

# Software Enhancement Effort Estimation using Stacking Ensemble Model within the Scrum Projects: A Proposed Web Interface

Zaineb Sakhrawi<sup>1</sup><sup>a</sup>, Asma Sellami<sup>2</sup><sup>b</sup> and Nadia Bouassida<sup>2</sup><sup>c</sup>

<sup>1</sup>University of Sfax, Faculty of Economics and Management of Sfax, Sfax, Tunisia

<sup>2</sup>University of Sfax, Higher Institute of Computer Science and Multimedia, Sfax, Tunisia

**Keywords:** Software Enhancement Effort Estimation, Functional Change, Functional Size, COSMIC FSM Method, Scrum, Stacking Ensemble Model, Web Application.


**Abstract:** The frequent changes in software projects may have an impact on the accuracy of the Software Enhancement Effort Estimation (SEEE) and hinder management of the software project. According to a survey on agile software estimation, the most common cost driver among effort estimation models is software size. Indeed, previous research works proved the effectiveness of the COSMIC Functional Size Measurement (FSM) method for efficiently measuring software functional size. It has been also observed that COSMIC sizing is an efficient standardized method for measuring not only software size but also the functional size of an enhancement that may occur during the scrum enhancement project. Intending to increase the SEEE accuracy the purpose of this paper is twofold. Firstly, it attempts to construct a stacking ensemble model. Secondly, it intends to develop a localhost web application to automate the SEEE process. The constructed stacking ensemble model takes the functional Size of an enhancement or a functional change, denoted as FS(FC), as a primary independent variable. The stacking ensemble model combines three Machine Learning (ML) techniques: Decision Tree Regression, Linear Support Vector Regression, and Random Forest Regression. Results show that the use of the FS(FC) as an input to SEEE using the stacking ensemble model provides significantly better results in terms of MAE (Mean Absolute Error) = 0.206, Mean Square Error (MSE) = 0.406, and Root Mean Square Error (RMSE) = 0.595.


## 1 INTRODUCTION


Effort estimation is one of the key activities supporting software project planning and management (Abran, 2015). However, the frequent changes throughout the software project may influence the accuracy of the effort estimates and hinder the project's success (Usman et al., 2018). Even when using the agile approach, time constraints and a lack of quantitative evidence frequently result in overestimations or underestimates, as well as faulty plans (Trzeciak, 2021). The agile software development life cycle (SLC) lacks a specifically planned mechanism for maintenance (Rehman et al., 2018). Software Maintenance is the field of Software Engineering which have been ignored over the last period. It has not received the same degree of attention that the other phases have

(Bourque and Fairley, 2014). Software maintenance activity has been classified into two categories: correction and enhancement (ISO/IEC, 2006). We will concentrate on the enhancement category in our research. Enhancement is defined as "a change made to an existing software product to meet a new requirement" (ISO/IEC, 2006).

Agile approach like Scrum embraces changes (Rehman et al., 2018). When managing Scrum software projects, providing an accurate estimation of an enhancement effort when requirements are likely to change is critical (Arora et al., 2020). There are numerous estimation techniques available, including expert opinion, Planning Poker (PP), and a few others (Vyas et al., 2018). According to a survey of five studies on basic estimation techniques (Vyas et al., 2018), the PP technique is the most commonly used in scrum projects. The basis of the PP is practitioners', which are expressed in the form of Story Points. In practice, it is used to estimate the effort required to complete software requirements or User Stories (US).

<sup>a</sup> <https://orcid.org/0000-0003-1052-3502>

<sup>b</sup> <https://orcid.org/0000-0002-6739-5508>

<sup>c</sup> <https://orcid.org/0000-0003-0434-2465>

The US is a short description of intent presenting user request (Desharnais et al., 2011). Besides PP, several international standards provide well-documented methods for measuring or approximating the US functional size, such as the COSMIC FSM method. There is a growing body of work on the use of the COSMIC function points (Ungan et al., 2014) for estimation and performance measurement of software development projects which can be adapted for predicting agile Software Enhancement effort too (Sakhrawi et al., 2021c). Rather than the increasing use of the COSMIC FSM method, our other motivation for this research study stems from the fact that existing single ML techniques used for SEEE have several limitations (Wang et al., 2021) while other innovative models, such as the ensemble model, have yet to be adopted in the industry for estimating software effort. Indeed, recent research publication (Sakhrawi et al., 2021a) has investigated the use of ensemble learning for improving SEEE for traditional projects (*i.e.*, using waterfall methodology) compared to individual models.

