CHEAP - Cost Estimation Prediction Models Unveiled

Antonio La Marca
a.lamarca24@studenti.unisa.it
Matricola: 0522501557

Daniele Donia
d.donia@studenti.unisa.it
Matricola: 0522501575

Pasquale Somma
p.somma11@studenti.unisa.it
Matricola: 0522501543

Abstract

Software cost estimation provides the foundation for planning the job scheduling and budget; the accuracy of an estimation can determine if the project is able to meet its objectives. However, an effective cost estimation requires a thorough understanding of the project scope, the deliverables and the effort needed to develop the project, so it is very complex. For this reason, our main goal is to develop a machine learning model to estimate software project development effort in order to help project managers in their job. In particular, we intend to use a specific Ensembling method, that is Stacking, to improve the generalization and the accuracy of the predictions and mitigate the small size of the datasets related to the issue at hand. Moreover, the designed model should be able to adapt to new data, in order to provide project managers with an accurate and reliable cost prediction system that help them plan development activities efficiently. A comprehensive analysis of the existing cost estimation models is finally carried out with the purpose of assessing how competitive the proposed model could be compared to state-of-art technologies.
To access the Github repository, click on this link: **Repository**.

1 Context of the project

During the initial phases of software development process, software cost estimation is one of the most important and critical activity to perform. An accurate estimation can help the project manager define an appropriate allocation of the resources. It also makes the difference diminishing the possible clashes between collaborators in the later stages of the project. Accurate cost estimation for software projects can bring a variety of benefits to both a company building software and an organization acquiring software.
On the contrary, a project, whose cost is wrongly evaluated, might encounter several problems, such as under-employment, underscale quality control measures, and, in some cases, fail.
For this reason, throughout the years many researchers have tried to develop software cost estimation models in order to improve the estimation process and obtain more dependable results. These models can be classified on the basis of their formulation patterns:

- Analogy-based estimation
- Expert judgement method
- Algorithmic models containing empirical methods
- Non-algorithmic models centered on soft computing

Constructive Cost Model (COCOMO) [6], is the most widely used algorithmic technique in software project cost estimation. It is a hierarchy of software cost estimation models and it defines mathematical equations to evaluate effort, development time and the maintenance effort.
Nevertheless, a lack in terms of effectiveness and robustness in conventional approaches, such as COCOMO, has been pointed out in several studies [18, 19]. In particular, Kaushik et al. [9] underlined the difficulties in modelling complex relationships between the contributing factors and operating with categorical data.
Owing to the limitations that characterize algorithmic models, researchers moved towards the study and design of non-algorithmic models [17]. In particular, those based on machine learning, as stated by Abdulmajeed et al. [2], appeared to be the right choice for cost prediction because of their ability to learn from prior gathered project data.

2 Goals of the project

Our goal is to develop a machine learning model that is intended as a useful tool for companies to avoid the waste of the sources and the failure of their software projects.

We strive for delivering project managers with an accurate and reliable cost prediction system that is able to help them plan development activities efficiently. For this reason, we define the following research question:

> **RQ$_1$.** *To what extent can the proposed model estimate the effort required to develop a software project?*

To achieve this goal, we also intend to understand which characteristics of the software and its development process mainly affect the required effort. Therefore, we seek to provide a response to the following research question:

> **RQ$_2$.** *What features of the selected dataset are appropriate to improve the prediction accuracy of the proposed model?*

Apart from the accuracy of the predictions made by our model, investigated through the previous research questions, it is interesting to understand how competitive the proposed model is in the Software Development Effort Estimation scenario. Specifically, we conceive the third research question as follows:

> **RQ$_3$.** *Up to what degree can our machine learning model be accurate in predicting software development effort comparing to state-of-art technologies?*

We also intend to adopt a "real-time up-to-date" strategy, allowing the model to adapt to new data collected from the users' interaction with the system. This approach could enable the model to continuously update and provide more precise predictions over time since it has access to the most current and relevant information (instead of relying solely on a fixed dataset). Thus, our last research question is:

> **RQ$_4$.** *To what extent does the re-training of the operating model using users' feedback affect the accuracy of the estimation?*

3 Model construction

The small size of all the identified datasets made it necessary to look for techniques to increase the accuracy of our approach.
The choice fell on Ensembling, which allows to train different machine learning models, orthogonal to each other, with the purpose of reducing variance and increasing generalization [10].
In this study, we implemented two specific ensembling techniques: stacking and blending. Stacking involves combining predictions obtained from various models into a single prediction using another model; blending [21], on the other hand, is a variant of stacking. Given the characteristics of the chosen dataset, analyzed in Section 4.3, we have decided to combine four models within the ensemble.

