IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 June 2023, accepted 26 July 2023, date of publication 23 August 2023, date of current version 28 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3307310

RESEARCH ARTICLE

Applying Machine Learning to Estimate the Effort and Duration of Individual Tasks in Software Projects

ANDRÉ O. SOUSA[1], DANIEL T. VELOSO[1], HENRIQUE M. GONÇALVES[1], JOÃO PASCOAL FARIA[1,2], (Member, IEEE), JOÃO MENDES-MOREIRA[1,2], RICARDO GRAÇA[3], DUARTE GOMES[4], RUI NUNO CASTRO[3], AND PEDRO CASTRO HENRIQUES[4]

[1]Faculty of Engineering, University of Porto, 4200-465 Porto, Portugal
[2]INESC TEC, 4200-465 Porto, Portugal
[3]Fraunhofer Portugal AICOS, 4200-135 Porto, Portugal
[4]Strongstep, 4100-429 Porto, Portugal

Corresponding author: João Pascoal Faria (jpf@fe.up.pt)

This work was supported by the Norte Portugal Regional Operational Program (NORTE 2020), under the Portugal 2020 Partnership Agreement, through the European Regional Development Fund (ERDF) under Project PROMESSA-NORTE-01-0247-FEDER-039887.

ABSTRACT Software estimation is a vital yet challenging project management activity. Various methods, from empirical to algorithmic, have been developed to fit different development contexts, from plan-driven to agile. Recently, machine learning techniques have shown potential in this realm but are still underexplored, especially for individual task estimation. We investigate the use of machine learning techniques in predicting task effort and duration in software projects to assess their applicability and effectiveness in production environments, identify the best-performing algorithms, and pinpoint key input variables (features) for predictions. We conducted experiments with datasets of various sizes and structures exported from three project management tools used by partner companies. For each dataset, we trained regression models for predicting the effort and duration of individual tasks using eight machine learning algorithms. The models were validated using k-fold cross-validation and evaluated with several metrics. Ensemble algorithms like Random Forest, Extra Trees Regressor, and XGBoost consistently outperformed non-ensemble ones across the three datasets. However, the estimation accuracy and feature importance varied significantly across datasets, with a Mean Magnitude of Relative Error (MMRE) ranging from 0.11 to 9.45 across the datasets and target variables. Nevertheless, even in the worst-performing dataset, effort estimates aggregated to the project level showed good accuracy, with MMRE = 0.23. Machine learning algorithms, especially ensemble ones, seem to be a viable option for estimating the effort and duration of individual tasks in software projects. However, the quality of the estimates and the relevant features may depend largely on the characteristics of the available datasets and underlying projects. Nevertheless, even when the accuracy of individual estimates is poor, the aggregated estimates at the project level may present a good accuracy due to error compensation.

INDEX TERMS Effort estimation, duration estimation, machine learning, task estimation, software projects.

I. INTRODUCTION

Software development teams often break down their work into smaller work units, such as tasks or issues. Estimating the effort and time (duration) required to complete them is crucial as a foundation for determining project costs and forecasting delivery dates. This aids in optimizing resource allocation, setting realistic deadlines, and ultimately delivering a high-quality product on schedule.

However, crafting these estimates during the initial planning phases, when they are most valuable, is challenging due to the uncertainties inherent in software development

The associate editor coordinating the review of this manuscript and approving it for publication was Tyson Brooks.

(such as requirements uncertainty) and reliance on the team's prior experiences. A team lacking sufficient experience in the business domain, the technologies involved or the estimation techniques will likely produce poor estimates. Moreover, software estimation can be time-consuming and prone to significant human bias and variance.

For that reason, various estimation methods, from empirical to algorithmic, have been developed over the years to help developers and managers in software estimation in different contexts, from plan-driven to agile. Recently, machine learning (ML) based techniques, particularly ensemble methods, have shown potential for software estimation (see section II). Such techniques combine two ingredients needed by any successful estimation method: use historical data, to reduce bias, and combine multiple estimates, to reduce variance. However, they are still underexplored for individual task estimation, as we intend to do here.

To address the challenges and limitations mentioned above, we developed three software modules, leveraging ML algorithms to estimate the effort and time (duration) required to handle new tasks or issues within different project management tools: SCRAIM from Strongstep, Project Control from Fraunhofer AICOS, and JIRA also utilized by Fraunhofer AICOS. These modules form part of a larger initiative—PROject ManagEment Intelligent aSSistAnt, or PROMESSA—that aims to facilitate tasks such as project prioritization, resource allocation, risk analysis, and effort estimation using ML techniques. These modules offer services to train estimation models based on historical data, extracted from the tools' repositories, and predict the effort and duration of new tasks or issues using those models.

This article aims to present and discuss the results obtained with the three software modules developed, and compare and discuss the differences between them, given that the model in each module was trained with a different dataset. In doing so, we aim to identify the more relevant features when training ML models focused on estimating the effort and duration of issues or tasks in a software development project. By testing the same ML algorithms on three different datasets extracted from the previously mentioned tools, we can assess if any algorithms perform consistently better despite variations in the input data. Lastly, the findings will provide insight into how ML-based estimation methods perform against traditional methods.

More specifically, we aim to address the following research questions:

- **RQ1**: What are the best-performing ML algorithms for task effort and duration estimation?
- **RQ2**: How effective are ML techniques in predicting the effort and duration of individual tasks or issues of software projects in production environments?
- **RQ3**: What are the most influencing features for predicting the effort and duration of individual tasks?

The main contributions of this study are:

- *Exploration of ML for task-level software estimation*: We explore ML algorithms for predicting the effort and duration of individual tasks in software projects, whilst most previous studies only looked at the project as a whole. This contributes to a growing body of literature on ML applications in software engineering.
- *Empirical evaluation of ML algorithms for task-level software estimation*: We empirically compare eight ML algorithms on three datasets provided by two companies using four evaluation metrics. We show that ensemble algorithms, like Random Forest, Extra Trees Regressor, and XGBoost, outperform non-ensemble ones for estimating the duration and effort of individual tasks. This provides useful insights into which algorithms perform best for this specific application.
- *Insights on estimation accuracy*: We show that the models' estimation accuracy can vary significantly across datasets, depending on factors such as the dataset size and degree of variance of the target variable, among others. Even when the accuracy of individual estimates is poor, we show that aggregated estimates at the project level may still be accurate due to error compensation. This enhances our understanding of how project and data specifics influence the accuracy of task estimates and how we interpret and apply these estimates.
- *Insights on feature importance*: We found that the input variables (features) important for accurate task estimation vary significantly across datasets, possibly due to differences in project management scope (macro or micro), project types, and data gathering practices. This could help practitioners to focus data collection efforts on the most important variables and inform the design of future ML models for this task.

The remaining sections of this article are structured as follows. A review of related work on the usage of ML for software estimation is presented in section II. In section III, the datasets used to train the models are described, and the methods used for preprocessing and testing are presented. Section IV presents the results of the algorithms tested in each of the software modules developed, as well as an interpretation of those results. Lastly, section V presents the conclusions extracted based on the work developed and the results obtained, and points out possible future work.

II. RELATED WORK

In the last two decades, extensive research has been made in terms of applying data mining, statistical and ML algorithms for software estimation. Table 1 summarizes the main characteristics of the studies analyzed in this section.

According to Pospieszny et al. [5], most research focused on estimating effort and duration at the initial phases of a project, since forecasts in these stages are generally more challenging due to uncertainty and limited knowledge, and the failure in such initial phases could compromise project success.

TABLE 1. Summary of studies on the usage of ML for effort and duration estimation in software projects.

