cost estimation can improve accuracy, efficiency, scalability, and consistency, and can use continuous learning as time passes, which can lead to better project planning and decision-making.

6.6 Evaluation metrics

In this section, the evaluation metrics that are used to see how accurately a model will perform will be discussed. The metrics that are listed here will frequently occur in the next section.

6.6.1 MRE (Magnitude of relative error)

$$\text{Formula MRE: } \frac{\text{actual-estimated}}{\text{actual}} \quad (1)$$

6.6.2 MMRE (Mean Magnitude of Relative Error)

The MMRE calculates the average relative difference between the actual values and the predicted values, expressed as a percentage of the actual values. The lower the MMRE, the better the accuracy of the prediction model.

$$\text{Formula MMRE: } \frac{1}{n}\sum_{x=1}^{n} \text{MRE} \quad (2)$$

6.6.3 MSE (Mean Squared Error)

The MSE is a commonly used metric in statistics and machine learning to measure the average squared difference between the predicted values and the

actual values. In other words, it is a way to quantify how far off the predicted values are from the actual values.

Formula MSE: $\frac{1}{n}\sum_{x=1}^{n}(Y_i - \hat{Y_i})^2$ (3)

6.6.4 RMSE (Root Mean Squared Error)

The RMSE measures the standard deviation of the residuals and is expressed in the same units as the response variable. The lower the RMSE, the better the accuracy of the prediction model.

Formula RMSE: $\sqrt{\sum_{i=1}^{n}\frac{(\text{Predicted}_i\text{-Actual}_i)^2}{n}}$ (3)

6.6.5 MdMRE (Median Magnitude of Relative Error)

The MdMRE measures the median relative difference between the actual values and the predicted values expressed as a percentage of the actual values. The lower the MdMRE, the better the accuracy of the prediction model.

The use of median instead of mean in this metric makes it less sensitive to outliers, which may be useful in some contexts where the data is skewed or has extreme values.

Formula MdMRE: median(MRE) (4)

6.6.6 PRED (Prediction accuracy)

The PRED is the accuracy measure; thus, the greater the PRED value, the higher the model's accuracy score. ("Modeling of software project effort estimation: a comparative ...") It is defined as the percentage of projects whose predicted values are within (N%) of their actual values.[3]

Formula PRED: $\frac{100}{N} * \sum_{n=1}^{N}\left\{1, \text{if } MRE_n \leq \frac{T}{100}\right.$ (0, if not 1) (5)

6.6.7 MAE (Mean Absolute Error)

The MAE measures the average absolute difference between the actual and predicted values of the target variable. It is a commonly used metric for regression problems, where the goal is to predict a continuous numerical value. A lower MAE value indicates better accuracy of the model.

Formula MAE: $\frac{1}{n} * \sum_{i=1}^{n}\left|(y\text{-}y')\right|$ (6)

6.7 Methodologies

In this section, the focus will lay on existing machine learning methodologies that have already been researched will be considered e.g., GEHO-NFN, AA-SEE, and ANN. These are just some examples of relevant studies.

6.7.1 Genetic elephant herding optimization (GEHO) based Neuro-fuzzy network (GEHO-based NFN).

In the first research paper, the most efficient method for solving software cost estimation problems is the GEHO-based NFN. In this proposed method, a software cost estimation model is devised and developed using the Genetic elephant herding optimization-based Neuro-fuzzy network (GEHO-based NFN) model. Here, the software data was collected from the database, and to select the prominent features, they used a correlation coefficient-based feature selection. After, they designed a Neuro-fuzzy system that is further trained to get the optimum weights using genetic algorithm-based. For the entire formulated method, an implemented Neuro-fuzzy model and GA-based EHO were combined to get the optimized results.

The EHO (Elephant Herding Optimization) technique is a swarm-based meta-heuristic optimization algorithm used for solving optimization problems. It is based on the behavior of elephant herds, where the leader elephant, or the "matriarch", leads the herd toward food sources and water. The algorithm simulates the behavior of elephant herds by creating a population of solutions, where each solution represents an elephant. The algorithm uses a herd leader, which is the solution with the best fitness value, to guide the other solutions toward the optimal solution. During the optimization process, the algorithm uses two main strategies: exploitation and exploration. Exploitation involves exploiting the best solution found so far, while exploration involves searching for new solutions in the search space. These two strategies are used to balance convergence towards the global optimum and diversification of the search process.

Using this method, these results for the evaluation metrics ranging over 60% of the training data are achieved. Note, the model was trained on the COCOMONASA2 dataset.

MMRE (6.6.2)	0.261
RMSE (6.6.4)	0.236
MdMRE (6.6.5)	0.246
PRED (6.6.6)	52.63%

Table 5: Evaluation metrics with their corresponding values.

6.7.2 Optimized Learning-based Cost Estimation (OLCE)

In the second research paper, an Optimized Learning-based Cost Estimation (OLCE) approach is proposed. the main purpose of OLCE is to implement automation towards software development as well as perform further training for Neural Network platforms using a search-based optimization scheme. This scheme is meant for evolving the units of a Neural Network with its structure and parameters of learning to perform a prognostic evaluation of the stability of Software Cost Estimation. According to their proposed search-based optimization

scheme, the scheme constructs an initial set of populations followed by scoring and scaling it. It further retains the best outcome while parents are selected to generate an outcome that is further used for scoring and scaling the population.

In all the above-mentioned steps of operation, the scheme follows three core rules to finetune the outcome of the population:

- Rule for Selection: The main principle that selects the individual referred to as the parent, resulting in a modified population in successive rounds of operation.

- Rule for Aggregation: This is the second principle that generates an updated population by merging information from two parents.

- Rule for Transformation: This is the tertiary principle that transforms a single parent to produce a modified set of the single population randomly.

The unique features of the OLCE scheme are as follows:

1. The scheme is capable of producing multiple outcomes of a population, with the stopping point of iteration determined by achieving the optimal score according to the fitness function.

2. The scheme employs a randomly selected number to determine the next cycle of population, which enhances the system's responsiveness and speed.

3. It is capable of processing multiple evaluation functions aimed at increasing the convergence rate.

To understand the unique features of their proposed scheme, Figure 6 will elaborate on the operation of the working in comparison to the conventional technique.