- Support Vector Regressor is a powerful model known for its ability to handle non-linear data and complex relationships.
- Elastic Net gives a contribute to the ensemble by capturing the few linear relationships within the dataset.
- Adaptive Boosting is a popular ensemble learning algorithm, capable of handle unbalanced datasets or complex relationships between features and the target variable.
- Random Forest is also an ensemble learner that is robust to outliers and missing data and can capture nonlinear relationships between features and the target variable.

In fact, using different base learners with diverse strengths and capabilities, Ensembling can effectively capture various types of patterns and relationships present in the data.

The meta-model that utilizes the predictions of these various models is K-Nearest Neighbors (KNN). It is a non-parametric model that excels at capturing non-linear relationships by considering the similarity of neighboring data points. It does not assume any specific distribution, allowing it to handle non-normal data effectively. This choice was motivated by its simplicity and computational efficiency, obtaining a balance between model complexity and computational resources.

Additionally, two distinct approaches were employed to fine-tune the hyperparameters of the model with the purpose of enhancing the accuracy of the predictions. The initial technique involved the utilization of a Particle Swarm Optimization algorithm, which has shown promising outcomes in tuning ensemble models for software effort estimation, as observed by Palaniswamy and Venkatesan [15]. Furthermore, Optuna [3], a hyperparameter tuning framework, was utilized to implement a Bayesian Optimization algorithm.
By employing these two approaches, a comparative analysis was conducted to determine the optimal configuration for maximizing the model's performance.

The model was activated and made operational through a Slack bot, called CHEAP (Cost-effective Heterogenous Ensemble for Accurate Predictions), since it is one of the most widely used and useful platforms for communicating within a team.
In particular, during the early stages of a project's life, the bot, on the basis of information gathered, is intended to forecast the effort needed to implement the project, so that the project manager can make timely changes.

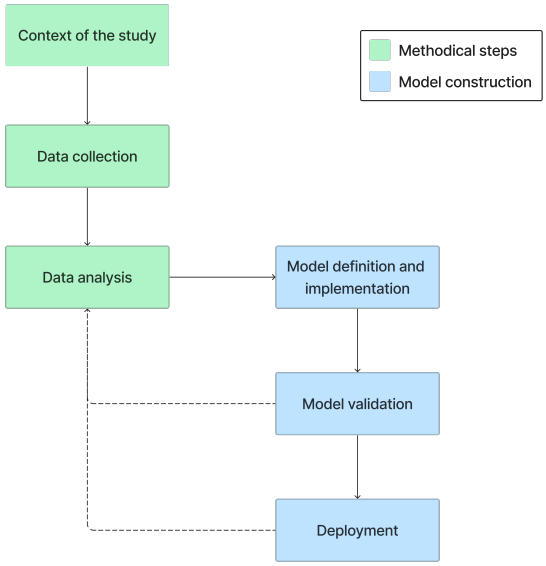

Figure 1: Methodical steps.

4 Methodical steps to address the goals

4.1 Context of the study

Primarily, an analysis of the state of art regarding software cost estimation was conducted. This analysis included the investigation of previous studies and the datasets employed in order to have a complete view of what has already been achieved by the currently deployed technologies. For this reason, we identified some interesting datasets concerning software project effort estimation and selected those who describe the size of the project in terms of function point and object point approach. In particular we focused on the following ones: Maxwell [12], Desharnais [7], ISBSG [8], SEERA [13]. These datasets are considered suitable for the issue at hand given that they capture the information about the complexity of projects, the development environment and the required effort. Each dataset is described in more detail below.

4.1.1 Maxwell dataset. It contains 62 projects dated between 1985 and 1993 and collected by a Finnish bank. Each project is described by 27 numerical attributes, which describes the development environment and project requirements. Moreover, there are no missing values for any record.

4.1.2 Desharnais dataset. It is a PROMISE Software Engineering Repository dataset and consists of 81 projects collected from Canadian software houses between 1982 e 1988. It includes 10 attributes, mainly related to the project size and staff experience.
Unfortunately, 4 projects contain missing values.