Study	Data	Target variables	Methods	Results
Pospieszny *et al.* [5]	ISBSG repository, 1,192 projects	Proj. effort & duration	SVM	MMRE: 0.13-0.15, PRED(30): 81-82%
Wen *et al.* [6]	Review of 84 studies	Proj. effort	ANN, CBR, DT	MMRE: 0.35-0.55, PRED(25): 45-75%
Tronto *et al.* [7]	COCOMO, 63 projects	Proj. effort	ANN	MMRE: 0.25-0.63, R^2: 0.80-0.98
Berlin *et al.* [8]	Israeli, 238-245 projects	Proj. effort & duration	ANN	MMRE: 0.44-1.04, PRED(25): 36-56%
López-Martin [9]	ISBSG repository, 355 projects	Proj. effort	RBFNN	MAE: 0.31-0.46, R^2: 0.54-0.68
Rhaman *et al.* [10]	ISBSG repository, 214 projects	Proj. effort	RBFNN, DT	MMRE[1]: 0.38 (DT) - 0.79 (RBFNN)
Minku *et al.* [11]	PROMISE & ISBSG repos, 12 sets	Proj. effort	Single & ensemble	MMRE: 0.37-2.00, PRED(25): 0.17-0.55
Kocaguneli *et al.* [12]	PROMISE repo, 1,198 projects	Proj. effort	Solo & multi	better for multimethods
Fernández *et al.* [15]	UCI ML repository, 83 datasets	Proj. effort	77 models	$R^2_{best} \geq 0.8$ on 49% of the datasets
Ardimento *et al.* [21]	Bugzilla, 8,498 bug reports	Bug resol. slow or fast	NLP, BERT	Accuracy, Precision, Recall & F1 > 0.9

1 - The paper includes the raw data, so we calculated the MMRE for a better comparison with other studies (instead of using the RMSE).

Using several ML algorithms, Pospieszny et al. [5] obtained predictive models with a very good performance for estimating the total project effort and duration, on a dataset provided by the International Software Benchmarking Standards Group (ISBSG). After cleansing, the dataset consisted of 1,192 projects. Using Support Vector Machines (SVM), they achieved Mean Magnitude Relative Error (MMRE) scores of 0.13 and 0.15 and PRED(30) scores of 81.2% and 81.6% in the effort and duration models, respectively, where PRED(30) represents the percentage of samples with the Magnitude of Relative Error (MRE) ≤ 0.30. As input, they used 11 variables describing overall project characteristics (industry sector, application type, development type, development platform, language type, package customization, relative size, architecture, usage of agile methods, used methodology, and resource level), including the project's relative size in function points grouped into categories.

An extensive and comprehensive research was made by Wen et al. [6], reviewing 84 studies concerning ML methods for software effort estimation. According to their results, in the last two decades, researchers focused their attention mainly on tailoring individual algorithms for best performance, such as Artificial Neural Networks (ANN), Case-Based Reasoning (CBR) models and Decision Trees (DT). From this research, it was also clear that ML models performed better than other traditional and statistical ones, with MMRE ranging from 0.35 to 0.55 and PRED(25) from 45 to 75%. The researchers also noted that, depending on the dataset and the preprocessing approach, ML algorithms may perform very differently due to outliers, missing values, or overfitting problems. A recent update to the research of Wen et al. was published by Cabral et al. [24], involving a review of 30 studies. Their findings reinforce previous results and show that ensemble learning techniques have outperformed the individual models.

The discrepancy in results and approaches for building ML models is even more perceptible when analysing individual studies such as the work from Tronto et al. [7], where the accuracy of ANN approaches was compared with multiple regression models for effort estimation using the COCOMO dataset, with MMRE and PRED as evaluation metrics, and product size and several cost drivers as independent variables. Another example is the work of Berlin et al. [8], which examined and compared the accuracy of ANN and Linear Regression (LR) models for effort and duration predictions in an IBSG dataset and an Israeli dataset, using product size, historical productivity, and product complexity (functional complexity) as independent variables. Both concluded that ANN outperformed the other models tested, even though the results differ between them. It is also important to note that in Berlin's research, a log transformation of input and output variables was used to improve the accuracy.

One more example of this discrepancy in results is the work of López-Martin [9], which focused on comparing prediction precision between different neural network types on effort estimation and normalising of dependent variables. As input variable, they used the adjusted function points (AFP) of each project, which measures the functional complexity adjusted by several factors. MAE (mean absolute error), MdAE (median absolute error) and R^2 (coefficient of determination) were used as the evaluation metrics. In this research, the Radial Basis Function Neural Network (RBFNN) performed better with a high confidence level, although, for instance, in Pospieszny's research the MLP-ANN was preferred [5]. Rhaman and Islam [10] compared different types of ANN and Decisions Trees (DTs) using the Root Mean Square Error (RMSE) as the evaluation metric, and the product size as the independent variable, and concluded that DTs were more effective in small-median datasets, whereas the ANNs showed better accuracy in larger datasets.

Other interesting works such as Minku and Yao [11], concerning the sensitivity of ML methods to noise within the datasets, supported the notion that models should not rely on individual algorithms but instead a group of them, which should increase prediction accuracy and strengthen the models to deal with noisy data and overfitting problems as a result. Other publications support this idea, such as Kocaguneli et al. [12], which proposed various ensemble methods, such as boosting, bagging, and complex random sampling techniques. Even though the ensemble techniques may present advantages, Azhar et al. [13] stated that these methods might introduce substantial performance overhead if applied too excessively. With that in mind, Ho [14] concluded that a set of different but limited number of algorithms and simple ensemble approaches, such as averaging the estimates

obtained, should be used for building ML models for effort and duration forecasting.

Another extensive, yet more generic research was done by Fernández-Delgado et al. [15], where a review of 77 popular regression models belonging to 19 families of algorithms was made. All of them were trained using 83 different datasets, splitting the data into 50% for training, 25% for parameter tuning, and the last 25% for testing. A smart preprocessing and cross-validation strategy was used, and MAE, RMSE, and R^2 were used to evaluate and compare the different families of algorithms. This review ranked each algorithm between different-sized datasets and concluded that the most effective algorithms for the various regression problems were in the following order: cubist and M5 regression rules, Gradient Boosting Machines (GBM), Extremely Randomized Trees, Support Vector Machines (SVM), Neural Networks, Project Pursuit (PPR), and Nearest Neighbors. Additionally, a bagging ensemble of MARS models had decent performance. A time and error (in terms of memory and time errors) comparison was also performed, and the conclusion was that the regression rules and boosting ensembles were faster and less error-prone, while SVMs and Neural Networks were more error-prone and slower.

Despite the larger number of approaches taken to build ML models and forecast effort and duration, important recommendations can be extracted to implement them in practice. Some of these recommendations emphasize preprocessing techniques such as the deletion or imputation of values, which, depending on the dataset, can help deal with outliers and missing values [16]. As stated by Strike et al. [17], missing values should be discarded to remove bias. To remove outliers, Ruan et al. [18] propose using the common rule of three standard deviations from a mean. However, since this method may be biased since the mean is also affected by outliers, Leys et al. [19] proposed instead a median absolute deviation approach or the typical interquartile method to detect and remove outliers. Berlin et al. [8] and López-Martín [9] also indicate that log transformation of effort and duration tend to generate more accurate estimates, although not necessary in ML algorithms.

Besides the limitations above and, as stated by Pospieszny's [5], the application of ML methods to small and outdated datasets (*e.g.,* COCOMO, NASA), and the fact that these rarely follow current software methodologies and modern development approaches, led to inconclusive results and fewer implementations of ML algorithms for effort and duration estimation within organisations.

Regarding predictions at the level of individual tasks or issues, there are some studies that employ ML [21] and search-based [22] approaches to predict the resolution time of newly reported issues or bugs, primarily during maintenance stages.