In accordance with the findings obtained in (Sakhrawi et al., 2021a), our research paper’s goal is to improve the accuracy of SEEE for scrum projects by building a stacking ensemble model using the FS (FC) as an independent variable. Our constructed stacking ensemble model combines three different ML techniques: Decision Tree Regression, Linear Support Vector Regression, and Random Forest Regression (DTRegr, LinearSVR, and RFR). The selected three ML techniques are recently used for SEEE and proved as accurate models for SEEE in both traditional and scrum contexts (Sakhrawi et al., 2021c). Estimation results using the stacking ensemble model are compared to those using the three ML techniques (DTRegr, LinearSVR, and RFR) separately. Although the stacking ensemble model produces significant results, using it manually is time-consuming. Of course, manual solutions are not practical. For this reason, we propose developing a local web application called "ERWebApp" to automate the SEEE process.

The rest of this paper is organized as follows. Section 2 provides an overview of the key concepts that will be used. Section 3 discusses the related work. Section 4 provides a detailed description of our research work process. Section 5 discusses the experimental results. In section 6, we present the evaluation performed throughout the threats to validity. Finally, in section 7, we summarize the presented work and outlines some of its possible extensions.

## 2 BACKGROUND

This section describes an overview of the COSMIC FSM method, the PP technique, and the stacking ensemble model.

### 2.1 COSMIC

The Common Software Measurement International Consortium (COSMIC) is an FSM method proposed in 1999 to overcome some limitations of FPA (Martino et al., 2020). COSMIC is used for measuring software functionality. The method is designed to be independent of any implementation decisions embedded in the operational artifacts of the software to be measured. The process for measuring the COSMIC functional requirement size is composed of three phases: Measurement strategy phase, mapping phase, and Measurement phase (Berardi et al., 2011). Using COSMIC FSM a functional process is composed of a set of functional sub-processes that may be either a data movement or a data manipulation. There are four types of data movements: Entry (E), eXit (X), Read (R), and Write (W).

- An Entry moves a data group into a functional process from a functional user.
- An eXit moves a data group out of a functional process to a functional user.
- A Write moves a data group from a functional process to persistent storage.
- A Read moves a data group from persistent storage to a functional process.

A data group is a set of attributes that describes one object of interest. The COSMIC measurement unit is one data movement of one data group indicated as one CFP (COSMIC Function Point). The size of a functional process is determined by the sum of the data movements it includes. The functional size of a functional process (noted by FS (FP)) is given by Equation 1 (Berardi et al., 2011).

$$FS(FP) = \sum FS(Entries) + \sum FS(eXits) + \sum FS(Reads) + \sum FS(Writes) \quad (1)$$

The size of a change (an enhancement) presenting a functional process is the sum of its data movements that have been added, deleted, and modified. The software functional size after the change is the sum of the sizes of all the added data movements minus the size of all the removed data movements.

## 2.2 Overview of Planning Poker

The Planning Poker technique was initially proposed by Grenning (Grenning, 2002) and popularized by Cohn (Haugen, 2006) where agile software effort is predicted in terms of units of work referred to as 'story points' quantifying the implemented US (Vyas et al., 2018). The US is a requirement presented in a specific way to highlight the type of user as well as the goal or functionality that the user needs to perform in order to obtain some benefits (Haugen, 2006). The main objective of PP was to build more effective prediction sessions and to get more implications from the whole industry project team (Haugen, 2006). Even though the PP is the most widely used technique to estimate software development effort, it was criticized in the same way as expert judgment. In addition, it does not provide direct information on the software size to be delivered.

