Software Cost Estimation using Synthetic data and ensemble modelling

Robin Ramaekers[1][0000-1111-2222-3333]

[1] Thomas More Geel, Kleinhoefstraat 4, 2440 Geel, Belgium
[1] FAI - Univerzita Tomáše Bati, Nad Stráněmi 4511, 760 05 Zlín, Czech Republic
R0832585@student.thomasmore.be

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 Neighbours) 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. Using the fpa_ISBSG dataset enhanced with synthetic data, the study outcomes show that using the ensemble model to combine different Machine Learning and Deep Learning algorithms (1,2,3) results in a Mean Absolute Error (MAE) of … and an accuracy score of … 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.

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 exper-

tise, 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 Neighbours, 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 standardisation regarding the use of ensemble models trained on synthetic data.

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 Neighbours (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 & 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) developed a model capable of understanding complex relationships and patterns found in the data sources for Software Cost Estimation.

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 1, 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.

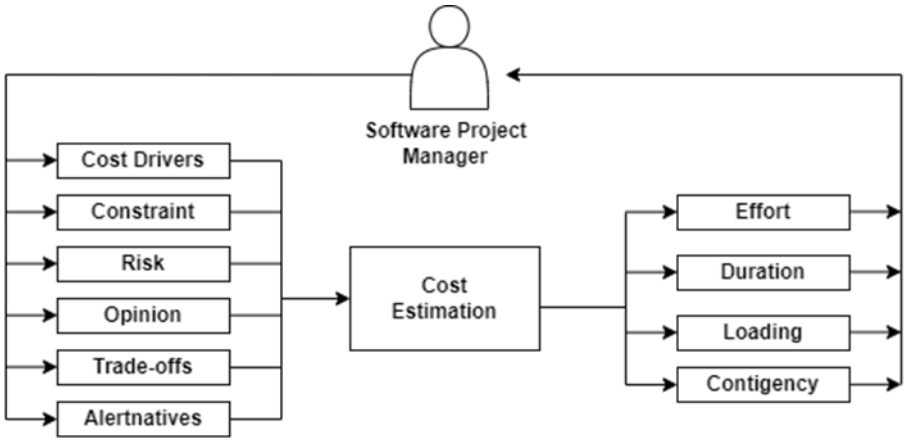

Figure 1: The typical process of Software Cost Estimation [1]

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 [1].

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. [1]
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. [2]

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. [3]

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. [4]

4 Methods used

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. [5]

- 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. [5]

- 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. [5]

- Consistency: AI algorithms can provide consistent estimates based on predefined rules and parameters, which can reduce the variability that may occur when humans estimate costs. [5]

- 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. [5]

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.

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.