In [21], the authors achieve accuracy, precision, recall, and F1 scores greater than 0.9 in predicting if the resolution time of a newly reported bug will be below ('fast') or above (''slow') the median resolution time in the training dataset. They used a dataset with 8,498 bug reports extracted from the LiveCode's Bugzilla installation. The resolution time of a bug was calculated as the time elapsed between the date when the bug report was assigned for the first time and the date when the bug status was set to 'resolved'. The problem of predicting the resolution time of a bug ('slow' or 'fast') is formulated as a supervised text categorization task, based on the bug description and comments and a pre-trained language model (BERT). However, they didn't try to predict the actual resolution time (or duration) as we intend to do here.

In [22], the authors also try to predict the resolution time of newly reported issues, representing feature requests, bug reports, etc. They used genetic algorithms to iteratively generate candidate models and search for the optimal model to estimate issue resolution time. They used a dataset with 8,260 issues from five large open-source projects, calculating the resolution time for each issue by subtracting the 'resolved' time from the 'created' time. Compared to baselines (based on mean and median) and state of the art techniques (such as Random Forest), they achieved significantly better MAE and Standardized Accuracy (SA) values. SA evaluates the effectiveness of an estimation model relative to random guessing. However, they did not compute metrics based on relative errors, such as MMRE, suitable for comparison across different datasets.

In conclusion, most of the studies found seek to make estimates at the project level, not at the level of individual tasks or issues, as we do here. Besides that, most of the studies take advantage of size or complexity estimates as input to estimate the project effort and duration, whilst we don't assume the existence of any prior size or complexity estimate to help predict the task effort or duration.

III. DATA AND METHODS

A. DATASETS

The three datasets used in this work were provided by two PROMESSA project's partner companies: Strongstep and Fraunhofer AICOS. The relevant variables selected in each dataset (as input features or target variables), as well as other characteristics of those datasets, are summarized in Table 2.

1) SCRAIM DATASET

The first dataset, provided by Strongstep, is a relatively small dataset, consisting of 853 useful samples (tasks) across 27 projects stored in SCRAIM, with the most relevant variables indicated in Table 2.

2) JIRA DATASET

The second dataset, provided by Fraunhofer AICOS, is significantly larger, consisting of an initial set of 11,798 samples (issues) across 68 projects in JIRA, with the most relevant variables indicated in Table 2.

This dataset included a lot of samples that could not be used to train the models due to missing values in target variables or important training features. Additionally, outliers were

TABLE 2. Summary of the most relevant variables selected for effort and duration estimation and characteristics of the datasets.

Variable[1] or Characteristic	Description	Category	SCRAIM	JIRA	Project Control
Description	Text string that describes the task/issue.	Input	X	X [3]	X [2]
Type	Categorical variable describing the task/issue type (*e.g.*, Bug, Feature, Story).	Input	X	X	X [2]
Status	Categorical variable describing the task/issue status (*e.g.*, Open, Doing, Closed).	Input	X	X	X
Priority	Categorical variable describing the task/issue priority (*e.g.*, Minor, Major).	Input	X	X	X
Private	Categorical feature that describes if the task/issue is private or not.	Input	X	-	-
Seq. Num.	Sequence number of the task/issue.	Input	-	-	X
Project ID	Unique ID of the project to which the task/issue belongs.	Input	-	X	-
Assignee ID	Unique ID of the person assigned to work on the task/issue.	Input	-	X	-
Creator ID	Unique ID of the person that created the task/issue.	Input	-	X	-
Reporter ID	Unique ID of the person that reported the task/issue (usually equal to the creator).	Input	-	X	-
#Participants	Number of participants in the task/issue.	Input	X	-	-
#Sprints	Number of sprints the task/issue is associated with.	Input	-	X	-
#Links	Number of tasks/issues linked to the current one.	Input	-	X	-
P Type	Categorical variable indicating the type of the enclosing project.	Input	-	-	X
P Duration	Numerical variable indicating the duration of the enclosing project/baseline.	Input	-	-	X
P Funding Type	Categorical variable indicating the funding type for the project.	Input	-	-	X [5]
WP Type	Categorical variable indicating the type of the enclosing workpackage (WP).	Input	-	-	X
WP Duration	Numerical variable indicating the duration of the enclosing WP.	Input	-	-	X
WP Seq. Num.	Sequence number of the enclosing WP.	Input	-	-	X
Effort	Effort required to complete the task or issue.	Target	X	X	X
Duration	Time required to complete the task or issue from start to finish.	Target	X	X	X
#Projects	Number of projects in the dataset.	Meta	27	68	1,127
#Samples	Number of tasks/issues in the dataset.	Meta	853	11,798 [4]	54,153
Provider	Provider of the dataset.	Meta	Stronstep	Fraunhofer	Fraunhofer

1 - The actual field names and possible values may vary by dataset.
2 - The task type was derived from the task description in the Project Control dataset and used instead of it. This was done by checking the presence in each task description of keywords associated with each possible task type. In 32% of the tasks, no match was found, so they were categorized as 'Others'.
3 - The issue description was not used in the JIRA dataset, because the results obtained were not better (or were even worse) than the ones obtained without using it, for a higher processing complexity.
4 - The number of samples in the JIRA dataset was reduced to 3,435 and 4,094 for the Effort and Duration estimation models, respectively, after removing samples with missing values and outliers in target variables or important training features.
5 - The Project Control dataset included five additional input variables, used as training features with less importance, related to other project characteristics.

identified and consequently removed too. As a result, the number of samples used for the effort and duration estimation models dropped to 3,435 and 4,094, respectively.

3) PROJECT CONTROL DATASET

Lastly, the third dataset, also provided by Fraunhofer AICOS, consists of 54,153 samples (tasks) in 1,127 projects, all housed within a MySQL database integral to Project Control - the company's proprietary system for macro-managing R&D&I software projects, from proposal drafting and approval to execution and reporting.

In Project Control, a project plan is essentially a Gantt chart, in which the project is broken down into workpackages and tasks, with planned duration (in months) and start and finish dates (in months since the beginning of the project) for each workpackage and task, and planned resource allocation for each task (e.g., one person half-time). A project plan may have several baselines over time, at important (re)planning moments (proposal drafting, re-planning after approval, etc.).

During project execution, team members register their time allocation to project tasks in timesheets, hence allowing computing the actual effort per task.

The target variables for this dataset are the task duration (in months) and task effort (in person-months), i.e., the goal is to help project managers estimate the duration and effort of each task, based on task attributes or attributes derived from related entities available at the time of planning.

This dataset is our largest dataset; it is also unique as it provides information on macro-management tasks, providing a broader perspective on project management.

B. METHODS

The tests were performed on eight ML algorithms: Random Forest (RF), Extra Trees Regressor (ETR), Gradient Boosted Trees (GBT), XGBoost, Lasso, K-Nearest Neighbors (KNN), Support Vector Regression (SVR), and Artificial Neural Network (Multilayer perceptron, MLP-ANN). The first four are ensemble algorithms.

The rationale behind selecting these algorithms is to independently assess various modelling strategies, each with distinct strengths and theoretical underpinnings. By evaluating both ensemble and non-ensemble methods, we aim to understand the performance trade-offs in terms of accuracy, robustness, interpretability, and complexity in the context of software project data. This can help determine the most suitable algorithms for predicting the effort and duration of individual tasks in software projects.

In order to proceed with the data preprocessing and the hyperparameter tuning, a validation set with 25% of the data was split as better explained in each subsection.

The overall processing workflow is illustrated in Figure 1.