## 2.3 Stacking Ensemble Model

The stacking model is invented by Wolpert (Wolpert, 1992). It is recently used for estimating software effort (Sampath Kumar and Venkatesan, 2021)(Priya et al., 2021). The stacking model combines lower-level ML techniques for achieving more accurate estimation. The constructed linear estimation model consists of two learning levels (Kraipeerapun and Amornsamankul, 2015). The first learning level is called Level-0, where models are trained and tested in independent cross-validation examples from the original input data. Then, the output of Level-0 and the original input data is used as input for level-1, called generalized (*i.e* the meta-model). Level-1 is constructed using the original input data and the output of level-0 generalizers (Kraipeerapun and Amornsamankul, 2015).

## 3 RELATED WORK

Recently, research studies have reported that the COSMIC FSM method has been successfully used in agile software projects (Sakhrawi et al., 2021c). The use of the COSMIC FSM method increases the quality of the documentation in agile projects (Ungan et al., 2014). Compared to the scrum story points, the estimation model built using the COSMIC FSM method provides more accurate results for estimating development effort (Ungan et al., 2014)(Sakhrawi et al., 2021c). Hence, the use of software functional size in COSMIC Functional Points (CFP) units as independent variables to build an effort estimation

model give more accurate results than the use of Story Points as independent variables (Sakhrawi et al., 2021c).

More recently, the use of the ensemble model, combining more than one single ML technique, has achieved attention in the software engineering research (Sampath Kumar and Venkatesan, 2021; Sakhrawi et al., 2021a). In addition, a systematic review conducted by (Idri et al., 2016) has confirmed that the ensemble methods outperformed their constituents (single models). In the same context, (Sakhrawi et al., 2021a) proposed a stacking ensemble model that has revealed promising capabilities in improving the accuracy over single models for traditional SEEE. Regarding a systematic mapping study on the use of ML techniques for SEEE (Sakhrawi et al., 2021b), this is the first study to our knowledge that investigates the use of the ensemble method in software maintenance (enhancement) effort estimation within the scrum context.

## 4 RESEARCH PROCESS

Figure 1 presents our proposed research work process. We compare the SEEE accuracy of the selected ML techniques when used (trained and tested) separately with the SEEE accuracy of the constructed stacking ensemble model. The stacking ensemble model is then integrated into a local web application (ERWebApp) to help estimators (*e.g.*, development team) make SEEE automatically.

### 4.1 Data Collection Data

In this section, we will use real projects developed using the scrum framework and derived from industry (Sakhrawi et al., 2021c). The chosen dataset is an example of Software Enhancement in which the functional specifications are considered an enhancement. Because our work focused on enhancement, we use the following filters to eliminate trivial projects: Actual Development Time (man-days), full life cycle effort for a project exceeding 80 man-hours, and "Stats" other than "implement" were excluded. When assessing the performance of an estimation model, the choice of a suitable one is based on the quality of its inputs, data sets, and, most importantly, the use of international standards (Abran, 2015). Indeed, since the FS (FC) gives accurate results (Sakhrawi et al., 2021c), we propose to measure the FS of all US in the dataset. Then, we add the new measurements to the existing dataset attributes. The enhancement FS in the dataset (that is expressed in the form of US) was