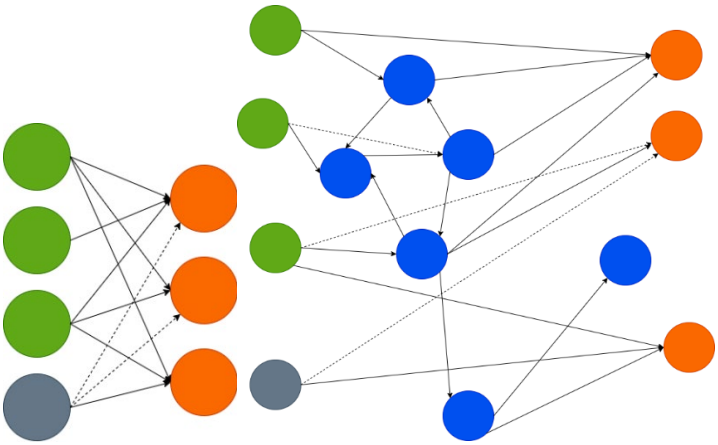


Figure 8: Standard Neural Network architecture and OLCE architecture

For further information regarding the working of this scheme: OLCE: Optimized Learning-based Cost Estimation for Global Software Projects by Research Square

6.7.3 ANN with NEAT optimization (Artificial Neural Network with Neuro-Evolution of Augmenting Topology).

In the third research paper, an Artificial Neural Network with Neuro-Evolution of Augmenting Topology is proposed. Artificial Neural Networks (ANNs) are a type of Machine Learning model inspired by the structure and function of the human

brain. (“About Artificial Neural Network - Engine Related input/output data?”) ANNs consist of interconnected nodes or neurons organized into layers. The input layer receives the input data, and the output layer produces the output. (“Neural Networks: Sigmoid Functions and Output Layers”) There may also be one or more hidden layers in between. Each neuron applies a non-linear activation function to the sum of its weighted inputs, and this activation value is then passed on to the next layer.

NEAT is an algorithm for optimizing ANNs through the process of evolution. NEAT starts with a minimal network structure and evolves it over time by adding or removing neurons and connections. NEAT maintains a population of different network structures, and each structure is evaluated for its fitness based on how well it performs on a given task. The fittest networks are then selected to produce offspring through crossover and mutation, which are added back to the population for the next generation. NEAT allows the topology of the network to evolve along with the weights and biases, which can lead to more efficient and effective networks.

For further information regarding the working of this scheme: Effective ANN Model based on Neuro-Evolution Mechanism for Realistic Software Estimates in the Early Phase of Software Development

Using this method, these results for the evaluation metrics ranging over 60% of the training data are achieved. Note, the model was trained on the COCOMONASA2 dataset.

MMRE (6.6.2)	0.113581
RMSE (6.6.4)	39.335067
MSE (6.6.3)	1547.247493
MAE (6.6.7)	22.151230
PRED (6.6.6)	68.91522

Table 6: Evaluation metrics with their corresponding values.

6.8 Gap in knowledge

One of the gaps in knowledge with utilizing Machine Learning algorithms in Software Cost Estimation is the lack of large-scale, high-quality datasets for training and testing the Machine Learning models. Most Software Cost Estimation datasets are relatively small, and some may be of questionable quality, which can affect the overall performance and generalizability of the AI models. This also makes it harder to use more powerful techniques like Deep Learning CNNs (Convolutional Neural Networks).

Another gap is the interpretability and transparency of Machine Learning models. Machine Learning algorithms are considered "black box" models, meaning that it can be difficult to understand on what assumptions and variables they made their prediction. This makes it challenging to explain the reasoning behind cost estimates to stakeholders, this can be significant in software development projects. The use of decision trees can eliminate this problem, but decision trees are the most basic form of classification thus they are not optimized for a large set of variables.

Furthermore, there may be challenges in identifying relevant features or variables to include in Machine Learning models for Software Cost Estimation. This is because many factors can influence software development costs, some of which may be difficult to measure or quantify. As a result, identifying the most important features to include in Machine Learning models can be a challenging task.

6.8.1 Future proposal

My future proposal is to explore the possibilities of Deep Reinforcement Learning (DRL) in correlation to Software Cost Estimation. Deep Reinforcement Learning is a technique that uses Reinforcement Learning (RL) and Deep Learning (DL). Reinforcement Learning (RL) is a learning process where an agent learns to make decisions in challenging environments by interacting with the environment and receiving feedback in the form of a reward signal. The agent takes actions that affect the environment's state and receives feedback on how well it is performing. The aim is to learn a policy that maximizes the accumulated reward over time. It would be theoretically possible to use in Software Cost Estimation (SCE) because it is a complex problem with many variables and uncertainties. DRL can learn from experience and adapt to changing conditions which is important in software engineering which is a rapidly changing environment.

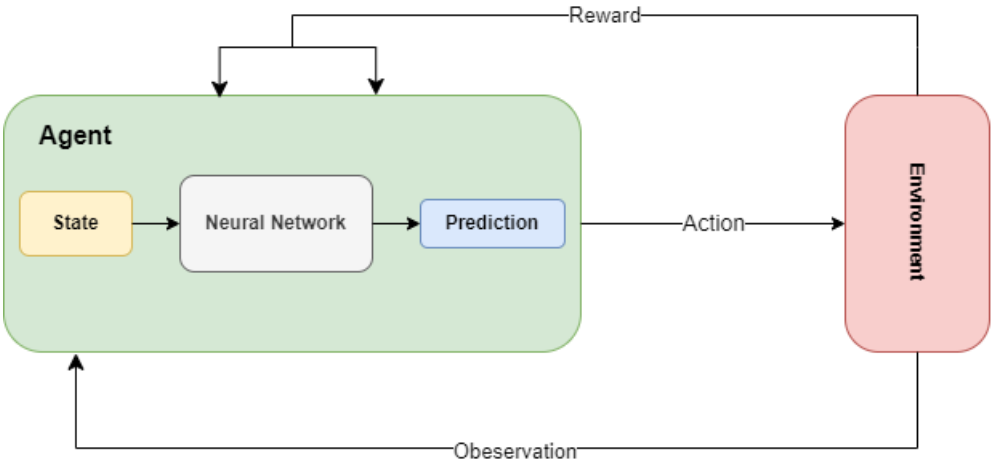


Figure 9: generalized DLR process.

6.9 Conclusion

In conclusion, developing algorithms and models to aid the process of Software Cost Estimation has come a long way. I have discovered that the most used technique to solve SCE problems are Machine Learning solutions. These solutions can yield fairly good results in terms of accuracy and values. But with the recent advancements of the rapid development in Deep Learning, it may be interesting to conduct research where the two techniques (ML and DL) are combined. That is why I propose a Deep Reinforcement Learning (DLR) architecture which is commonly used in solving very complex problems, seeing how many variables and environmental factors are used in Software Cost Estimation. It would be a perfect fit. The use of DLR in Software Cost Estimation is new and could be a huge advancement over the traditional machine learning approaches.