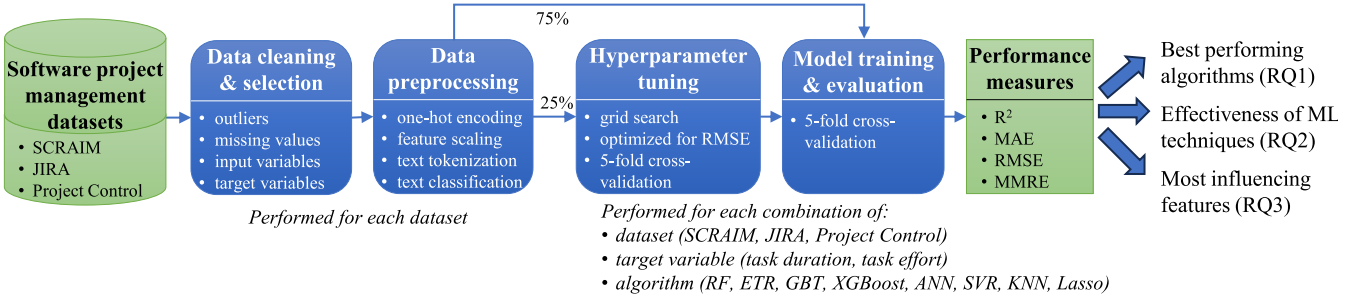


FIGURE 1. Processing workflow followed for each combination of dataset, target variable and algorithm.

1) DATA PREPROCESSING

As for data preprocessing tasks that were common across all projects, the categorical variables were encoded through *one-hot encoding*. This technique represents each class of the categorical variable through a new attribute with a value of 0 or 1, where 0 indicates the absence of that class, and 1 indicates its presence. This was used, for example, in the Type, Status, and Priority variables of the second dataset.

Feature scaling was used to scale the range of values of quantitative independent variables, such as the #Participants variable in the first dataset, or the #Sprints and #Links in the second dataset. This is an important step in the data preprocessing workflow because, if one attribute's range of values differs significantly from the range of values of another attribute, the former will become dominant in models that use distance measures, e.g., K-Nearest Neighbors, negatively affecting the predictive performance of these models.

Tokenization was applied to text fields, namely the task description in the first dataset. It is used to transform each task description into a vector of word counters. The vector contains an entry for each word that occurs in the set of all task descriptions in the dataset, ignoring stop words and punctuation marks. Before tokenization, the words were reduced to their root form (*stemming*).

2) HYPERPARAMETER TUNING

Before running tests for the different algorithms, *hyperparameter tuning* was performed with the goal of obtaining the best set of hyperparameters for each algorithm, dataset and target variable (in a total of 48 combinations). Hyperparameters differ from an algorithm's internal parameters in that they cannot be learned from the data during the training phase [1].

This process was performed in the validation set (containing a random sample of 25% of the data) with the use of 5-fold cross-validation, grid search, and RMSE as the metric to optimize. As an example, the optimal number of neighbours obtained for the KNN algorithm was 3 for both target variables in the Project Control dataset.

3) MODEL EVALUATION

To assess the quality of the models trained, various evaluation metrics were used:

- **R^2** - also known as the coefficient of determination, it refers to the proportion of variance in the target variable that can be explained by the independent variables used [2]. A higher value means that an equally higher proportion of the aforementioned variance can be explained by the independent variables.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Yi - \widehat{Y}i)^2}{\sum_{i=1}^{n}(Yi - \overline{Y}i)^2} \tag{1}$$

- **Mean Absolute Error (MAE)** - the average of the difference between the predicted values and the actual ones. Can be used to check how far the predictions made were from the real results [3].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Yi - \widehat{Y}i\right| \tag{2}$$

- **Root Mean Square Error (RMSE)** - this metric gives errors with larger absolute values more weight, penalizing variance in the dataset as a result [4]. A lower value of RMSE indicates better results. Most of the ML algorithms we use (including all the ensemble ones) try to minimise the RMSE, by default.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2} \tag{3}$$

- **Mean Magnitude of Relative Error (MMRE)** - measurement of the difference between the actual values and the predicted values, relative to the actual values. A lower value of MMRE indicates better predictive performance. It is a scale-independent metric suitable for comparison across different datasets. But it has also been criticised due to its extreme sensitivity to individual predictions with excessively large MREs [23].[1]

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|Yi - \widehat{Y}i\right|}{Yi} \tag{4}$$

These metrics provide different information, thus analysing their results provides a lot of insight into the quality of the predictions made by the models developed.

[1] In the case of underestimates, the MRE may range from 0 to 1, whilst in the case of overestimates it may range from 0 to ∞.

TABLE 3. Results of the SCRAIM task duration model evaluation with different algorithms.

Algorithm	R^2	MAE (hours)	RMSE (hours)	MMRE
Random Forest	**0.37**	1.99	**3.23**	0.75
Gradient Boosted Trees	0.33	2.04	3.29	0.79
Extra Trees Regressor	0.35	2.02	3.27	0.78
XGBoost	0.32	**1.88**	3.28	0.64
Neural Network (MLP-ANN)	0.22	2.24	3.59	0.80
Support Vector Regression	0.13	1.96	3.80	**0.56**
K-Nearest Neighbors	0.17	2.11	3.69	0.67
Lasso	0.24	2.23	3.52	0.92

TABLE 4. Results of the SCRAIM effort estimation model evaluation with different algorithms.

Algorithm	R^2	MAE (hours)	RMSE (hours)	MMRE
Random Forest	**0.51**	**2.61**	**5.29**	0.67
Gradient Boosted Trees	0.48	2.84	5.53	0.84
Extra Trees Regressor	0.41	3.10	5.87	0.96
XGBoost	0.35	2.77	5.93	**0.64**
Neural Network (MLP-ANN)	0.30	2.84	5.67	0.65
Support Vector Regression	0.19	3.08	6.58	0.74
K-Nearest Neighbors	0.19	3.25	6.81	0.76
Lasso	0.29	3.27	6.38	0.81

K-fold cross-validation with $k = 5$ was used during the testing phase to obtain the results that will be presented. In this process, the validation set (25% of samples) was set aside temporarily, and the remaining 75% of samples are used in the K-fold instance. With $k = 5$, the K-fold instance will split them into 5 different groups. Then, leaving a different group for testing, all the remaining groups together with the validation set are used for training. This is repeated 5 times leaving a different group from the five for testing (cross-validation process).

IV. RESULTS AND DISCUSSION

A. SCRAIM DATASET

1) DURATION ESTIMATION WITH THE SCRAIM DATASET

The SCRAIM dataset was used to train the task duration model. The results are presented in Table 3.

The best R^2 score (0.37) and best RMSE score (3.23) were obtained by Random Forest, the best MAE score (1.88) was achieved by XGBoost, and the best MMRE score (0.56) was achieved by Support Vector Regression.

Overall, ensemble algorithms are the best options for this particular dataset regarding the R^2, MAE and RMSE metrics. Support Vector Regression presents the best MMRE, but with a significantly worse R^2, so ensemble algorithms seem to be the better options with this dataset.

2) EFFORT ESTIMATION WITH THE SCRAIM DATASET

The effort estimation model was trained using the SCRAIM dataset. Table 4 presents the results.

The best R^2 (0.51), MAE (2.61), and RMSE (5.29) scores are presented by the Random Forest algorithm. XGBoost presents the best MMRE (0.64).

TABLE 5. Results of the JIRA issue duration model evaluation with different algorithms.

Algorithm	R^2	MAEhours (hours)	RMSE (hours)	MMRE
Random Forest	0.33	127.11	167.01	10.22
Gradient Boosted Trees	0.31	135.32	169.69	10.19
Extra Trees Regressor	0.28	130.45	174.34	10.24
XGBoost	**0.37**	**125.52**	**162.87**	10.27
Neural Network (MLP-ANN)	0.23	140.92	179.24	10.17
Support Vector Regression	0.17	144.52	186.35	**9.45**
K-Nearest Neighbors	0.19	145.96	184.75	9.74
Lasso	0.14	154.10	189.80	10.18

TABLE 6. Results of the JIRA effort estimation model evaluation with different algorithms.