4.1.3 ISBSG dataset (13^{th} release). It is formed through the collection and analysis of more than 6000 software project data from organizations around the world, between 2000 and 2013. The data is categorized in 129 attributes including information about the

project and the development type, schedule of the task, architectures, hardware and techniques. Unfortunately, most of the projections have missing values in many fields.

4.1.4 SEERA dataset. It is used for research in software cost estimation and it consists of data collected from 120 software projects developed by 42 different private and public organizations in Sudan, between 1993 and 2019. It includes 76 project attributes concerning the information about the organization, the development environment, the users, the developers (team size, experience of the team, etc), the project (programming language, application domain, functional requirements, etc) and other factors that could influence the development effort. It presents missing data in 44 attributes.

An important difference between the datasets is that the SEERA dataset includes Object Points (OP) as a software metric, while the Maxwell, ISBSG and Desharnais datasets include Function Points (FP) as a metric. The main difference between FP and OP lies in the fact that the former are based on the evaluation of the functionalities provided by the system, while the latter are based on the evaluation of functional objects. In both cases, the result of the evaluation is a number of points, which represents the size of the software in terms of functionalities provided or functional objects provided.

4.2 Data collection

For our experiments, we intended to use datasets that had already been selected and analyzed by researchers for previous studies concerning software development effort estimation.
The **Maxwell dataset** was created by Dr. Margaret Hamilton at the Software Engineering Laboratory at NASA and is available from the open-access digital repository Zenodo.
The **Desharnais dataset** was created by Alain Desharnais and is available from the Kaggle platform.
The **SEERA dataset** is maintained by the SEERA project team at the University of Oulu and is also available from Zenodo.
The **ISBSG dataset** is a set of projects collected by the ISBSG (International Software Benchmarking Standards Group) from many industries and many business areas.

Beyond these datasets, the model is periodically trained after its deployment using Slack users' feedbacks. Through the interaction with the bot, users have the opportunity to report any inaccuracies in the predictions and the actual effort required to complete projects, on which the model is re-trained.

4.3 Data analysis

After the data collection, we planned to perform a statistical analysis of the selected datasets. Bardsiri et al. [5] already provided an accurate analysis of the Maxwell, ISBSG and Deshernais datasets, focusing on descriptive statistics, correlation coefficients, data distribution, data visualization and regression analysis.
Indeed, we only extended the analysis to the SEERA dataset.
However, Shepperd et al. [20] raised important questions about the quality of the datasets made available by the Promise data repository. Specifically, the researcher put the attention on data consistency and inconsistencies, stating that these problems could have a "far reaching impact upon the final results and conclusions" of the studies that use these datasets. Shortcomings regarding the inconsistency of Deshernais dataset, along with the ISBSG dataset, were also pointed out by Mustafa and Osman [13]. They also expressed doubts about the trustworthiness of the Maxwell dataset, which has also a limited number of records and attributes and a narrow domain.
For this reason, we decided to focus our study on the SEERA dataset. It comprises a collection of the most recent projects and the metadata related to them, encompassing a diverse range of domains (as mentioned in Section 4.1). The inclusion of such detailed information could enable a thorough analysis of the projects and facilitate insights into the development process.

As mentioned earlier, the selected dataset presents different features about projects. In order to answer $\mathbf{RQ}_2$, we needed to determine the relevant ones for the prediction; this process was based on the state-of-art analysis and the comprehension of the cost estimation domain, included the application of bio-inspired algorithms, which often allow to improve the accuracy of the predictions [4]. Specifically, we excluded attributes that are not available or measurable in the initial phases of the development process and the ones that are highly correlated with each other, measured using Pearson's correlation; subsequently, we applied a genetic algorithm to select the optimal features for the prediction model.

An initial analysis of the dataset allowed us to identify the presence of non-linear data and a non-normal distribution. This information was used to set up the subsequent data cleaning phase. To ensure data consistency and reliability, we implemented data preprocessing techniques as part of our methodology.