7 RESEARCH PAPER: SOFTWARE COST ESTIMATION USING SYNTHETIC DATA AND ENSEMBLE MODELING

Abstract

One of the most vital steps when beginning a new software engineering project is Software Cost Estimation (SCE). SCE aids in resource allocation, risk management, and decision-making, and is correlated with a project's success or failure. Software Cost Estimation (SCE) is subject to human bias, hence artificial intelligence (AI) and machine learning (ML) are being used in research to find possible solutions. This paper will examine the significance of Software Cost Estimation (SCE) and how the use of Artificial Intelligence (AI) can automate this process, making it as easy and efficient as possible. Therefore, the proposed research suggests an ensemble algorithm trained on synthetic data based on a Use-case Point (UCP) and a Functional Point Analysis (FPA) data set. Using the UCP71 dataset enhanced with synthetic data, the study outcomes show that using the ensemble model to combine different Machine Learning and Deep Learning algorithms (Linear Regression, Lasso, and K-Nearest Neighbors) results in a Mean Absolute Error (MAE) of 33.01 and an accuracy score of 100 percent considering that the predicted values fall in the 25 percent range of the actual value.

Keywords: Software Cost Estimation, Artificial Intelligence, Machine Learning, Ensemble model, Neural Networks, Synthetic data.

7.1 Introduction

Software Cost Estimation (SCE) is critical to any new software development project. Its primary objective is to predict the software project development cost before the project phases' commencement. This includes ascertaining the resources required for development, the time needed for development, and the estimated effort required to achieve the end goal of the development phase. The success or failure of a project is closely tied to the accuracy of the Software Cost Estimation process. Regrettably, Software Cost Estimation is a challenging due to several factors such as lack of expertise, historical data, human bias, and overconfidence. To address these issues, Artificial Intelligence (AI) models have been developed over the past decade to enhance the accuracy of Software Cost Estimation. AI is rapidly progressing and has shown significant promise in automating and making predictions based on presented data. This progress is attributable to more powerful hardware and the availability of public data. While the COCOMO model is the most commonly adopted model in the software industry due to its simplicity, flexibility, availability of historical data, and wide applicability, it also has its strengths and weaknesses. Consequently, researchers have been exploring various methods to improve the accuracy of Software Cost Estimation. With the increasing availability of data sources regarding SCE, many researchers have developed an interest in the topic, paving the way for further exploration and advancement in this field. Several Machine Learning techniques have been applied to SCE, such as Lasso, K-Nearest Neighbors, Linear Regression (LR), and Support Vector Machines (SVM). These algorithms can achieve decent accuracy and minimal loss. In this paper, we will explore the of newer Machine and Deep Learning techniques. Our proposed research investigates the possibilities of synthetic data and ensemble models. We plan to use synthetic data to address the lack of historical data and

improve the overall consistency of the data. This data will be utilized to train and evaluate the performance of our models. The best-performing models will be incorporated into an ensemble model using the stacking technique, which will make a final prediction leveraging the strengths of each of the previously mentioned models. This technique can solve complex problems characterized by numerous variables and an adaptive environment similar to that of software engineering. This paper is divided into 11 sections: 1 Introduction, 2 Related works, 3 Software Cost Estimation (SCE), 4 Data sources, 5 Neural network implementation, 6 Evaluation metrics, 7 Used algorithms formulations, 8 Used methodologies, 9 Gap in knowledge, 10 Conclusion, and 11 Literature cited. The application of Artificial Intelligence in Software Cost Estimation is gaining importance, as it has the potential to improve accuracy, reduce costs, and enhance decision-making processes in software development projects, thereby underscoring the need for a broader adoption and standardization regarding the use of ensemble models trained on synthetic data.

7.2 Related works

Several studies have delved into the utilization of synthetic data. For instance, Trivellore E. Raghunathan (2020) conducted an in-depth review emphasizing the significance of data, and how synthetic data can make research more accessible for scholars.

Concurrently, Yingzhou Lu, Huazheng Wang, and Wenqi Wei (2023) explored the application of machine learning for synthetic data.

Meanwhile, Marta Lenatti, Alessia Paglialonga, Vanessa Orani, Melissa Ferretti, and Maurizio Mongelli (2023) researched Characterization of Synthetic Health Data Using Rule-Based Artificial Intelligence Models. Benjamin N Jacobsen explored the politics of synthetic data in regards to Machine Learning.

Lastly Necmi Gürsakal, Sadullah Çelik & Esma Birişçi (2023) made an in-depth introduction to synthetic data. Concluding that synthetic data holds a promising future in the realm of Artificial Intelligence.

Researchers M Maher and JS Alneamy (2022) examined the use of a stacked ensemble model in Software Cost Estimation, employing algorithms like K-Nearest Neighbors (KNN), Support Vector Regression (SVR), Artificial Neural Networks (ANN), Bayesian Regressor, Linear Regressor, and Random Forest. Their approach resulted in a Mean Magnitude of Relative Error (MMRE) of 0.14.

Additionally, Jin Gang Lee, Hyun-Soo Lee, Moonseo Park and JoonOh Seo (2022) have made an early-stage cost estimation model for power generation project with limited historical data using ensemble modelling. This produced an error rate of 12,90 percent on average.

Lastly, Priya Varshini A, G, Anitha, Kumari K, Varadarajan, and Vijayakumar (2021) researched a stacked ensemble model using random forest-based algorithms to be implemented in Software cost Estimation. This model was trained on diverse datasets such as Desharnais, China, and COCOMO81.

In the field of Software Cost Estimation, numerous studies have focused on the use of Neural Networks, particularly in the context of Machine Learning. Sudhir Sharma and Shripal Vijayvargiya (2022), for example, developed several Neural Network models using Machine Learning to perform Software Cost Estimation. They benchmarked all the models on different data sources to determine the best-performing model and technique.

Additionally, K Nitalaksheswara Rao, Jhansi Vazram Bolla, Satyanarayana Mummana, CH. V. Murali Krishna, O. Gandhi, and M James Stephen (2022) constructed an optimized, learning-based Cost Estimation model and scheme. This approach struck a balance between accurate predictions and computational efficiency.

Other research has delved into the application of Artificial Neural Networks (ANN). Puppala Ramya, M. Sai Mokshith, M. Abdul Rahman, and N. Nithin Sai (2023) de-veloped a model capable of understanding complex relationships and patterns found in the data sources for Software Cost Estimation.

7.3 Software Cost Estimation

As outlined in Section 1, the importance of Software Cost Estimation (SCE) in new software development projects is paramount. The primary objective of SCE is to provide a predictive analysis of the cost associated with software development before initiating project phases. The term 'cost' encompasses the number of resources to be employed in the development, the time invested in development, and the estimated effort required to reach the end goal of the development phase.

The following figure, Figure 10, portrays the typical process of Software Cost Estimation (SCE). This iterative process involves adjusting various attributes until the cost of the target software is deemed justifiable. While other development teams may provide their input, it is crucial to note that this process is overseen by the Software Project Manager, who is ultimately responsible for making the final decision.