Algorithm	R^2	MAE (hours)	RMSE (hours)	MMRE
Random Forest	**0.36**	5.43	**9.23**	5.62
Gradient Boosted Trees	0.30	5.65	9.64	5.57
Extra Trees Regressor	0.23	5.72	10.11	5.60
XGBoost	0.18	**5.25**	10.45	3.22
Neural Network (MLP-ANN)	0.07	6.38	11.12	5.55
Support Vector Regression	-0.04	5.97	12.34	**2.65**
K-Nearest Neighbors	0.10	5.99	10.97	5.30
Lasso	0.05	6.40	11.23	5.53

Overall, ensemble algorithms performed better than the other algorithms on all evaluation metrics.

B. JIRA DATASET

1) DURATION ESTIMATION WITH THE JIRA DATASET

Table 5 shows the results of the tests performed for the issue duration model trained with the JIRA dataset.

The XGBoost algorithm presents the best scores for R^2 (0.37), MAE (125.52) and RMSE (162.87). Support Vector Regressions presents the best score for MMRE (9.45).

Overall, ensemble algorithms outperform the other algorithms regarding the R^2, MAE and RMSE metrics. Support Vector Regression presents the best MMRE, but with significantly worse scores for the remaining metrics, so ensemble algorithms seem to be the better options with this dataset.

We did not include the issue description in the training features, because the performance results using it did not improve, for a higher processing complexity. Using the issue description, the best scores for R^2 (0.35), MAE (126.20) and RMSE (165.20) and MMRE (9.45) are worse than those obtained without using the issue description (Table 5).

2) EFFORT ESTIMATION WITH THE JIRA DATASET

Table 6 shows the results of the tests performed for the issue effort model trained with the JIRA dataset.

This time, Random Forest presents the best scores for R^2 (0.36) and RMSE (9.23), while its MAE score was only behind XGBoost's. Once again, Support Vector Regressions presents the best score for MMRE (2.65).

Once more, ensemble algorithms outperform the other algorithms regarding the R^2, MAE and RMSE metrics.

Support Vector Regression presents the best MMRE score, but with significantly worse scores for the remaining metrics.

We did not include the issue description in the training features, for reasons similar to the ones presented for the duration estimation model. Using the issue description, the best scores for R^2 (0.26), MAE (5.58) and RMSE (10.82) and MMRE (0.92) in the effort estimation models, are worse than the best scores obtained without using the issue description (Table 6). The only exception is a significant improvement in MMRE (0.94) with Neural Network, but with much worse scores in the other metrics.

3) COMPARISON WITH DEVELOPER ESTIMATES

In the JIRA dataset, effort estimates performed by the developers were also available for a subset of the samples (1,371 out of 3,435 samples), besides the actual effort spent. Hence, a test was also performed to evaluate how the model's predictions compared to the developers' predictions.

Regarding the developers' estimates, the MAE for this new dataset is 5.53, worse than the MAE of the best-performing models in Table 6. For a more direct comparison between the developers' estimates and the model-based estimates, we trained new effort estimation models based on the smaller dataset (with 1,371 samples). In this dataset, all ensemble algorithms achieved better MAE scores (5.06 to 5.38) than the developers (5.53).

Given that ML-based methods for effort estimation are less time-intensive than traditional approaches, our findings highlight the potential advancements these techniques offer in the field, especially if more data of good quality can be used to train the models.

4) COMPARISON WITH REFERENCE THRESHOLDS

Even if the ML-based estimates are better than the developer estimates in the JIRA dataset, no algorithm gets close to the threshold proposed by Jorgeensen and Shepperd [20], who consider a model accurate when MMRE ≤ 0.25, among other criteria.

However, it is important to note that such thresholds have been primarily proposed and applied for estimates at the project level, as opposed to our focus on individual tasks or issues. In section IV-E, we will further discuss this issue.

C. PROJECT CONTROL DATASET

1) DURATION ESTIMATION WITH THE PROJECT CONTROL DATASET

Table 7 shows the results of the tests performed for the task duration model trained with the Project Control dataset.

All ensemble algorithms consistently outperform the other algorithms in all metrics, presenting an equal score for R^2 (0.96).

The ensemble models also satisfy the model accuracy thresholds defined by Jorgeensen and Shepperd [20]: MMRE ≤ 0.25 and PRED(0.3) $\geq 75\%$. The Extra Trees Regressor model performs best on these metrics, with MMRE = 0.11 and PRED(0.3) = 91%, which can be considered very good for practical application.

TABLE 7. Results of the Project Control task duration model evaluation with different algorithms.

Algorithm	R^2	MAE (months)	RMSE (months)	MMRE
Random Forest	**0.96**	0.69	1.90	0.17
Gradient Boosted Trees	**0.96**	0.48	**1.78**	0.12
Extra Trees Regressor	**0.96**	**0.46**	1.79	**0.11**
XGBoost	**0.96**	0.51	**1.78**	0.12
Neural Network (MLP-ANN)	0.73	2.43	4.97	0.56
Support Vector Regression	0.92	0.92	2.77	0.23
K-Nearest Neighbors	0.91	1.00	2.78	0.23
Lasso	0.60	4.34	6.06	1.06

TABLE 8. Results of the Project Control task effort model evaluation with different algorithms.

Algorithm	R^2	MAE (person-months)	RMSE (person-months)	MMRE
Random Forest	0.89	0.55	1.86	0.50
Gradient Boosted Trees	**0.90**	0.46	**1.77**	0.37
Extra Trees Regressor	**0.90**	**0.45**	1.81	**0.36**
XGBoost	**0.90**	0.46	1.78	0.39
Neural Network (MLP-ANN)	0.75	1.18	2.79	1.59
Support Vector Regression	0.83	0.62	2.28	0.65
K-Nearest Neighbors	0.81	0.76	2.41	0.63
Lasso	0.26	2.31	4.84	2.96

2) EFFORT ESTIMATION WITH THE PROJECT CONTROL DATASET

Table 8 shows the results of the tests performed for the task effort model trained with the Project Control dataset.

Once more, all ensemble algorithms consistently outperform the other algorithms in all metrics.

The Extra Trees Regressor performs best on the MMRE and PRED(0.3) metrics, with MMRE = 0.36 and PRED(0.3) = 80%. Hence it meets the aforementioned model accuracy threshold for PRED(30), and is close to meeting the threshold for the MMRE. Hence, this model should be applied in practice with caution. Nevertheless, much smaller MMRE scores are expected to occur due to error compensation when effort estimates are aggregated to the project level (see section IV-E).

D. FEATURE IMPORTANCE

Assessing the significance of various features used in training is crucial. It helps unravel whether the variation in metric results largely stems from the disparity in data samples used for training the models, the models assigning different levels of importance to various features, or a blend of both. This analysis aids in enhancing our understanding of the models' performance and underlying mechanisms.

1) FEATURE IMPORTANCE IN THE PROJECT CONTROL DATASET

Table 9 shows the ranking of feature importance given by a Random Forest Regressor for the task duration target variable

TABLE 9. Feature importance evaluation for the task duration target variable in the Project Control dataset (top 6).

Variable	Description	Relative Importance
WP Duration	Workpackage duration	100
P duration	Project duration	41
Type	Task type	40
Seq. Num.	Task sequence number	17
WP Seq. Num.	WP sequence number	15
WP Type	Workpackage type	11

TABLE 10. Feature importance evaluation for the task effort target variable in the Project Control dataset (top 6).

Variable	Description	Relative Importance
Duration	Task duration	100
WP Duration	Workpackage duration	48
WP Seq. Num.	WP sequence number	47
P Funding Type	Project funding type	46
Type	Task type	41
Seq. Num.	Task sequence number	34

in the Project Control dataset. Unsurprisingly, duration information from the higher-level nodes of the project structure (workpackage and overall project) plays a key role in predicting individual task duration, followed by a task-specific attribute (task type). The top-down planning approach validates the use of these predictors. Since task duration is bound by the timings of the workpackage and overall project, it's logical that longer projects and workpackages often lead to longer tasks. The influence of the task and workpackage numbering (the next influencing variables) might be explained by patterns in project structuring (e.g., longer tasks preceding shorter ones).

