Software Cost Estimation using Synthetic data and ensemble modeling

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, 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. 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 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. This paper is divided into 10 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, 11 Literature cited. The use of Artificial Intelligence in Soft-ware 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.

2 Related works

Several studies have investigated the use of synthetic data. For example, Trivellore E. Raghunathan (2020) has made an in-depth review stating the importance of data and how synthetic data can make data more accessible for researchers. Meanwhile, Yingzhou Lu, Huazheng Wang, Wenqi Wei (2023) have investigated the use of machine learning for synthetic data, they concluded that synthetic data has a promising future in Artificial Intelligence. M Maher and JS Alneamy (2022) have researched the use of a stacked ensemble model in Software Cost Estimation using KNN, SVR, ANN, Bayesian Regressor, Leaner Regressor, and Random Forest resulting in a MMRE of 0.14. 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.

3 Software Cost Estimation (SCE)

As stated in Section 1, the significance of Software Cost Estimation (SCE) in a new software development project cannot be understated. The main objective 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 started. 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 below, Figure 1, the typical process for doing Software Cost Estimation (SCE) is shown. In this cycling process, the various attributes are altered until the cost of the target software is found justified. The other development teams can give their input or opinion but note that this process is carried out by the Software Project Manager who is responsible for making the final decision.

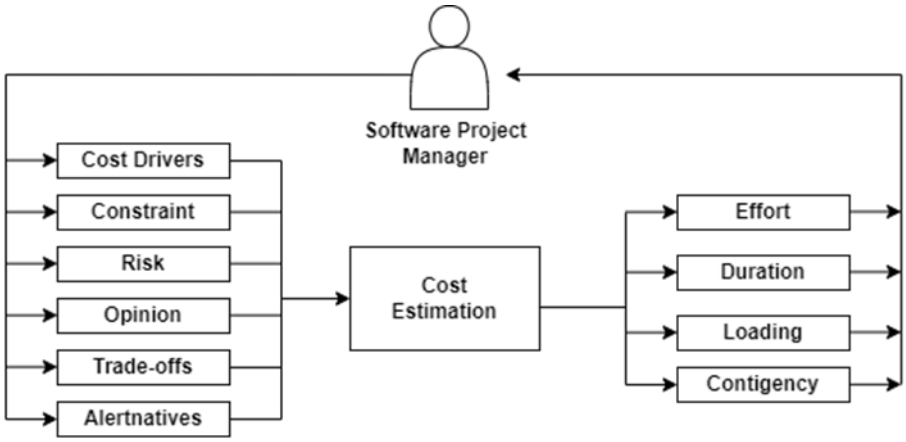

Figure 1: The typical process of Software Cost Estimation [4]

Figure 1 illustrates a crucial practice for analyzing and predicting potential risks and trade-offs in the software costing process. By using this process, software project managers can evaluate the project and identify and address potential risks to achieve the target cost. It's important to note that the project manager and department teams

collaborate to determine the final cost. Cost estimation involves various input variables, including cost drivers, constraints, risks, opinions, trade-offs, and alternatives. Meanwhile, the estimation's outcomes are assessed based on effort, duration, loading, and contingency. In the following section, we'll delve into different taxonomies and challenges related to software cost estimation. [4]

3.1 Taxonomies

Currently, there are numerous models available for Software Cost Estimation, including COCOMO (Constructive Cost Model), COCOMO II, Function Point Analysis (FPA), PERT (Program Evaluation and Review Technique), and others. Each model has its own strengths and weaknesses, and the choice of which model to use depends on the specific requirements, constraints, and scope of the software project. More detailed information on the advantages and disadvantages of each taxonomy is provided in the following section. Out of all the models, COCOMO is the most commonly used model in the software industry due to its simplicity, flexibility, availability of historical data, and broad applicability. Additionally, there are standard taxonomy cost prediction methods available.

- Analogous Estimating: Analogous Estimating, also known as top-down estimating, is a project management technique that involves estimating the cost or duration of a current project by comparing it to a similar completed project. This approach relies on data from a previous project to estimate the cost of the current one and is particularly useful when information about the current project is limited.

- Bottom-Up Estimating: Bottom-Up Estimating is a project management technique that involves dividing a project into smaller components and estimating the cost or duration of each component to determine the total cost or duration of the project. This approach is considered more precise than Analogous Estimating, but it requires more time and effort to implement.

- 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.

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

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:

Analogous Estimating

- Finding a suitable reference project: The accuracy of Analogous Estimating relies on finding a reference project that closely matches the current project. However, finding a suitable reference project can be a difficult task, especially when dealing with unique or intricate projects.

- Historical data: Analogous Estimating is based on using historical data from a completed project to forecast the cost of the current project. However, there may be insufficient historical data available, which could pose challenges in achieving an accurate cost estimation for the current project.