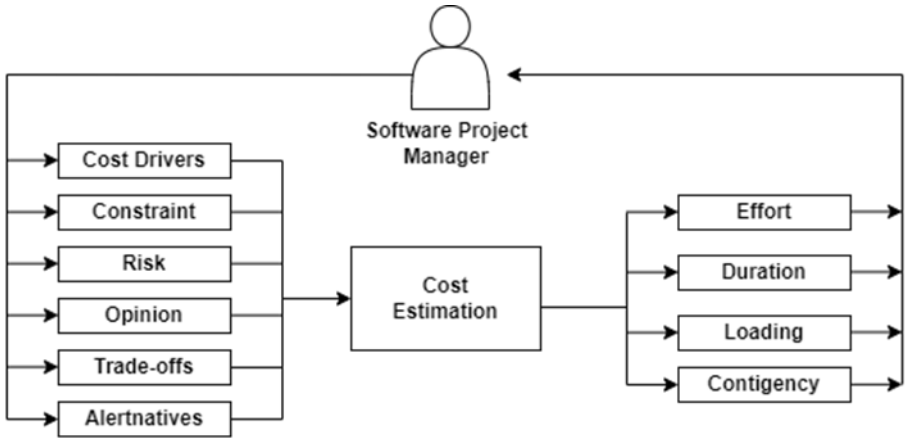


Figure 10: The typical process of Software Cost Estimation [2]

Figure 1 delineates an essential methodology for analysing and predicting potential risks and trade-offs in the software costing process. Software project managers can leverage this process to evaluate the project and identify and mitigate potential risks to achieve the target cost. It's crucial to recognize that the project manager and department teams collaborate to determine the final development cost. Cost estimation comprises various input variables, including cost drivers, constraints, risks, opinions, trade-offs, and alternatives. Concurrently, the outcomes of the estimation are evaluated based on effort, duration, loading, and contingency. In the subsequent section, we will explore various taxonomies and challenges associated with software cost estimation. [2]

7.3.1 Existing methods and key challenges

Presently, a plethora of models exist for Software Cost Estimation, including the Constructive Cost Model (COCOMO), COCOMO II, Function Point Analysis (FPA), and the Program Evaluation and Review Technique (PERT), among others. Each model brings its unique strengths and weaknesses to the table, and the decision of which to employ hinges on the software project's specific requirements, constraints, and scope. A more comprehensive exploration of the advantages and disadvantages of each taxonomy will be provided in the subsequent section. Of all the models, the COCOMO is most prevalently adopted within the software industry owing to its simplicity, flexibility, the availability of historical data, and its wide applicability. These represent generic examples; At the current time no researchers have explored the use of synthetic data in regards to Software Cost Estimation but it has been researched in fields such as economics, agriculture and phishing detection. the method best suited to a project will depend on the project's specific use case and characteristics.

Challenges of existing methods

As previously stated, there are still several challenges that must be tackled when utilising these methods.

Analogous Estimating

Analogous Estimating, also known as top-down estimating, is a project management technique that projects the cost or duration of a current initiative by comparing it to a similar, previously completed project. This approach utilizes data from a past project to estimate the cost of the current one, making it particularly valuable when detailed information about the current project is limited. The precision of Analogous Estimating hinges on identifying a reference project that mirrors the current project closely. However, discovering a fitting reference project can be challenging, particularly for unique or complex projects. Analogous Estimating relies on historical data from a completed project to predict the cost of the current project. Nonetheless, a lack of sufficient historical data could present difficulties in obtaining an accurate cost estimate for the present project. Analogous Estimating presumes that the details of the current project align closely with the reference project used for estimation. If significant differences exist between the two projects, the resulting estimate may not be accurate. Analogous Estimating necessitates making assumptions about the current project based on the reference project used for estimation. These assumptions can be prone to biases, such as optimism bias, which may lead to imprecise estimates. [3]

Compared to other cost estimation techniques, Analogous Estimating may lack transparency as it largely depends on the expertise of the estimator to yield accurate estimates. As a result, stakeholders may encounter difficulties in understanding and validating the estimates.

Bottom-Up Estimating

Bottom-Up Estimating is a project management technique that entails breaking down a project into smaller components, then estimating the cost or duration of each component to ascertain the total cost or duration of the project. This approach is generally regarded as more accurate than Analogous Estimating, but it requires more time and effort to implement. This method can be time-consuming, which might be challenging for larger and more complex software development projects. The process involves dividing the project into smaller tasks and estimating the cost of each task individually. This detailed and meticulous process can demand significant effort and time, making it potentially less feasible for large-scale projects. The successful implementation of Bottom-Up Estimating depends on a solid understanding of software development processes and practices. If the project team lacks expertise in cost estimation or a deep knowledge of the tasks involved in the project, it can be difficult to generate accurate estimates. One common challenge in software development projects is the changing nature of project requirements. These evolving requirements can complicate the task of accurate cost estimation. As the project requirements shift, initial estimates may need to be revised, leading to additional time and costs for the project. Like Analogous Estimating, Bottom-Up Estimating also relies on historical data from past projects to inform its estimates. However, just like in Analogous Estimating, the lack of sufficient or relevant historical data can pose challenges to achieving accurate cost estimation for the current project. [4]

Human bias is also a common issue in cost estimation techniques. This bias may skew the estimates, leading to either overestimation or underestimation of project costs. It's important to be aware of this potential bias and take steps to mitigate its impact on the cost estimation process.

Three-Point Estimating

The Three-Point Estimating technique heavily relies on the evaluator's judgement to formulate the optimistic and pessimistic estimates for a task, introducing an element of subjectivity. This could potentially lead to inaccurate estimates if the evaluator is biased, overly optimistic or pessimistic, or lacks the required expertise to make a well-informed judgement. Overconfidence can pose a significant problem in the Three-Point Estimating technique. Project managers may overestimate their ability to precisely calculate the cost of a task, which could lead to overly optimistic estimates. This optimism can have serious repercussions, leading to cost overruns and project delays later in the development cycle. It's crucial to maintain a balanced and realistic perspective when estimating costs to avoid such pitfalls. As discussed in the 'Analogous Estimating' section, lack of transparency can also pose a challenge. Since Three-Point Estimating heavily relies on the individual evaluator’s judgement, it might be difficult for others, including stakeholders, to understand or validate the estimates. This lack of transparency could potentially lead to disagreements or misunderstandings regarding the estimated cost. Like other cost estimation techniques, Three-Point Estimating also leverages historical data from past projects to inform its estimates. However, as previously mentioned, the absence of sufficient or relevant historical data can pose challenges to achieving accurate cost estimation for the current project. Lastly, changing requirements, a

common challenge in software development projects, can complicate the task of accurate cost estimation. As project requirements shift, initial estimates may need to be revised, leading to additional time and costs. This consideration is vital in the Three-Point Estimating technique, just as it is in other cost estimation methods. [5]

7.4 Methods used

7.4.1 Neural network

The use of Artificial Intelligence in Software Cost Estimation has been a topic of research for several decades, but its practical implementation started gaining traction in the early 2000s.
One of the earliest examples of the use of AI in Software Cost Estimation was the COCOMO II model developed by Barry Boehm in 2000. COCOMO II is a constant software cost estimation model that uses a set of equations and algorithms to estimate the effort, time, and cost required to develop a software project. The model uses Machine Learning techniques to learn from historical data and refine its estimates over time.
Since then, many researchers and practitioners have explored the use of different AI techniques such as Neural Networks, decision trees, fuzzy logic, and genetic algorithms for Software Cost Estimation. These techniques have been applied to various software development processes, including agile development, traditional waterfall development, and iterative development.
Today, many Software Cost Estimation tools and platforms use AI and Machine Learning algorithms to provide accurate and reliable estimates. These tools leverage large datasets of historical project data to identify patterns and correlations and make predictions based on those patterns.