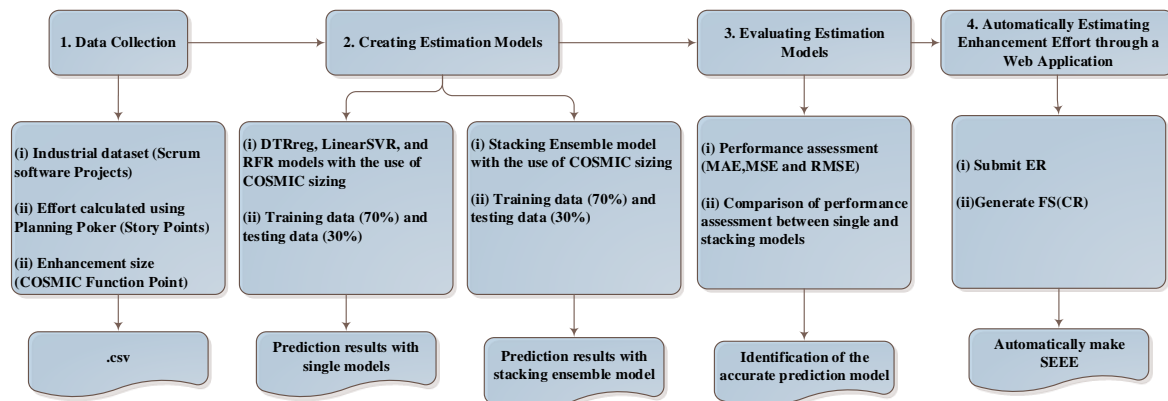


Figure 1: Research Work Process.

measured using the COSMIC method. The COSMIC FSM process includes three phases (Berardi et al., 2011): the measurement strategy, the mapping of concepts, and the measurement of the identified concepts.

**Measurement Strategy Phase.** The Measurement Strategy phase defines what will be measured (the purpose of measurement). And therefore, the scope to be measured is determined. The strategy phase is important to ensure that we measure what is required to be measured. The following are the main parameters that must be identified during this phase:

- The purpose: Estimating Effort for implementing an enhancement (*i.e.*, a Functional Change).
- Overall scope: Generating the Functional Change (FC) size and estimating the associated enhancement effort.
- Functional users: are human users in this case.
- Layer: An industrial dataset (eight sprints).
- Level of granularity: one level of granularity

**Mapping of Concepts.** An Enhancement within scrum context is presented by US. A US exhibits a high-level requirement description. There is no general standard US representation (Sellami et al., 2018). The level of granularity for sizing requirements (or enhancements) in the form of a US must be that of the COSMIC functional processes (Berardi et al., 2011). For that reason, the mapping of a US to a COSMIC Functional Process necessitates distinguish the identification of following concepts:

As an <Actor>: represents the user of the US referred to as the functional user in COSMIC.  
I want to <Goal>: represents the

enhancement request referred to as US or a Functional Process (FP) in COSMIC.  
so that <value or expected benefit>

Regarding Table 1, the level of granularity for sizing an enhancement in the form of a US must be that of the COSMIC FP (Berardi et al., 2011). FC are mainly classified into three types: add (new requirements to be created), delete (existing requirements to be deleted), and modify (existing requirements to be modified) (Sakhrawi et al., 2019).

**Measurement Phase.** In the Measurement phase, the COSMIC Functional Process (FP) consists of a set of functional sub-processes. The FP size is the number of data movements (DM) it includes and the software size is the sum of the sizes of all its FPs. According to (Berardi et al., 2011), the DM can be identified based on some common word cases (such as create, select, delete, add, share, display, etc.) (Sakhrawi et al., 2019). Using the identified DM types that are repeatedly executed by the user can facilitate the measurement process. Table 2 explained the ability of the COSMIC FSM method on detailing the Functional change process (from process to sub-process), which exposes more iterations and therefore more data movements.

## 4.2 Constructing Estimation Models

We have constructed two estimation models presented in Figure 1. The first model constructs three ML techniques (DTRegr, LinearSVR, and RFR) for SEEE **separately**. The second model constructs a stacking ensemble model (that **combines** the three selected ML techniques). For the models set of experiments, we use the classic approach with a simple 70%(training)-30% (validation/test) split. Exper-

Table 1: Example: mapping of US with COSMIC FC.

US Id	User Story description			COSMIC Functional Change description	
	As ... (actor)	I can ... (Goal)	Effort in Story Points	Change Type (add, delete, or modify)	FC description
1	Organization User	Add a custom evidence type to an assessment criterion (because the standard evidence types are not appropriate for me)	4	ADD()	Add Custom Evidence Type

Table 2: Sizing an enhancement “Add Custom Evidence Type” in CFP units.