- Project details: Analogous Estimating assumes that the specifics of the current project are similar to the reference project utilized for estimation. Nevertheless, if there are significant variations in the specifics of the two projects, the estimate obtained may not be accurate.

- Assumptions and biases: Analogous Estimating involves making assumptions about the current project based on the reference project utilized for estimation. These assumptions can be susceptible to biases, like optimism bias, that may result in imprecise estimates.

- Lack of transparency: Analogous Estimating may lack transparency compared to other cost estimation techniques since it heavily depends on the proficiency of the estimator to generate precise estimates. Consequently, stakeholders may face difficulties in understanding and validating the estimates.

Bottom-Up Estimating

- Time-consuming: Bottom-Up Estimating may pose a challenge in large and complex software development projects as it involves dividing the project into smaller tasks and estimating the cost of each task individually. This meticulous process can be time-consuming and demand a significant amount of effort to produce precise estimates.

- Lack of expertise: Bottom-Up Estimating requires a good 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 commonly face changing requirements, which can complicate accurate cost estimation. The evolving requirements may require the revision of initial estimates, leading to additional time and costs for the project.

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

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

Three-Point Estimating

- Subjectivity: Three-Point Estimating technique relies on the evaluator's discretion to determine the optimistic and pessimistic estimates of a task. This subjectivity may result in imprecise estimates if the estimator is biased or lacks the required expertise.

- Overconfidence: Project managers 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.

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, ISBSG, China, etc. datasets. These datasets focus on specific fields of Software Cost Estimation such as Functional Points (FPA), Lines Of Code (LOC), and Use Case Point (UCP) analysis. In my research, I decided to work with Functional Point Analysis (FPA) and Use-Case point (UCP) reason being that both of these have their strengths and weaknesses e.g., standardization, accuracy, flexibility, and improved communication.

4.1 Used Software Cost Estimation (SCE) data sources

In this section, the choice of our used data sources is clarified.

Functional Point Analysis (FPA)
One of the most used Functional Point Analysis data sources is the ISBG data source. This dataset contains a lot of detailed information about historic projects. I will be using this data source because of its reliability and in-depth information about a lot of historical projects. A more in-depth explanation of this data source will be in the next section.

Use Case point Analysis (UCP)
For training my model on a UCP dataset I will be using the UCP71 data source. This dataset is frequently used to evaluate the performance of UCP estimation models. A more in-depth explanation of this data source will be in the next section.

4.2 EDA Data sources

A description of the mentioned data sources is provided in this section. It is crucial to comprehend the data in a dataset before creating an Artificial Intelligence (AI) model. This makes it possible to do an in-depth Exploratory Data Analysis (EDA) of the provided data.

FPA_ISBSG
The FPA_ISBSG is a Functional Point Analysis (FPA) data source used for Software Cost Estimation. It contains 1712 records and 58 columns from various industry sectors. This data was collected and compiled by the International Software Benchmarking Standards Group (ISBSG). We will use 7 of these columns (EI, EO, EQ, ILF, EIF, Size, xIndustrySectorId) to predict the effort that is required to complete a project.

Table 1: The table below will explain the meaning of the "effort multipliers."

EI	This column indicates the **External Inputs.**
EO	This column indicates the **External Outputs.**
EQ	This column indicates the **External Inquiries.**
ILF	This column indicates the **Internal Logical Files.**
EIF	This column indicates the **External Interface Files.**
xIndustrySectorId	This column indicates the **Industry Sector.**
Size	This column indicates the **size of a software project.**
effort	This column indicates the **effort required to complete a software project.**

UCP71
The UCP71 is a Use-Case Point (UCP) data source used for Software Cost Estimation. It contains 71 records and 29 columns from various industry sectors. This data was collected University of Central Florida (UCF). We will use 6 of these columns (UAW, UUCW, TCF, ECF, size) to predict the effort that is required to complete a project.

Table 2: The table below will explain the meaning of the "effort multipliers."

UAW	This column indicates the **Unadjusted Actor Weight.**
UUCW	This column indicates the **Unadjusted Use Case Weight.**
TCF	This column indicates the **Technical Complexity Factor.**
ECF	This column indicates the **Environmental Complexity Factor.**
size	This column indicates the **size of the software project.**
effort	This column indicates the **effort required to complete a software project.**

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.

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.

5.2 Advantages

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.

- 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 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 Used methodologies

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

8.1 Distribution of data

In our proposed methodology we will be splitting 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.

8.2 Synthetic data

Synthetic data is defined as data that has been artificially created. It mimics the traits and qualities of actual data. Synthetic data can be produced using statistical models, machine learning algorithms, or other data generation techniques and is utilized for many things e.g., training models and testing algorithms. It is made to combat a lack of historical data.