There are several advantages of using AI in Software Cost Estimation, some examples are listed below:

- Improved accuracy: AI can analyze large datasets and identify patterns and correlations that may not be obvious to humans. This allows for more accurate cost estimations based on historical data. [6]

- Increased efficiency: AI can automate many of the tasks involved in software cost estimation, such as data analysis and modeling. This reduces the time and effort required to generate estimations and allows teams to focus on other critical tasks. [6]

- Scalability: Using AI it is possible to scale for handling large and complex software projects, which may be challenging for humans to estimate accurately. This allows organizations to estimate costs for projects of any size and complexity. [6]

- Consistency: AI algorithms can provide consistent estimates based on pre-defined rules and parameters, which can reduce the variability that may occur when humans estimate costs. [6]

- Continuous learning: AI algorithms can learn from new data and adjust their estimates over time, which can improve the accuracy of cost estimates as more data becomes available. [6]

Overall, the use of AI in software cost estimation can improve accuracy, efficiency, scalability, and consistency, and can use continuous learning as time passes, which can lead to better project planning and decision-making.

7.5 Methodology

In this section, the focus will lay on the methodologies implemented in the research e.g., Distribution of data, synthetic data, and the ensemble model.

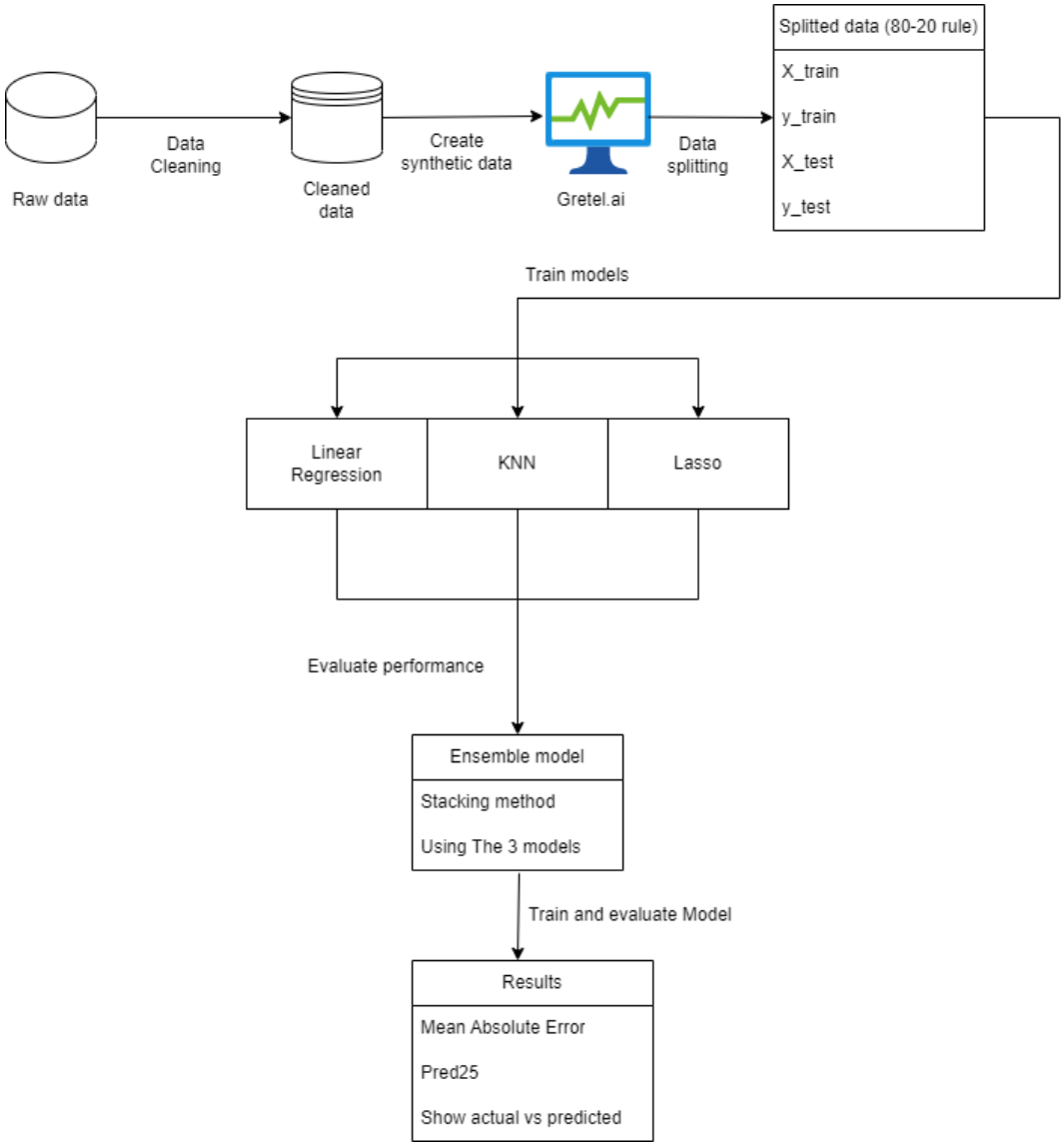


Figure 11: the whole process visualized.

7.5.1 Experimental datasets

There exists a multitude of datasets pertinent to Software Cost Estimation, which serve as a basis for training artificial intelligence models. This includes datasets like cocomo81, cocomonasa1, ISBSG, and China, among others. Each of these datasets specializes in different aspects of Software Cost Estimation, such as Functional Points (FPA), Lines Of Code (LOC), and Use Case Point (UCP) analysis.

In the context of my research, I have elected to focus on the Functional Point Analysis (FPA) and Use-Case Point (UCP) methodologies. This decision stems from the unique combination of advantages and disadvantages that each method brings to the table, including factors like standardization, accuracy, flexibility, and improved communication. By intertwining the use of FPA and

UCP, I plan to leverage their individual strengths while counteracting the potential limitations of each method.

The ISBG dataset, renowned for its extensive and detailed information on historical projects, is one of the most frequently utilized sources for Functional Point Analysis. I have selected this dataset due to its reliability and the comprehensive nature of its project histories. A thorough exploration of this dataset will be presented in the following section.

In terms of training my model on a UCP dataset, I will utilize the UCP71 data source. This dataset is regularly employed to assess the performance of UCP estimation models. Further details on this data source will also be provided in the subsequent section.