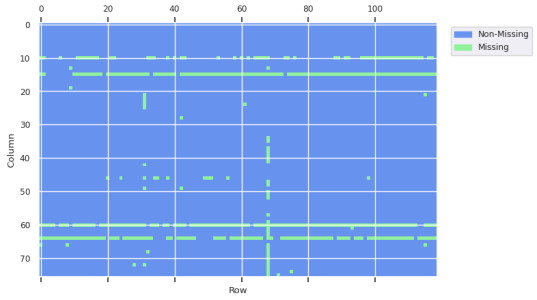

Figure 2: Map of null values in the SEERA dataset

One essential step involved addressing data formatting, which included ensuring compliance with the decimal separator standard and handling missing values represented by both the symbol '?' and NaN (Not a Number) value. To handle the latter issue, we adopted a standardized approach by replacing all occurrences of "?" with NaN values. This allowed us to treat missing values uniformly throughout the dataset and facilitated subsequent data analysis.
Additionally, we conducted a thorough evaluation of the dataset to identify the number of NaN values present in each row. This assessment provided valuable insights into the quality and completeness of the data. Rows with a significant number of missing values were identified and subsequently excluded from our analysis.
The remaining NaN values were imputed using a K-Nearest Neighbors, as it is particularly useful with non-linear data.
We then removed the most relevant outliers using an Isolation Forest algorithm. In the end, the feature 'User Manual', which indicated

certain features of the projects' user manual, was transformed to simply indicate whether or not a user manual was intended.
We did not apply any scaling to the data because the values of the selected features are all very small and there is not much difference between them. Furthermore, for many of the selected models in the Section 3, scaling is not necessary. We just scaled the input data for the meta model, using min-max scaling, because k-Nearest Neighbors relies on distance calculation and scaled data can help get more accurate results and converge sooner.

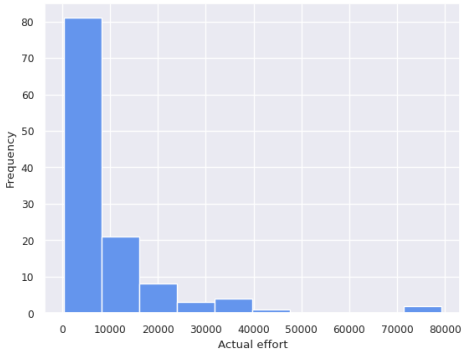

Figure 3: Distribution of values for 'Actual effort'

The scarcity of projects with high effort in the dataset was addressed by applying the SMOGN oversampling technique [11], with the purpose of enhancing the prediction performance for such projects. Once we chose and cleaned the SEERA dataset to train the designed model, we evaluated to what extent it was able to estimate the effort required to develop a project with the purpose of addressing $\mathbf{RQ}_1$. In particular, we adopted the 8-fold cross-validation technique to assess the quality of the predictions and we evaluated the model using the following metrics:

- Mean Magnitude of Relative Error (MMRE), measures the average magnitude of the relative differences between the predicted values and the true values.
- Root Mean Square Error (RMSE), is the square root of the mean of the square of all of the error (difference between predicted and actual values). It is commonly used in regression problems and for identifying the features that contribute the most to the model's error.
- R-Squared (R^2), measures the proportion of variance in the dependent variable that is explained by the independent variables in a regression model. It is a widely used metric in regression analysis and can be helpful in identifying the features that have the most impact on the dependent variable.

All these metrics can be helpful in comparing the performance of different models and to evaluate the accuracy of machine learning models. These metrics were also taken into account to perform the analysis on the re-trained-model, necessary to answer $\mathbf{RQ}_4$.

Finally, we compared the results achieved by the model with those of other software cost estimation approaches in the literature ($\mathbf{RQ}_3$), evaluated using the same metrics. Unfortunately, there was no approach applied to the SEERA dataset, thus we had to rebuild and evaluate them using it. In particular, we opted for the MultiLayer Perceptron method introduced by Nassif et al. [14] and various ensemble approaches, such as voting and bagging [1].
The results of the evaluation and the comparative analysis are shown in Section 5.

5 Preliminary results and findings

Regarding the solution proposed in Section 3, a fundamental step for increasing its performances was to determine which features were the most relevant in effort prediction ($\mathbf{RQ}_2$).
In particular, the genetic algorithm employed for this task selected those related to the developer and team attributes, along with the technologies used. Surprisingly, sizing methods, such as object points, were not considered useful for the prediction. The complete list of selected features is available in the Github **repository**.

Based on the analysis of model performance using the SEERA dataset, we assessed to what extent the proposed model is able to estimate the effort required to develop a software project ($\mathbf{RQ}_1$).
At first, we compared the results yielded by the stacking and blending approaches, which are shown in Table 1.