Table 10 shows the ranking of feature importance given by a Random Forest Regressor for the task effort target variable in the same dataset. The task duration is the most important variable, followed by workpackage duration and sequence number. Once more, duration-related variables are considered the most important. Using the planned task duration to predict the task effort is valid because of the planning approach followed (the task duration is estimated before the effort). The effort required for a task (e.g., two person-months) is the result of a resource allocation (e.g., one person half-time) by the task's duration (e.g., four months). Given the potential for resource allocations to exhibit a degree of uniformity across tasks, understanding the duration of a task and its corresponding workpackage is likely to offer valuable insight into the spectrum of effort values deemed acceptable for the task.

2) FEATURE IMPORTANCE IN THE JIRA DATASET

Feature importance was also analysed in the JIRA dataset, with the goal of evaluating what features are considered more important for each target variable and trying to find features that are consistently important across the two target variables. Tables 11 and 12 show the ranking of feature importance given by a Random Forest Regressor for the duration and effort target variables, respectively.

TABLE 11. Feature importance evaluation for the issue duration target variable in the JIRA dataset (top 6).

Variable	Description	Relative Importance
Assignee ID	Assignee ID	100
Status Resolved	Issue of status Resolved	70
Type Bug	Issue of type Bug	68
#Sprints	Num. of associated sprints	42
Creator ID	Creator ID	37
Reporter ID	Reporter ID	34

TABLE 12. Feature importance evaluation for the issue effort target variable in the JIRA dataset (top 6).

Variable	Description	Relative Importance
Assignee ID	Assignee ID	100
#Sprints	Num. of associated sprints	76
Creator ID	Creator ID	75
Status Resolved	Issue of status Resolved	72
Reporter ID	Reporter ID	65
Type Bug	Issue of type Bug	25

Overall, despite some variations in relative importance, the key features for both models do not differ a lot. It is then likely that the difference seen in their R^2 scores is caused by the smaller size of the dataset used for the issue effort tests as compared to the issue duration tests.

Based on further analysis of the datasets and discussions with their providers, the importance of these features might be explained as follows:

- Assignee ID: could be attributed to the variability in work pace among individuals;
- Reported ID and Creator ID (usually identical): higher-level staff, like product owners, may create broader tasks, while developers or QA personnel often create shorter ones;
- #Sprints: issues tied to more sprints often require increased effort and time. Reasons may include task extension, issue reopening, etc., necessitating effort re-estimation at each sprint's start. This effect is expected as our dataset targets the total issue effort and duration, not per sprint;
- Status Resolved: typically, 'Resolved' issues—those addressed but awaiting testing—register less time and effort compared to those in subsequent stages like 'Done' or 'Closed'. Training a model using 'Resolved' issues may be useful if developers want to predict the time and effort required until resolution, rather than the final closure, which may depend on other personnel;
- Type Bug: issues of type 'Bug' tend to require significantly less time and effort to fix than other types of issues (e.g., 'New feature').

3) FEATURE IMPORTANCE IN THE SCRAIM DATASET

Because tokenization replaces the task description by a large number of training features (one per word), it is not practical to analyze feature importance in the same way as in the previous datasets.

TABLE 13. Model performance in the SCRAIM dataset depending on the usage of the task description.

Target variable With(out) task description	Max. R^2	Min. MAE	Min. RMSE	Min. MMRE
Effort				
With task description	**0.51**[1]	**2.61**[1]	**5.29**[1]	**0.64**[2]
Without task description	0.35	2.77	5.93	0.85
Duration				
With task description	**0.37**[1]	**1.88**[2]	**3.23**[1]	**0.56**[3]
Without task description	0.00	2.31	4.05	0.83

1 - Random Forest; 2 - XGBoost; 3 - Support Vector Regression

Instead, two estimation models were built for each target variable and algorithm, one including the task description in the input variables (tokenized) and the other excluding the task description, and their predictive performance was compared.

The performance results are summarized in Table 13. For each metric and experiment, it is shown the value of the best-performing algorithm. The detailed results for the different algorithms, using the task description, have been presented before in Tables 3 and 4.

The results obtained using the task description in the training features presented much better results. In fact, this feature is so important that without it all the R^2 scores were negative or 0 in the duration estimation model, which means that, without the task description, no algorithm can explain the relationship between the input variables and the task duration. The elimination of other features from the set of input variables led to smaller performance degradation, so it is possible to conclude that for both SCRAIM models (task duration and task effort estimation), the task description feature is the most important one.

E. COMPARISON WITH RELATED WORK AND EFFECT OF ESTIMATION GRANULARITY

As pointed out in section II, most of the existing studies seek to make estimates at the project level, not at the level of individual tasks or issues, as we do in this article, so the performance results are not directly comparable. Besides that, most of the studies take advantage of size or complexity estimates as input for project effort and duration estimation, whilst we don't assume the existence of any size or complexity estimates (e.g., story points) to help predict the effort or duration required by a task or issue.

For example, Pospieszny et al. [5] achieved very good MMRE scores (0.13-0.15) and PRED(30) scores (81-82%) in predicting overall project effort and duration in an ISBSG dataset with 1,192 projects. Their input variables included the project's relative size in function points grouped into categories (similar to T-shirt sizes) – usually an important predictor of the project effort and duration. An additional difference to our work is that they log-transformed the target variables, but did not apply the reverse transformation when computing the performance metrics, which possibly led to overly optimistic scores, as acknowledged by the authors.

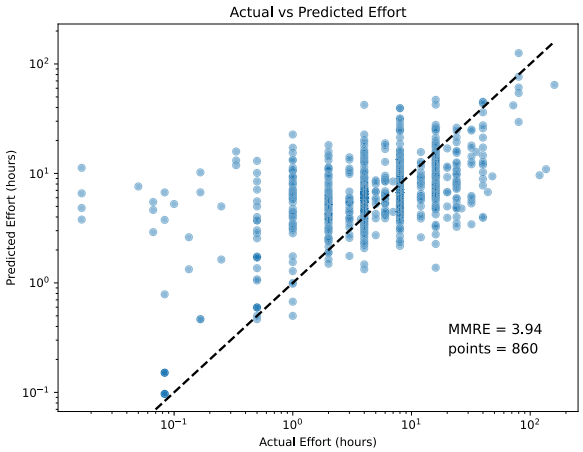


FIGURE 2. Actual versus predicted effort by issue in the JIRA dataset.

As mentioned in section II, there are some studies that try to predict the resolution time of newly reported issues or bugs. Ardimento and Mele [21] try to predict if the resolution of a bug will be 'slow' or 'fast', but not the actual resolution time (or duration) as in our case. Al-Zubaidi et al. [22] try to predict the resolution time of newly reported issues, representing feature requests, bug reports, etc. but don't report metrics suitable for comparison across different datasets.

Because of the error compensation that occurs when individual estimates are aggregated (*e.g.*, when summing the effort estimates of individual tasks to arrive at an effort estimate for the whole project), it is logical that the MRE of the aggregated project estimate is smaller than the MRE of individual task estimates, especially if the individual estimates are unbiased and independent.

In general, obtaining accurate effort and duration estimates at the level of individual tasks or issues is much more challenging, and very few works address such challenges. In practice, individual effort estimates with a small accuracy but a good balance between over and under-estimates may lead to accurate estimates at the project level - the most important ones for making commitments with customers. This phenomenon can be observed even in our dataset that presented inferior effort estimation performance - the JIRA dataset, as depicted in Figures 2 and 3.

Figure 2 illustrates the correlation between actual effort values and those predicted by a Random Forest Regressor in a test dataset. The actual values display a certain degree of discretization as time logging is often performed in multiples of hours. In contrast, each point in Figure 3 represents a unique project within the test dataset; the 'x' and 'y' coordinates indicate the cumulative actual and predicted effort values for the specific project, respectively. The differences in estimation accuracy are striking, either by visual inspection or by observing that the MMRE drops from 3.94 to 0.23 when task effort estimates are aggregated to the project level.