This section presents descriptions of the aforementioned data sources. Understanding the data in a dataset is paramount before developing an Artificial Intelligence (AI) model, as this comprehension allows for an in-depth Exploratory Data Analysis (EDA) of the available data.

FPA_ISBSG

The FPA_ISBSG is a Functional Point Analysis (FPA) data source employed for Software Cost Estimation. Comprising 1712 records and 58 columns from an array of industry sectors, this data was assembled by the International Software Benchmarking Standards Group (ISBSG). We will utilize seven of these columns (EI, EO, EQ, ILF, EIF, Size, xIndustrySectorId) to predict the effort required to complete a project:

- EI: Represents the External Inputs.
- EO: Stands for the External Outputs.
- EQ: Denotes the External Inquiries.
- ILF: Indicates the Internal Logical Files.
- EIF: Denotes the External Interface Files.
- xIndustrySectorId: Signifies the Industry Sector.
- Size: Indicates the size of a software project.
- Effort: Represents the effort required to complete a software project.

Column	Count	Mean	Min	Max
EI	1711	140.04	0	9404
EO	1711	117.07	0	3653
EQ	1711	73.74	0	2886
ILF	1711	109.02	0	10821
EIF	1711	35.75	0	1572
Size	1711	495.88	4	19050
Effort	1711	4907.54	10	150040

Table 7: the statistical properties of the columns.

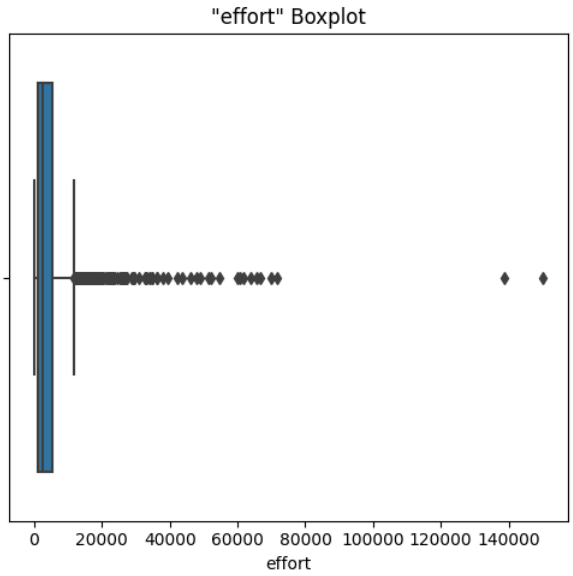


Figure 11: Effort boxplot

UCP71

The UCP71, a Use-Case Point (UCP) data source, is employed for Software Cost Estimation. With 71 records and 29 columns from various industry sectors, this data was collected by the University of Central Florida (UCF). Six of these columns (UAW, UUCW, TCF, ECF, Size, and Effort) will be utilized to predict the effort needed to complete a project:

- UAW: Represents the Unadjusted Actor Weight.
- UUCW: Stands for the Unadjusted Use Case Weight.
- TCF: Denotes the Technical Complexity Factor.
- ECF: Indicates the Environmental Complexity Factor.
- Size: Represents the size of the software project.
- Effort: Indicates the effort required to complete a software project.

Column	**Count**	**Mean**	**Min**	**Max**
UAW	71	10.45	6	19
UUCW	71	385.49	250	610
TCF	71	0.92	0.71	1.11
ECF	71	0.86	0.51	1.08
Size	71	328.56	33.39	398.50
Effort	71	6571.26	5775	7970

Table 8: the statistical properties of the columns.

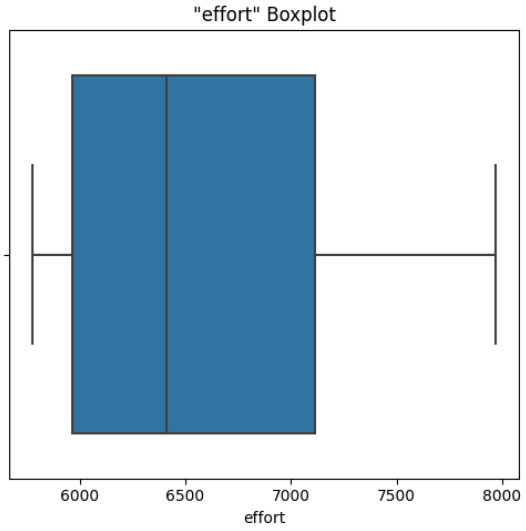


Figure 12: Effort boxplot

7.5.2 Validation method

In our proposed methodology, we will split the data into train and test sets with a distribution of 80 percent for training data and 20 percent for the test data (80-20 rule). This is the most commonly used heuristic for splitting Artificial Intelligence (AI) training data as it provides enough data to train and evaluate AI models.

7.5.3 Handling a small dataset issue

Synthetic data is data that has been artificially produced, designed to mimic the properties and characteristics of real data. It can be generated using statistical models, machine learning algorithms, or other data generation techniques. Synthetic data is utilized for various purposes such as training models and testing algorithms, serving as a solution to the issue of insufficient historical data.

Rule-based generation is a technique involving the use of predefined rules. These rules are based on the structure and attributes of the original data, and can also be informed by domain knowledge. Rule-based generation is beneficial when there is certainty about the structure and characteristics of the data that needs to be generated. Moreover, it can be employed to generate data that adheres to a specific pattern or distribution.

Model-based generation is another technique that employs machine learning algorithms to generate data that mirrors the properties of the original data. These algorithms are trained on the original data and are able to discern relationships and patterns to generate synthetic data that is structurally similar.

The Monte Carlo simulation is a technique that uses statistical models, and simulations to generate synthetic data. It creates random samples using a probability distribution that resembles the original data and employs these samples to generate synthetic data.

Advantages and limitations of synthetic data include its ability to mitigate data scarcity and facilitate data sharing. However, synthetic data does come with certain limitations such as the potential introduction of bias, the challenge in

fully capturing complex relationships in the data, and the lack of diversity in the generated data.

In our case, we employ Gretel.ai to generate synthetic data using the model-based generation technique as this is the most advanced synthetic data generator currently available. The data generated will be based on our provided datasets (UCP71 and FPA_ISBSG). The service will generate 5000 records using machine learning, with the new data maintaining the statistical properties of the original data. Subsequently, the newly generated data will be used to train and evaluate the performance of our ensemble model.

7.5.4 Ensemble method

An ensemble algorithm is a technique that combines multiple machine and deep learning models, it uses these models and takes their strengths to form a new model with improved predictive power and accuracy. Ensemble algorithms can be used to solve a variety of problems e.g., classification, regression, and clustering.

The ensemble algorithm has 3 different methods: bagging boosting and stacking they will be explained in the section below.

Bagging (Bootstrap aggregating) is a method that uses a number of copies of a single model that has been trained on various subsamples of the training data. Then, a final prediction is created by combining the results of these models. By averaging the results of various models, bagging lowers the variance of the model.