FP Description	Functional User	Description of Sub-FP	Data Group	Object of interest	DM type	CFP
Add Custom Evidence Type	Organisation User	Request to view a specific widget	Custom Evidence type	Custom Evidence type	Entry	1
		Add a “custom” evidence type against a specific attainment criterion	Custom Evidence Type	Custom Evidence type	-	-
	Organisation User	Add a custom evidence type and give him a name (free form text)	Custom Evidence Type	Custom Evidence type	Entry	1
		Verify changes	Custom Evidence Type	Custom Evidence type	Read	1
		Save changes	Custom Evidence Type	Custom Evidence Type	Write	1
		The custom evidence type is saved	Custom Evidence Type	Custom Evidence type	eXit	1
	The custom evidence type is saved	Custom Evidence Type	Custom Evidence type	eXit	1	
						Total

iments were performed using the Google Colaboratory<sup>1</sup> python programming. The most recent inventory tool is Google Colaboratory, also known as Google Colab. It makes GPUs available to researchers who lack resources or cannot afford one.

### 4.3 Evaluating Estimation Models

To evaluate the accuracy of the estimation models, we used three widely used evaluation metrics (Idri et al., 2015): the mean absolute error (MAE), the mean square error (MSE), and the root mean square error (RMSE).

<sup>1</sup><https://colab.research.google.com/notebooks/intro.ipynb?recent=true>

#### 4.3.1 LinearSVR Algorithm Performance Assessment versus DTRegr, and RFR Models

Regarding Table 3, error metrics (such as MAE, MSE and RMSE) reveal quite values using linearSVR (MAE=0.240 ; MSEs= 0.923 ; RMSE=0.646).

Table 3: Estimation analysis using MAE, MSE and RMSE.

Method/ parameters	MAE	MSE	RMSE
LinearSVR	0.240	0.923	0.646
DTRegr	0.5	1.0	1.0
RFR	0.952	1.479	1.216

### 4.3.2 Stacking Ensemble Model

When constructing a stacking ensemble model, it is critical to select which ML techniques will be used as "estimators". The meta-model provided via the "final\_estimator" argument (DTRegr) is trained to combine the estimation of the selected ML techniques provided via the "estimators" argument (LinearSVR, RFR). Each model is trained on the constructed dataset using the FS(FC) as a primary independent variable. Then, the outputs of "estimators" are fed into the "final\_estimator" which combines each estimator model with a weight and delivers the final estimation.

**Selecting Estimators and final\_estimator.** The main parameters of the stacking ensemble regression model are defined in scikit-learn<sup>2</sup> as follows: StackingRegressor(estimators, final\_estimator=None, \*) explained in Table 4. Thus, we try to identify which

Table 4: Stacking ensemble regression model parameters<sup>7</sup>.

Parameters	Description
estimators	Base estimators which will be stacked together.
final_estimator	An estimator which will be used to combine the base estimators

technique from the three ML techniques can be used as "final\_estimator" and which ones should be used as "estimators". Table 5 illustrates the estimation analysis results using MAE, MSE, and RMSE. Error metrics reveal quite values when the DTRegr algorithm is used as the final\_estimator (MAE=0.206 ; MSE=0.406; RMSE=0.595). Figure 2 shows the ML "estimators" and the average of their estimations.

Table 5: Estimation analysis using MAE, RMSE.

estimators	final estimator	MAE	MSE	RMSE
LinearSVR and RFR	DTRreg	0.206	0.406	0.595
LinearSVR and DTR-reg	RFR	0.788	0.922	0.960
DTRreg and RFR	Linear_SVR	0.761	1.250	1.118

<sup>2</sup><https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble>

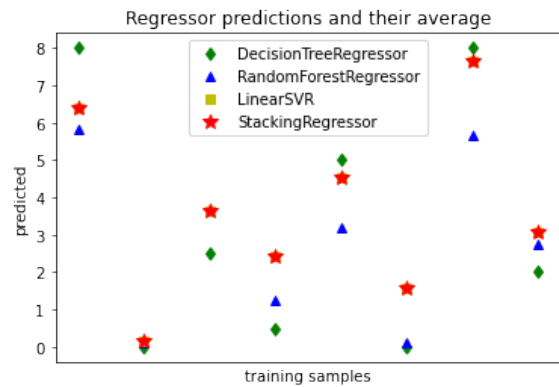


Figure 2: ML "estimators" and the average of their estimations.