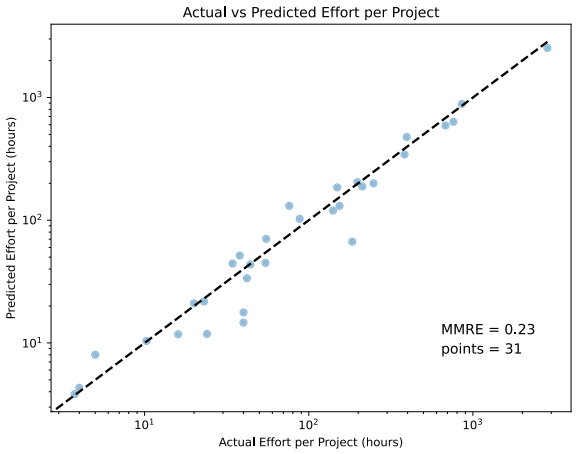


FIGURE 3. Actual versus predicted effort by project in the JIRA dataset.

TABLE 14. Top-performing algorithm for each dataset and target variable.

Data set	Target variable	Best algorithm	Normalized score
SCRAIM	Duration	XGBoost	92.12
SCRAIM	Effort	Random Forest	98.88
JIRA	Duration	XGBoost	98.00
JIRA	Effort	Random Forest	85.96
Project Control	Duration	Extra Trees Regressor	99.86
Project Control	Effort	Extra Trees Regressor	99.45
overall	overall	XGBoost	92.02

F. ANSWERS TO RESEARCH QUESTIONS

We next try to answer our initial research questions.

RQ1: What are the best-performing ML algorithms for task effort and duration estimation?

Table 14 shows the top-performing algorithm for each dataset and target variable, based on normalized scores computed as follows: (i) for each experiment and metric, the best-performing algorithm is given a normalized score of 100; (ii) the remaining algorithms are given normalized scores relative to this best-performer; (iii) finally, each algorithm's overall score is calculated by averaging its normalized scores across all four metrics we're considering.

The top-performing algorithms are all ensemble-based – Random Forest, Extra Trees Regressor and XGBoost. Overall, across all datasets and target variables, XGBoost is the best-performing, with an average normalized score of 92,02, closely followed by the other ensemble-based algorithms (with values between 88.70 and 92.02), and at a significant distance to the best-performing non-ensemble algorithm (Supper Vector Regression, with 72.19).

Our findings across three datasets suggest that ensemble-based algorithms, like Random Forest, Extra Trees Regressor and XGBoost, are the best choice for task effort and duration estimation.

RQ2: How effective are ML techniques in predicting the effort and duration of individual tasks or issues of software projects in production environments?

TABLE 15. Effectiveness of the top-performing algorithm for each dataset and target variable.

Dataset	Target v.	#Samples	#Inp.v.	R^2	MMRE
SCRAIM	Duration	853	6	0.32	0.64
SCRAIM	Effort	853	6	0.51	0.67
JIRA	Duration	4,094	9	0.37	10.27
JIRA	Effort	3,435	9	0.36	5.62
Project Control	Duration	54,153	15	0.96	0.11
Project Control	Effort	54,153	16	0.90	0.36

Table 15 shows, for each dataset and target variable, the scores obtained by the top-performing algorithm (see Table 14) for the metrics that are comparable across datasets (R^2 and MMRE), as well as the number of samples in each dataset and the number of input variables used for training.

Overall, the coefficient of determination (R^2) and the estimation accuracy, as measured by the MMRE, varied significantly across the datasets, with the best results achieved with the Project Control dataset.

Taking as reference the thresholds proposed by Jorgeensen and Shepperd [20] for classifying an estimation model accurate (MMRE ≤ 0.25 and PRED(0.3) $\geq 75\%$), the estimation accuracy of our models can be classified as good in the third dataset, poor in the second dataset, and intermediate in the first dataset.

However, it is important to note that these thresholds have been primarily proposed and applied for estimates at the project level, as opposed to our focus on individual tasks or issues. In Section IV-E, we demonstrate that even when the accuracy of individual estimates is poor, the aggregated estimates at the project level may present a good accuracy due to error compensation.

Hence further research is needed to determine the best metrics and thresholds for assessing the accuracy of effort and duration estimation models at the level of individual tasks or issues in software projects.

Regarding the causes of performance variation across the different datasets, the results show that having more data samples doesn't always result in better models. Even though the models trained with the Project Control dataset (which had the most data samples and training features) performed the best, models trained with the SCRAIM dataset performed better than those trained with the JIRA dataset, despite having fewer data samples and training features.

Therefore, other factors, besides the amount of data or the number of features, also play a significant role. For instance, the JIRA dataset contains issues of very diverse granularity, with effort ranging from as little as 1 minute to as much as 240 hours. If the predicted effort for a 1-minute task is 1 hour, its MRE will be 60. Given that the MMRE is extremely sensitive to individual predictions with excessively large MRE, this dataset's resulting high MMRE of 5.62 isn't surprising. On the other hand, the SCRAIM dataset covers a narrower range of effort, from 30 minutes to 147 hours, potentially contributing to more accurate overall predictions and a smaller MMRE. These observations suggest that the degree

of variance of the target variable may also be an important influencing factor on the prediction accuracy. Attaining high prediction accuracy can be challenging for target variables with greater variance unless we possess highly predictive variables to compensate for this variability.

RQ3: What are the most influencing features for predicting the effort and duration of individual tasks?

In the experiments conducted, the most influential features varied largely across datasets, possibly due to different underlying project management and data-gathering approaches.

The Project Control dataset is significantly different from the other two, as it comprises macro-management data produced according to a top-down planning approach, with time granularity in months, whilst the SCRAIM and JIRA datasets comprise mainly micro-management data, with time granularity in hours. Hence, it is natural that the most important features for estimating task effort and duration at the macro and micro-management levels are significantly different.

The main difference between the SCRAIM and JIRA datasets is the disparate importance of the task description feature in different datasets, with a high importance in the SCRAIM dataset and a very low (almost null) importance in the JIRA dataset, possibly due to different types of projects and data gathering practices in the respective source companies. In future work, we intend to explore other techniques, based on pre-trained language models, to try to take better advantage of the task descriptions.

G. THREATS TO VALIDITY

We next indicate some threats to validity that could be present in our research and measures we took to mitigate them.

1) EXTERNAL VALIDITY

Our study utilizes data from project management tools used by our corporate partners. Consequently, there's a risk that the results may not extend seamlessly to other software development contexts or industries. Furthermore, the specific characteristics of these datasets could influence the performance of the ML models, potentially limiting their efficacy with diverse types of data or projects. To address this concern, we've analyzed multiple datasets and identified patterns and variations in the outcomes across different datasets.

2) INTERNAL VALIDITY

Our study involves the application of various ML algorithms and evaluation metrics, raising concerns that the tuning of these algorithms and choice of metrics could potentially influence results. To mitigate this, we adopted best practices for hyperparameter tuning and employed multiple evaluation metrics.

3) REPRODUCIBILITY

In this study, we made use of proprietary datasets supplied by our corporate partners. This may pose challenges for other researchers aiming to reproduce the results due to limited access to the data. We recognize this constraint as an unavoidable consequence of using authentic corporate data. Although we've anonymized certain sensitive information, privacy concerns inhibit us from making the datasets publicly available.

V. CONCLUSION

With the goal of evaluating the applicability and effectiveness of ML-based techniques to estimate the effort and duration of individual tasks or issues in software projects, three ML-based software modules were developed. They provide estimates on issue effort and duration in three project management tools: SCRAIM, Project Control, and JIRA.

While the datasets used were different, the experimental setup remained as equal as possible to increase the relevance of the comparisons and analysis performed. The same set of algorithms was also tested across the three software modules, with the intention of attempting to find algorithms that performed consistently well.