Boosting is a method that uses a number of copies of a single model that has been trained on the remainder of the previous model. The final prediction is made by combining the predictions of all the models. Boosting reduces the bias of the model by fitting each model to the remainder of the previous model.

Stacking is a method that trains a meta-model based on the predictions of multiple models. This meta-model learns to combine the prediction of the base models to make a final prediction. This technique reduces bias and variance by combining the strengths of each model.

Advantages and limitations

Compared to single models, ensemble methods have several benefits, including increased accuracy and decreased overfitting. Additionally, ensemble algorithms are more resistant to data noise and outliers. Ensemble techniques can be computationally expensive and require more training data. Furthermore, compared to single models, ensemble algorithms are harder to interpret.

7.5.5 Tested model evaluation (work in progress)

	MAE	**MSE**	**PRED25**
UCP71	30.49	2607.98	100%
UCP71 enhanced	33.45	6279	100%
FPA_ISBSG	108.58	18248.43	31.82%
FPA_ISBSG enhanced			

7.6 Evaluation metrics

In this section, the evaluation metrics that are used to see how accurately a model will perform will be discussed. The metrics that are listed here will frequently occur in the next section.

7.6.1 PRED25 (Prediction accuracy)

The PRED25 serves as an accuracy indicator, the higher the PRED value, the more accurate the model is. This term refers to the percentage of projects whose anticipated values are within (25%) of their actual values. A high percentage indicates better model performance, the only exception is when a model is overfitted.

Important note: $|y_{test}|$ in the last formula represents the **length** of the y_{test} array.

$$y_{abs} = |y_{test} - y_{pred}|$$
$$y_{\leq 25} = \{y_{*25} \; in \; y_* \; : \; y_{*25} \leq 25\}$$

$$Pred25 = \left(\frac{y_{<25}}{|y_{test}|}\right) \times 100$$

Where:

- y_{test} represents the array that contains the actual values.
- y_{pred} represents the array that contains the predicted values.
- y_{abs} is the absolute difference between the actual and predicted values.
- y_* is the percentage difference between the predicted and actual values.
  - $y_{\leq 25}$ is the number of predictions where the percentage difference is less than or equal to 25%

7.6.2 MSE (Mean Squared Error)

The MSE measures the average of squared differences between the predicted and actual values over every observation in a dataset. It is a commonly used metric for regression problems, where the goal is to predict a continuous numerical value. A lower MSE value indicates better accuracy of the model.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2$$

Where:

- n is the total number of observations in the dataset.
- Y is the actual value.
- Y^ is the predicted value.

7.6.3 MAE (Mean Absolute Error)

The MAE measures the average absolute difference between the actual and predicted values of the target variable. It is a commonly used metric for regression problems, where the goal is to predict a continuous numerical value. A lower MAE value indicates better accuracy of the model.

$$MAE = \frac{1}{n} \times \sum_{i=1}^{n} \left| y - y' \right|$$

Where:

• n is the total number of observations in the dataset.

• Y is the actual value.

• Y^ is the predicted value.

7.7 Used algorithms formulations

7.7.1 Linear regression (LR)

$$y = \beta 0 + \beta 1 + \varepsilon$$

where:

• y is the dependent variable (also known as the response variable)

• x is the independent variable (also known as the predictor variable)

• β0 is the y-intercept (the value of y when x is zero)

• β1 is the slope of the line (the change in y for a unit change in x)

• ε is the error term (the part of the variation in y that is not explained by x)

7.7.2 K-Nearest Neighbors (KNN)

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Where:

• y is the dependent variable (also known as the response variable)

• x is the independent variable (also known as the predictor variable)

• β0 is the y-intercept (the value of y when x is zero)

• β1 is the slope of the line (the change in y for a unit change in x)

• ε is the error term (the part of the variation in y that is not explained by x)

7.7.3 Support Vector Machines (SVM)

$$L_{lasso}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i\hat{\beta})^2 + \lambda\sum_{j=1}^{m}|\hat{\beta_j}|.$$

Where:

- w is the weight factor.
- x is the input factor.
- b is the bias term.

7.7.4 Least Absolute Shrinkage and Selection Operator (Lasso)

$$L_{lasso}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i\hat{\beta})^2 + \lambda\sum_{j=1}^{m}|\hat{\beta_j}|.$$

7.8 Results

The characteristics of the data and the issue being solved determine which ensemble method is best. Bagging is appropriate for decreasing a model's variance, whereas boosting is appropriate for decreasing a model's bias. Stacking is appropriate when several models with various strengths are available. Computational resources and interpretability should also be taken into consideration when choosing an ensemble method.

Implementation

In this section, an explanation of how I implemented the ensemble model in my research.

Models

In this section, the best performing machine learning algorithms will be discussed. These models will be used in a stacking ensemble model and will be evaluated on Mean Absolute Error (MAE), Mean Squared Error (MSE) and accuracy (PRED25).

Linear Regression (LR) is a statistical method used to study the relationship be-tween two variables, where one variable is the predictor or independent variable, while the other variable is considered to be the response or dependent variable. The result of this model is a Mean Absolute Error (MAE) of 16.54.

K-Nearest Neighbors (KNN) is a non-parametric machine learning algorithm capable of classification and regression tasks. In KNN, the algorithm tries to predict the label or value of a new data point by looking at the k closest points in the training set.

Least Absolute Shrinkage and Selection Operator (Lasso) is a Linear Regression (LR) algorithm that is used for feature selection and regularization. The goal of lasso is to minimize the sum of squared errors between the predicted values and the actual values.

Support Vector Machines (SVM) using Random search are a type of machine learning algorithm used for classification and regression analysis. SVMs are useful when working with complex and high-dimensional datasets.

In our case we use an ensemble model using the stacking method, first of all, we train several models using Machine Learning (ML) and Deep Learning (DL) algorithms. We evaluate the performance of these algorithms on the Test set, we will use the three best performing models in our case we use the K-Nearest Neighbors (KNN), Linear Regression, and Lasso algorithm. The stacking regressor will be trained based on these models resulting in a Mean Absolute Error (MAE) of 33.01 and an accuracy of 100% considering the predicted result is within a 25 percent range of the actual value.

7.9 Gap in knowledge and possible solutions

One of gaps in knowledge is the lack of standardized approaches, there is no standardized approach to using ensemble model for Software Cost Estimation e.g., how many hidden layers should be defined in the meta model, which regression technique should your stacked ensemble model implement, and which Machine Learning (ML) and/or Deep Learning (DL) algorithms should be used for training the ensemble model.

Another gap in knowledge can be the limited of availability of data in our case we eliminated this by using synthetic data.

Complex model selection can also be an issue since choosing the model and method e.g., bagging, boosting, stacking, and optimizing the hyperparameters can be complex and time-consuming. Using grid search and random search is a good solution to counter this, but both of these techniques are very computationally expensive so this might not be the optimal solution for companies that have older or outdated specifications.