Generation techniques
In this section, the most used data generation techniques will be explaine**d.**

Rule-based generation is A technique that involves using pre-defined rules. The rules are based on the structure and properties of the original data, they can also be based on domain knowledge. Rule-based generation is useful if you are certain of the structure and characteristics of the data you wish to generate. Additionally, you can use it to generate data that adheres to a particular pattern or distribution.

Model-based generation is a technique that uses Machine-Learning algorithms to generate data that simulates the properties of the original data. The algorithms are trained on the original data and can detect relationships and patterns to generate synthetic data that is similar in structure.

Monte Carlo simulation is a technique that involves using statistical models and simulations to generate synthetic data. The Monte Carlo simulation generates random samples using a probability distribution that is like the original data. It uses these samples to create synthetic data.

Advantages and limitations
A few advantages of synthetic data are its capability to decrease data scarcity and the ease of data sharing. Synthetic data does have certain limitations, including the potential for bias introduction, the difficulty in fully capturing complicated relationships in the data, and the lack of diversity in the generated data.

Implementation
In our case, we use Gretel.ai to generate synthetic data using the model-based generation technique. The generated data will be based on our provided datasets (UCP71 and FPA_ISBSG). The service will generate 5000 records using machine learning, the new data will maintain the statical properties of the original data. After, the newly generated data will be used to train and evaluate the performance of our ensemble model.

8.3 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.

Methods
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 a number of benefits, including increased accuracy and decreased overfitting. Additionally, ensemble algorithms are more resistant to data noise and outliers. Ensemble techniques can, however, be computationally expensive and necessitate more training data. Furthermore, compared to single models, ensemble algorithms are harder to interpret.

Choosing the appropriate method
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.

Result

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.

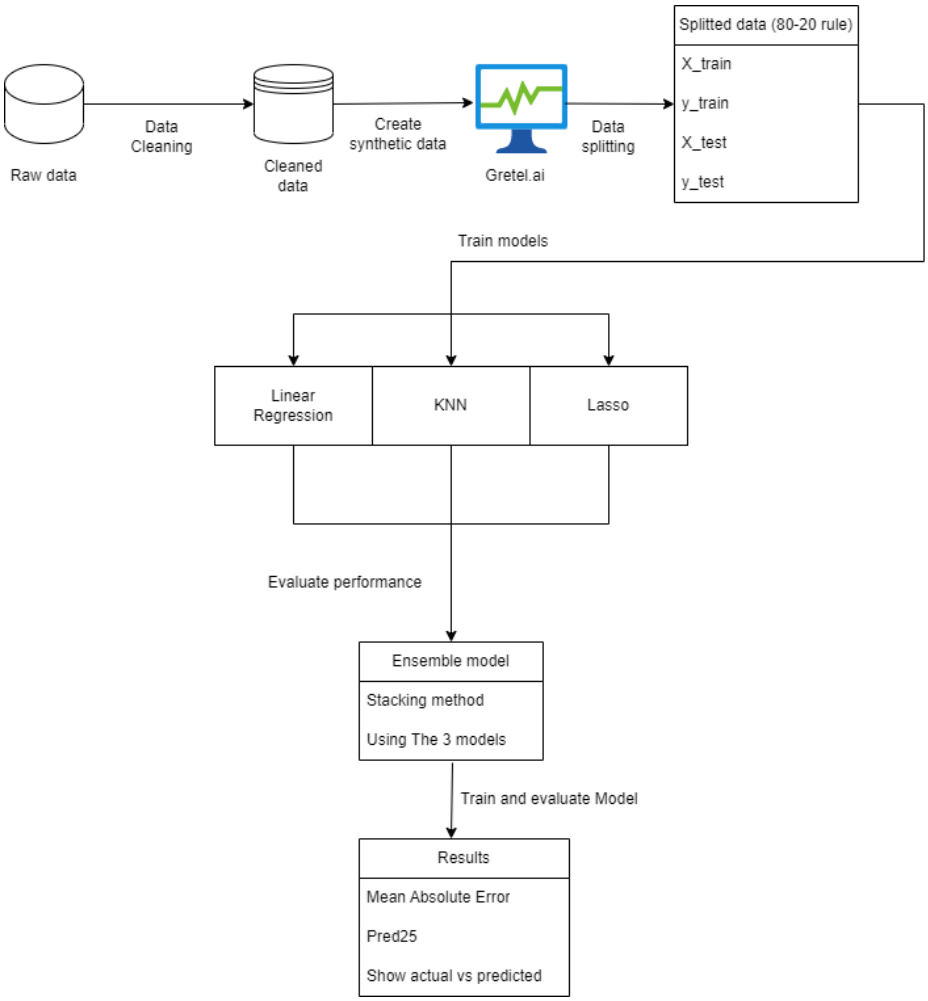

Figure 2: The whole process visualized.

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 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.

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.

11 Literature cited

Later nog invoegen wanneer ik de link heb van mijn paper.