**Make SEEE.** Constructing SEEE, each ML regression technique is trained on the selected dataset. The outputs of "estimators" are therefore fed into the "final\_estimator" which combines each regression estimator model with a weight and delivers the final estimation. To evaluate the overall performance of the constructed estimation model, we selected the r2\_score evaluation metric<sup>3</sup>. Figure 3 depicts the r2\_score results, with the best possible score of 1.0. The results show that the stacking ensemble model outperforms the other three ML techniques. The rr2\_score is 0.789.

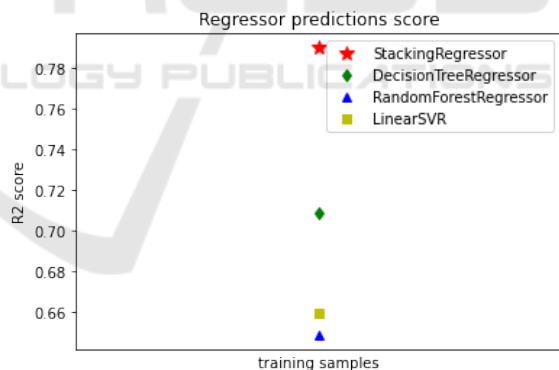


Figure 3: ML techniques Performance Assessment.

## 4.4 A Proposed Web Interface for SEEE

Regarding experimental results, we found that using the stacking ensemble model is the most accurate model. However, using this model manually is time-consuming. Of course, manual solutions are not practical. For this reason, we propose to develop an ERWebApp to rapidly and automatically make SEEE. The ERWebApp is designed to first generate the en-

<sup>3</sup>[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2\\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score)

hancement functional size and then estimate its corresponding effort.

The ERWebApp is developed using Bootstrap<sup>4</sup>, Anvil Platform<sup>5</sup>, and Python<sup>6</sup>. Python is used in the backend to create the estimation model that maps the input and output data based on the stacking ensemble model, while Anvil's Platform and Bootstrap are used in the frontend to display content. In order to transfer content between web applications and the estimation model, the use of the Anvil platform appeared to be beneficial. It is used to help in the visualization of the estimation model by creating and hosting the estimation web page, which is entirely written in Python using predictable and minimal resources (CPU, memory, threads). The ERWebApp is styled using Bootstrap. It is hosted on the user's computer's localhost and can be accessed using web browsers from any operating system.

#### 4.4.1 ERWebApp Users

The ERWebApp is designed to meet the needs of the three Scrum roles: product owner, scrum master, and development team members. It will enable estimators to express the ER in the form of US in the natural language. Then, the functional size of the US will be generated in terms of CFP units. The estimator then receives the estimated effort based on ontology and ML techniques. This implies that the three scrum roles would be able to perform the following actions in the web interface:

1. The Scrum Master: is responsible for the revised planning and deciding on the execution of the ER.
2. The Product Owner: Submit the ER description.
3. The Development Team: reformulate the ER in three steps: (1) specify a formal description of the ER, (2) generate the functional size of the ER in CFP units, and (3) estimate the effort required to implement the ER.

The principal parts of the ERWebApp are described throughout three pages of interfaces: a projects overview (for scrum master), a submit ER form (for product owner), and regulate ER form (for the development team). Focusing on making the ERWebApp easier to use, we created three sessions for the three scrum roles. Indeed, we created three login sessions/profiles for the three roles. Figure 4 depicts an example of the product owner's login session page.

<sup>4</sup><https://getbootstrap.com/>

<sup>5</sup><https://anvil.works/open-source>

<sup>6</sup><https://www.python.org/>

Figure 4: Login page of product owner.