Model	***MMRE***	***RMSE***	***R2***
Random Forest	0.85	6122.12	0.53
AdaBoost Regressor	1.17	6640.40	0.42
Elastic Net	1.23	6643.71	0.39
Support Vector Regression	1.06	5614.98	0.55
Stacking	**0.60**	**5568.85**	**0.63**
Blending	0.85	7397.35	0.40

Table 1: Performance metrics for ensemble and weak learners

The results presented in Table 1 demonstrate that the stacking method offers a more accurate and reliable prediction model for software effort estimation. This conclusion is supported by the superior performance of the stacking model compared to the blending approach and the constituent weak learners in the ensemble.

Model	***MMRE***	***RMSE***	***R2***
MLP	0.66	8122.66	0.24
Voting	0.94	7566.75	0.47
Bagging	1.81	9598.97	0.18
Stacking	**0.60**	**5568.85**	**0.63**

Table 2: Performance metrics for ensemble and other models

The proposed model was lastly compared with a selection of state-of-art technologies in software effort estimation prediction ($\mathbf{RQ}_3$). Table 2 shows an overview of the comparison.

Stacking demonstrated remarkable competitiveness with the state-of-the-art technologies analyzed on SEERA, considering the evaluated metrics. Among them, MLP was the only solution that approached our results in terms of MMRE. However, the R^2 value associated with MLP indicates poor data fitting for the model.

Despite the fact that our model achieved the best results in the various comparisons, the overall outcomes fall short of satisfaction,

as the MMRE should ideally hover around 0.25 to exhibit substantial predictive capability, whereas our model exhibited an MMRE of 0.60. Furthermore, the desired R^2 value is close to 1, but our model obtained a value of 0.63.

However, the model exhibited a modest indication of improvement due to the inclusion of retraining data provided by users ($\mathbf{RQ}_4$). Nonetheless, it is important to acknowledge that the available data was inadequate for a definitive validation of this hypothesis.
Intensive use of the model by users over time would be necessary.

6 Implications of the results

The findings in our study carry multiple implications. Firstly, the ensemble approach demonstrated remarkable competitiveness, surpassing both the weak learners and state-of-the-art technologies analyzed on the SEERA dataset. Indeed, it confirmed our intention to improve generalization and address the limitations of a small dataset. However, the overall outcomes were unsatisfactory as the proposed solution failed to meet performance standards.

These findings suggest that the SEERA dataset may not have been an optimal choice for addressing the specific issue at hand. The dataset's non-linear distribution and the absence of certain categories of projects, particularly those with high effort, could have contributed to this limitation. Additionally, the meta data incorporated in the chosen dataset seemed to provide comparatively less informative insights compared to the other datasets [16] mentioned in Section 4.1, as all the analyzed models on the SEERA displayed poor performances.

7 Conclusion

The present study aimed to address the challenge of software development effort estimation by providing a tool to aid managers in this critical activity. Through the exploration of machine learning techniques, particularly the ensemble learning approach of stacking, we found it to be well suited for the task at hand.
Among the various datasets available for software effort estimation, our focus was primarily on the SEERA dataset due to its inclusion of project-related meta-data.

The proposed model exhibited remarkable competitiveness, outperforming both the weak learners and state-of-the-art technologies when evaluated on the same dataset. The overall performances, however, fell short of the established standards, indicating the possibility of inaccuracies in the provided predictions.

To further enhance the accuracy and reliability of effort predictions, future research could explore alternative modeling approaches. For example, one potential avenue of investigation is to apply feature selection techniques for each weak learner within the ensemble. This process could help identify and prioritize the most informative features for each one of them, improving the overall performance of the model. Alternatively, an extension of the dataset could be derived from the bot activity, encompassing a wider range of project types. The expanded dataset could also be utilized for retraining the model, enabling the evaluation of how this process impacts the predictive capabilities. In terms of bot implementation, a potential improvement could involve simplifying user interaction. The commands will be enhanced based on the results of a survey that will be conducted among project managers.

Despite the need for improvements, this study represents a significant step towards developing a valuable tool for assisting managers in software development effort estimation. Notably, the Slack bot developed using the proposed model is the pioneering open-source solution dedicated to this specific purpose.

References

[1] [n. d.]. Effort Estimation Software Development. https://github.com/agupta15k/EffortEstimationSoftwareDevelopment. Online; accessed 12 Luglio 2023.