After analysing the results of the tests, we found that ensemble-based algorithms such as Random Forest, Gradient Boosted Trees, Extra Trees Regressor, and XGBoost were consistently the best-performing algorithms across all datasets. While the order varied depending on the evaluation metric and the dataset, they were always among the four best-performing algorithms regarding most of the evaluation metrics. This matched some of the results found in the literature, such as the research of Pospieszny et al. [5], Kogacuneli et al. [12], or Fernández-Delgado et al. [15].

These results show that, with the right type and amount of information, ML-based approaches for the prediction of task effort and duration are viable. The models developed have been integrated into the project's main module, making them ready to be used by the three project management tools that have been previously mentioned.

In future work, we plan to explore other text preprocessing techniques, namely pre-trained language models like BERT.

REFERENCES

[1] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on Bayesian optimization," *J. Electron. Sci. Technol.*, vol. 17, pp. 26–40, Mar. 2019, doi: 10.11989/JEST.1674-862X.80904120.

[2] F. Moksony, "Small is beautiful: The use and interpretation of R^2 in social research," *Szociológiai Szemle*, pp. 130–138, Jan. 1999.

[3] S. Bhatia and V. K. Attri, "Machine learning techniques in software effort estimation using COCOMO dataset," *J. Comput. Sci. Eng.*, vol. 1, no. 6, pp. 101–106, Jun. 2015.

[4] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?—Arguments against avoiding RMSE in the literature," *Geoscientific Model Develop.*, vol. 7, no. 3, pp. 1247–1250, Jun. 2014, doi: 10.5194/GMD-7-1247-2014.

[5] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," *J. Syst. Softw.*, vol. 137, pp. 184–196, Mar. 2018.

[6] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Inf. Softw. Technol.*, vol. 54, no. 1, pp. 41–59, Jan. 2012.

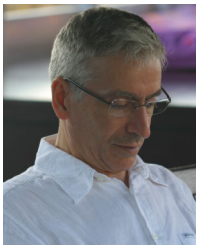
[7] I. F. de Barcelos Tronto, J. D. S. da Silva, and N. Sant'Anna, "An investigation of artificial neural networks based prediction systems in software project management," *J. Syst. Softw.*, vol. 81, no. 3, pp. 356–367, Mar. 2008.

[8] S. Berlin, T. Raz, C. Glezer, and M. Zviran, "Comparison of estimation methods of cost and duration in IT projects," *Inf. Softw. Technol.*, vol. 51, no. 4, pp. 738–748, Apr. 2009.
[9] C. López-Martín, "Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects," *Appl. Soft Comput.*, vol. 27, pp. 434–449, Feb. 2015.
[10] Md. T. Rahman and Md. M. Islam, "A comparison of machine learning algorithms to estimate effort in varying sized software," in *Proc. IEEE Region 10 Symp. (TENSYMP)*, Jun. 2019, pp. 137–142.
[11] L. L. Minku and X. Yao, "Ensembles and locality: Insight on improving software effort estimation," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1512–1528, Aug. 2013.
[12] E. Kocaguneli, T. Menzies, and J. W. Keung, "On the value of ensemble effort estimation," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1403–1416, Nov. 2012.
[13] D. Azhar, P. Riddle, E. Mendes, N. Mittas, and L. Angelis, "Using ensembles for web effort estimation," in *Proc. ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas.*, Oct. 2013, pp. 173–182.
[14] T. Kam Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.
[15] M. Fernández-Delgado, M. S. Sirsat, E. Cernadas, S. Alawadi, S. Barro, and M. Febrero-Bande, "An extensive experimental survey of regression methods," *Neural Netw.*, vol. 111, pp. 11–34, Mar. 2019.
[16] J. Huang, Y.-F. Li, and M. Xie, "An empirical analysis of data preprocessing for machine learning-based software cost estimation," *Inf. Softw. Technol.*, vol. 67, pp. 108–127, Nov. 2015.
[17] K. Strike, K. El Emam, and N. Madhavji, "Software cost estimation with incomplete data," *IEEE Trans. Softw. Eng.*, vol. 27, no. 10, pp. 890–908, Oct. 2001.
[18] D. Ruan, G. Chen, E. Kerre, and G. Wets, *Intelligent Data Mining: Techniques and Applications* (Studies in Computational Intelligence), vol. 5. Berlin, Germany: Springer-Verlag, Jan. 2010.
[19] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *J. Experim. Social Psychol.*, vol. 49, no. 4, pp. 764–766, Jul. 2013.
[20] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 33–53, Jan. 2007.
[21] P. Ardimento and C. Mele, "Using BERT to predict bug-fixing time," in *Proc. IEEE Conf. Evolving Adapt. Intell. Syst. (EAIS)*, May 2020, pp. 1–7.
[22] W. H. A. Al-Zubaidi, H. K. Dam, A. Ghose, and X. Li, "Multi-objective search-based approach to estimate issue resolution time," in *Proc. 13th Int. Conf. Predictive Models Data Analytics Softw. Eng.*, Nov. 2017, pp. 53–62.
[23] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion mmre," *IEEE Trans. Softw. Eng.*, vol. 29, no. 11, pp. 985–995, Nov. 2003.
[24] J. T. H. D. A. Cabral, A. L. I. Oliveira, and F. Q. B. da Silva, "Ensemble effort estimation: An updated and extended systematic literature review," *J. Syst. Softw.*, vol. 195, Jan. 2023, Art. no. 111542.

ANDRÉ O. SOUSA received the B.Sc. degree in information technology from the University of the Azores, and the M.Sc. degree in software engineering from the Faculty of Engineering, University of Porto, in 2021. His research interests include machine learning approaches to improve common tasks in the software development life cycle, such as risk management and effort estimation.

DANIEL T. VELOSO is currently pursuing the bachelor's degree in electrical and computer engineering with the Faculty of Engineering, University of Porto. His research interest includes computational intelligence to solve real-world problems in several domains.

HENRIQUE M. GONÇALVES received the M.Sc. degree in informatics and computing engineering from the Faculty of Engineering, University of Porto, in 2021. His research interests include machine learning in software project estimation and delivery forecasting.



JOÃO PASCOAL FARIA (Member, IEEE) is currently an Associate Professor of software engineering with the Faculty of Engineering, University of Porto, and a Senior Researcher with the Human-Centered Computing and Information Science Center, INESC TEC. His research interests include software engineering, namely software process improvement, software testing, and model-driven software engineering.

JOÃO MENDES-MOREIRA is currently an Associate Professor of data science with the Faculty of Engineering, University of Porto, and a Senior Researcher with the Artificial Intelligence and Decision Support Center, INESC TEC. His research interests include knowledge discovery, data mining, and machine learning.

RICARDO GRAÇA received the M.Sc. degree in informatics and computing engineering from the Faculty of Engineering, University of Porto, in 2013. He is currently an Android and Flutter Developer and a Researcher with Fraunhofer Portugal AICOS.

DUARTE GOMES received the M.Sc. degree in software engineering from the Faculty of Engineering, University of Porto, in 2018. He is currently a Project Manager and a Software Process Improvement Consultant with Strongstep–Innovation in Software Quality.

RUI NUNO CASTRO received the degree in electrical and computers engineering from the Faculty of Engineering, University of Porto, in 1996, and the M.Sc. degree in digital communications systems and technology from the Chalmers University of Technology, in 1999. He is currently a Research Management Group Leader with Fraunhofer Portugal AICOS, dealing with the management of public funded projects and is interested in the application of machine learning to project management.

PEDRO CASTRO HENRIQUES received the degree in informatics and computing engineering from the Faculty of Engineering, University of Porto, in 2005. He has been a Co-Founder and the CEO of Strongstep–Innovation in Software Quality, since 2009, where he leads several consulting and research and development projects.