#### 4.4.2 Product Owner Interface: Submit Enhancement Request

When a new enhancement occurs in an existing project, the product owner must submit the enhancement description. As shown in Figure 5, the development team is responsible for implementing enhancement. After the ER has been approved, a detailed description can be created (in other words, going from an informal to a formal ER description). The Product Owner can express and submit an enhancement request in natural language using this interface page.

Figure 5: Submit enhancement request.

#### 4.4.3 Development Team Interface

We concentrated on the development team session because that is the team in charge of managing and implementing the enhancement. The development team will provide the COSMIC sizing of an enhancement as well as an estimate of the corresponding effort.

Figure 6 depicts the output via the user interface.

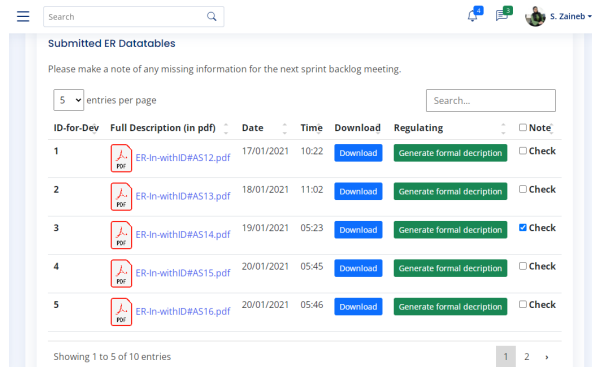


Figure 6: Development team interface.

**Enhancement Request Details.** The web interface page of the development team includes the enhancement details. Figure 6 contains two buttons: the blue button downloads the enhancement description validated by the product owner, and the green button generates a formal explanation of a specific enhancement request. Figure 7 shows a full formal description of a selected ER. The functional size of the specified ER is also represented in terms of CFP units. A button was created to demonstrate how the functional size of an enhancement, denoted as FS (FC), can be found and used to estimate the effort required to complete this enhancement. When the button "click to reveal in detail" in Figure 7 is selected, the details of FS (FC) are generated as shown in Figure 8 (details of functional size measurement are provided in Measurement phase paragraph in section 4.1).

**Estimating Enhancement Request Effort.** As shown in Figure 9 using the Anvil platform, we created a web page for SEEE. Through the Anvil platform, we can share a private link to the web page. Then, in the Bootstrap template, we included this link to be used by the development team. The Anvil platform is also used to connect the model built-in Google Collab to the input variable, which is the functional size of the enhancement request. The SEEE web page interface includes two sessions: the train session constructed on the backend with the Google Collab and the estimation session designed

Output of the "Generate Formal description" Button

#	As Actor	I want to	so that (optional)	Type	Functional Process	Functional Size	Measuring Details
2	As Organization User	I want to add a custom evidence type to an assessment criterion	so that the standard evidence types become appropriate for me	Add()	Add Custom Evidence Type	6 CFP	Click to show in detail

Figure 7: Enhancement request details.

Home / Customer Enhancement Request Records / ER Functional Size Records

Functional Size of ER (with COSMIC)

#	Functional User	Description of Sub-FP	Data movement type	CFP
1	Organization User	Request to view a specific widget	Entry	1
	System	Add a "custom" evidence type against a specific attainment criterion	-	-
	Organization User	Add a custom evidence type and give it a name (free form text)	Entry	1
	System	Verify changes	Read	1
	System	Save change	Write	1
	System	The custom evidence type is saved	eXit	1
	Organization User	The custom evidence type is saved	eXit	1
				6

Figure 8: Regulate enhancement request page.

for the development team with the Anvil platform. As shown in Figure 9, the development team needs to select the model to estimate the effort of the enhancement request. The stacking ensemble model-based SEEE must be uploaded by the developer. In addition, the development team must provide the functional size (approximated) of enhancement (FC) as input in CFP. As a result, our ERWebApp can estimate the Software Enhancement effort.

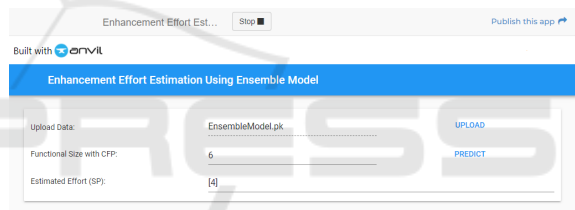


Figure 9: Estimating enhancement request effort.