[2] Ashraf Abdulmunim Abdulmajeed, Marwa Adeeb Al-Jawaherry, and Tawfeeq Mokdad Tawfeeq. 2021. Predict the required cost to develop Software Engineering projects by Using Machine Learning. In *Journal of Physics: Conference Series*, Vol. 1897. IOP Publishing, 012029.

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[4] Asad Ali and Carmine Gravino. 2022. Evaluating the impact of feature selection consistency in software prediction. *Science of Computer Programming* 213 (2022), 102715. https://doi.org/10.1016/j.scico.2021.102715

[5] Amid Khatibi Bardsiri, Seyyed Mohsen Hashemi, and Mohammadreza Razzazi. 2015. Statistical analysis of the most popular software service effort estimation datasets. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 7, 1 (2015), 87–96.

[6] Barry W Boehm. 1984. Software engineering economics. *IEEE transactions on Software Engineering* 1 (1984), 4–21.

[7] JM Desharnais. 1989. Analyse statistique de la productivitie des projets informatique a partie de la technique des point des function. *Masters Thesis, University of Montreal* (1989).

[8] Estimating ISBSG. 2007. Benchmarking & Research Suite Release 9, International Software Benchmarking Standards Group, 2005.

[9] Anupama Kaushik, Ashish Chauhan, Deepak Mittal, and Sachin Gupta. 2012. COCOMO estimates using neural networks. *International Journal of Intelligent Systems and Applications (IJISA)* 4, 9 (2012), 22–28.

[10] Ekrem Kocaguneli, Tim Menzies, and Jacky W. Keung. 2012. On the Value of Ensemble Effort Estimation. *IEEE Transactions on Software Engineering* 38, 6 (2012), 1403–1416. https://doi.org/10.1109/TSE.2011.111

[11] Nicholas Kunz. 2020. *SMOGN: Synthetic Minority Over-Sampling Technique for Regression with Gaussian Noise.* https://pypi.org/project/smogn/

[12] Katrina D Maxwell. 2002. Applied statistics for software managers. *Applied Statistics for Software Managers* (2002).

[13] Emtinan I Mustafa and Rasha Osman. 2020. SEERA: a software cost estimation dataset for constrained environments. In *Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering*. 61–70.

[14] Ali Bou Nassif, Mohammad Azzeh, Luiz Fernando Capretz, and Danny Ho. 2016. Neural network models for software development effort estimation: a comparative study. *Neural Computing and Applications* 27 (2016), 2369–2381.

[15] Sampath Kumar Palaniswamy and R Venkatesan. 2021. Hyperparameters tuning of ensemble model for software effort estimation. *Journal of Ambient Intelligence and Humanized Computing* 12 (2021), 6579–6589.

[16] AG Priya Varshini and K Anitha Kumari. 2020. Predictive analytics approaches for software effort estimation: A review. *Indian J. Sci. Technol* 13 (2020), 2094–2103.

[17] Poonam Rijwani and Sonal Jain. 2022. Software Effort Estimation Development From Neural Networks to Deep Learning Approaches. *Journal of Cases on Information Technology (JCIT)* 24, 4 (2022), 1–16.

[18] Rohit Kumar Sachan and Dharmender Singh Kushwaha. 2020. Anti-predatory NIA based approach for optimizing basic COCOMO model. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 710–715.

[19] Rohit Kumar Sachan, Ayush Nigam, Avinash Singh, Sharad Singh, Manjeet Choudhary, Avinash Tiwari, and Dharmender Singh Kushwaha. 2016. Optimizing basic COCOMO model using simplified genetic algorithm. *Procedia Computer Science* 89 (2016), 492–498.

[20] Martin Shepperd, Qinbao Song, Zhongbin Sun, and Carolyn Mair. 2013. Data Quality: Some Comments on the NASA Software Defect Datasets. *IEEE Transactions on Software Engineering* 39, 9 (2013), 1208–1215. https://doi.org/10.1109/TSE.2013.11

[21] Tianao Wu, Wei Zhang, Xiyun Jiao, Weihua Guo, and Yousef Alhaj Hamoud. 2021. Evaluation of stacking and blending ensemble learning methods for estimating daily reference evapotranspiration. *Computers and Electronics in Agriculture* 184 (2021), 106039. https://doi.org/10.1016/j.compag.2021.106039