Interpretability can also be an issue as ensemble models are difficult to interpret which can make it hard to explain end results to stakeholders.

The biggest gaps in knowledge for Synthetic data are the difficulty of modeling complex and Ethical concerns around privacy and data protection, this will need to be addressed before a widespread adoption can be made.

7.9.1 Future proposal

Using synthetic data to train and evaluate an ensemble model to do Software Cost Estimation (SCE) is still a relatively new and developing field. So, my proposal is to further research the use of synthetic data and ensemble models because the initial results of this research show promising results. If more research is done the gaps of knowledge can be eliminated, resulting in a standardized method to do Software Cost Estimation (SCE) using the ensemble algorithm. Using new Deep Learning algorithms using the grid search could also be an interesting approach since using synthetic data provide enough data to accurately train a Deep Learning model (DL) model. In the field of synthetic data, the ethical concerns should be addressed so this can be adopted in fields where there is limited data available.

7.10 Conclusion

In conclusion, using synthetic data to train and evaluate a stacking ensemble model is a viable way to do Software Cost Estimation (SCE). Synthetic data can eliminate the lack of historical data and open opportunities for Deep Learning approaches, which previously has been a big issue in the field of SCE. incorporating an ensemble model using the stacking method to do SCE is also very beneficial as this method can combine predictions from multiple algorithms that have previously been proven successful into one final prediction using the strengths of all these models. I think that this methodology of doing Software Cost Estimation has proven that there is a lot of potential in the future as the current limitations such as the lack of standardized approaches in regards to ensemble models and the ethical concerns in regards to synthetic data can be solved in the future allowing for wider adaptation in the software development field. Overall, this study has shown that using synthetic data and an ensemble model can provide a promising approach to Software Cost Estimation. By addressing the limitations of traditional methods, this approach has the potential to improve accuracy and reduce costs in software development projects. Further research is needed to address current challenges and ensure ethical considerations are met, but this study demonstrates that the use of synthetic data and ensemble modeling is a promising area of future exploration in software cost estimation.

8 CONCLUSION

In conclusion, this bachelor thesis has explored and examined Software Cost Estimation in correlation with Neural Networks through a comprehensive review and original research. The review section provided a comprehensive analysis of existing literature and highlighted key themes, trends, and knowledge gaps in the field. The research section, on the other hand, aimed to address one of these gaps by using synthetic data to train and evaluate an ensemble model.

Through the review, we gained valuable insights into the working of Software Cost Estimation and the existing methodologies, due to the lack of historical data most of the methodologies used Machine Learning methods as opposed to Deep Learning approaches. These findings suggest that there are still a lot of gaps in knowledge, this gives opportunities for further research in the Software Cost Estimation field. Furthermore, the research conducted in this thesis revealed that using synthetic data and ensemble modelling. These findings contribute to the existing body of knowledge by eliminating the lack of historical data and combining previously successful methods into one combined model.

By combining the insights gained from the review and the original research, this thesis provides a comprehensive understanding of Software Cost Estimation in correlation with Neural Networks. It is evident that Software Cost Estimation is a complex and multifaceted area, with numerous opportunities for future research. Researchers can build upon the findings of this thesis by solving the gaps in knowledge such as the regularization of the ensemble modeling and the ethical concerns of synthetic data by conducting further research.

Overall, this bachelor thesis not only consolidates the existing knowledge on Software Cost Estimation but also contributes new insights through the original research conducted. The findings presented here have implications for Software Engineering companies and provide a foundation for further exploration in the field. As Software Cost Estimation continues to evolve, it is essential that researchers and practitioners utilize the knowledge gained from this thesis to advance the understanding and address the challenges in this domain.

9 LITERATURE CITED

1. https://en.wikipedia.org/wiki/Tomas_Bata_University_in_Zlín
2. Ramaekers, R (2023) 'Software Cost Estimation using neural networks'. Unpublished paper, Tomas Bata University Zlin, Czech Republic
3. Monday.com (2022) 'How to use analogous estimating for better budgeting'. Monday.com Analogous estimating
4. Waida, M (2022) 'Bottom-Up Estimating in Project Management: A Guide'.Wrike.com Bottom-Up estimating
5. Guthrie, G 'How 3-point estimating can improve project planning and resource allocation'. https://nulab.com/learn/project-management/3-point-estimating/
6. Construction Site Management 'What are the advantages and disadvantages of using AI and machine learning for construction cost estimation' Linkedin.com Advantages and disadvantages of AI and machine learning.
7. Kumar, K.H., Srinivas, K (2023) 'An accurate analogy-based software effort estimation using hybrid optimization and machine learning techniques.' https://link.springer.com/article/10.1007/s11042-023-14522-x
8. Duggal H, Singh P (2012) 'Comparative Study of the Performance of M5-Rules Algorithm with Different Algorithms.' https://www.scirp.org/journal/paperinforcitation.aspx?paperid=18573
9. Sharma, S., Vijayvargiya, S (2022) 'Modeling of software project effort estimation: a comparative performance evaluation of optimized soft computing-based methods.' https://doi.org/10.1007/s41870-022-00962-5
10. Ramya, P., Sai Mokshith, M., Abdul Rahman, M., Nithin Sai, N. (2023). 'Software Development Estimation Cost Using ANN.' https://doi.org/10.1007/978-981-19-6880-8_23
11. Sudhir S & Shripal V (2022) 'An Optimized Neuro-Fuzzy Network for Software Project Effort Estimation' https://doi.org/10.1080/03772063.2022.2027282
12. K Nitalaksheswara R, Jhansi Vazram B, Satyanarayana M, CH. V. Murali K, O. Gandhi, M James S(2022) 'OLCE: Optimized Learning-based Cost Estimation for Global Software Projects' https://doi.org/10.21203/rs.3.rs-2024296/v1
13. Yingzhou L, Huazheng W, Wenqi W (2023) 'Machine Learning for Synthetic Data Generation: A Review' https://doi.org/10.48550/arXiv.2302.04062
14. Marta L, Alessia P, Vanessa O, Melissa F, Maurizio M (2023) 'Characterization of Synthetic Health Data Using Rule-Based Artificial Intelligence Models' https://doi.org/10.1109/jbhi.2023.3236722
15. Necmi G, Sadullah Ç, Esma B (2023) 'An Introduction to Synthetic Data' https://doi.org/10.1007/978-1-4842-8587-9_1
16. Jin Gang L, Hyun-Soo L, Moonseo P and JoonOh S (2021) 'Early-stage cost estimation model for power generation project with limited historical data' 10.1108/ECAM-04-2020-0261
17. Priya Varshini A, G, Anitha, Kumari K, Varadarajan, and Vijayakumar (2021) 'Estimating Software Development Efforts Using a Random Forest-Based Stacked Ensemble Approach' https://doi.org/10.3390/electronics10101195