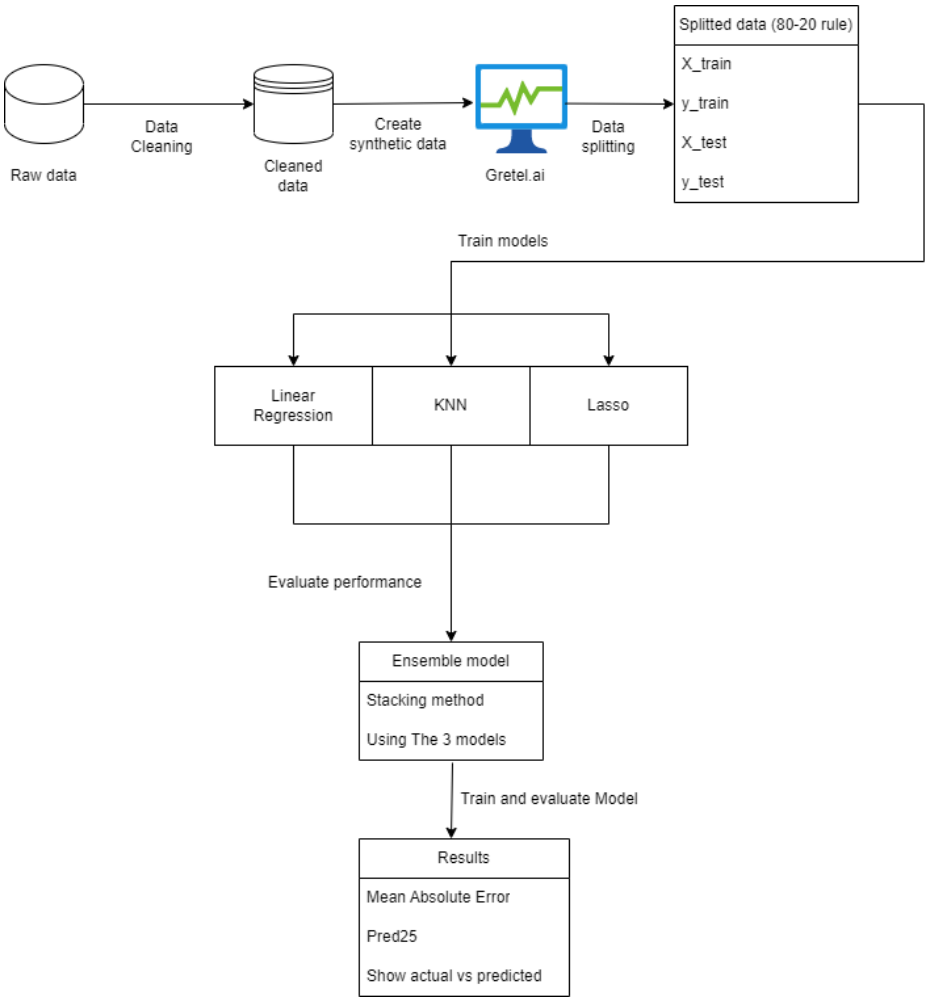

Figure 2: The whole process visualized.

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.

The statistical properties of these columns are as follows:

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

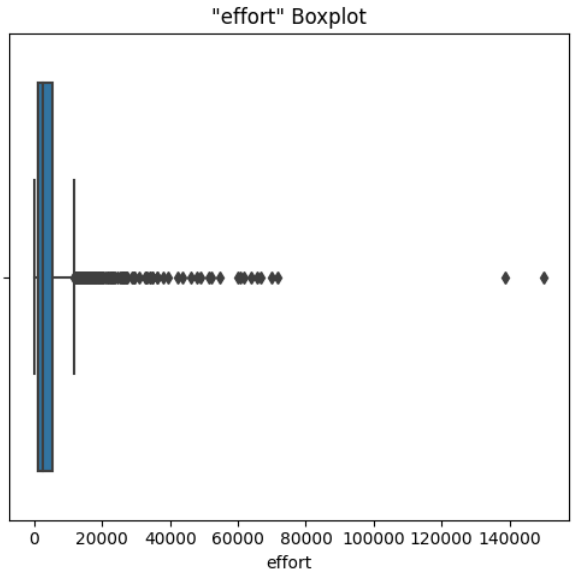

Figure x: Effort boxplot

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.

The statistical properties of these columns are as follows:

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

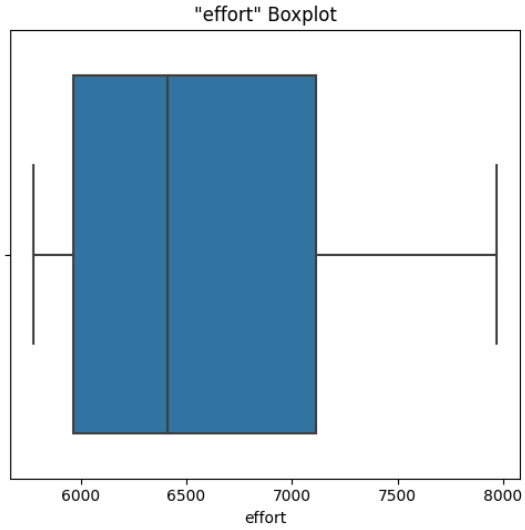

Figure x: Effort boxplot

Each of these columns provides crucial information that assists in the accurate prediction of the effort required to execute a software project successfully.

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.

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. 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.

5.4 Ensemble model

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.

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			

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.

6.1 PRED25 (Prediction accuracy)

The PRED25 serves as an accuracy indicator, the higher the PRED value, the more accurate the model is. **T**his term refers to **t**he 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.

$$y_{abs} = |y_{test} - y_{pred}|$$
$$y_{*} = \left(\frac{y_{abs}}{y_{test}}\right) \times 100$$
$$y_{\leq 25} = \{y_{*25} \; in \; y_{*} \; : \; y_{*25} \leq 25\}$$
$$Pred25 = \left(\frac{y_{<25}}{|y_{test}|}\right) \times 100$$

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%

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.

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 Used algorithms formulations

In this section, the formulations of the used algorithms will be explained. These formulations will be referenced in section 8.3 Used methodologies, implementation, and models.

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)

7.2 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)

7.3 Least Absolute Shrinkage and Selection Operator (Lasso) !!!!

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

Where:

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

7.4 Support Vector Machines (SVM)

$$f(x) = w \times x + b$$

Where:

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

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 between 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.

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, and 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 model 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.

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 further to research the use of synthetic data and ensemble models because the initial results of this research show promising results. If more research is done, knowledge gaps 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.

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 need for more 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.

11 Literature cited

1. Ramaekers, R (2023) 'Software Cost Estimation using neural networks'. Unpublished paper, Tomas Bata University Zlin, Czech Republic
2. Monday.com (2022) 'How to use analogous estimating for better budgeting'. Monday.com Analogous estimating
3. Waida, M (2022) 'Bottom-Up Estimating in Project Management: A Guide'. Wrike.com Bottom-Up estimating
4. Guthrie, G 'How 3-point estimating can improve project planning and resource allocation'. https://nulab.com/learn/project-management/3-point-estimating/
5. 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.