## 5 DISCUSSION AND COMPARISON

Recall that our proposed research process goal is to investigate (i) the importance of using the enhancement FS as a primary independent variable and (ii) the use of the stacking ensemble model for improving the accuracy of SEEE within the scrum context. We firstly conducted three selected ML techniques (DTRegr, LinearSVR, and RFR) separately. Experimental results show that LinearSVR is the most effective model. This is supported by the results, which show a minimum MAE of 0.240. To ensure the above results, we have investigated the idea of using a stacking ensemble model by combining the three ML techniques (DTRegr, LinearSVR, and RFR). Experimental results are compared with the three ML techniques when constructed separately. The effectiveness of the stacking ensemble model can be seen in the results (see Figure 2 and Figure 3). This is supported by the results with the minimum MAE of 0.206, MSE



of 0.406 and RMSE of 0.595, and a good  $r^2$ \_score of 0.789.

## 6 THREATS TO VALIDITY

The validity of this research results is pertinent to internal validity and external validity.

- Internal threats to validity are related to the size of the data set where the number of instances in the data set must be more significant. This work is limited to the numerical attributes of the Software Enhancement datasets for scrum projects, although many historical scrum project datasets contain categorical attributes.
- External threats to validity are proportional to the degree to which the study's findings can be applied to other projects. We only used one private scrum dataset. However, we believe that our proposed research study can be applied to a variety of project datasets.

## 7 CONCLUSION

In this study, we investigated the problem of accurate estimation of effort for software scrum enhancement projects. Three single ML techniques and stacking models were implemented and empirically tested. Based on the fact that COSMIC sizing is a powerful FSM method, it was used as an input feature for predicting enhancement effort, the enhancement FS is used as an independent variable. The following are the results of the experiments:

- LinearSVR is more accurate with small MAEs= 0.240, MSE= 0.923, and RMSE= 0.646 compared to DTRreg and RFR.
- The stacking ensemble model is more accurate with MAE of 0.206, MSE of 0.406 and RMSE of 0.595, and a good  $r^2$ \_score of 0.789 compared to the selected single ML techniques SEEE accuracy.

Then, we conclude our research work process by building a localhost ERWebApp. The web application's goal is to evolve into a company that can estimate the effort of a new enhancement request from FS in a scrum context. We did not place a high value on the roles of scrum master and product owner because the estimation process is handled by the development team. However, it will be improved in the future.

## REFERENCES

- Abran, A. (2015). *Software project estimation: the fundamentals for providing high quality information to decision makers*. John Wiley & Sons.
- Arora, M., Verma, S., Chopra, S., et al. (2020). A systematic literature review of machine learning estimation approaches in scrum projects. *Cognitive Informatics and Soft Computing*, pages 573–586.
- Berardi, E., Buglione, L., Cuadrado-Collego, J., Desharnais, J.-M., Gencel, C., Lesterhuis, A., Santillo, L., Symons, C., and Trudel, S. (2011). Guideline for the use of cosmic fsm to manage agile projects. *Common Software Measurement International Consortium*.
- Bourque, P. and Fairley, R. E. (2014). Guide to the software engineering-body of knowledge. *Online in Internet: URL: http://www.swebok.org [Stand: 12.01. 2005]*.
- Desharnais, J.-M., Buglione, L., and Kocaturk, B. (2011). Using the cosmic method to estimate agile user stories. In *Proceedings of the 12th international conference on product focused software development and process improvement*, pages 68–73.
- Grenning, J. (2002). Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting*, 3:22–23.
- Haugen, N. C. (2006). An empirical study of using planning poker for user story estimation. In *AGILE 2006 (AGILE'06)*, pages 9–pp. IEEE.
- Idri, A., azzahra Amzal, F., and Abran, A. (2015). Analogy-based software development effort estimation: A systematic mapping and review. *Information and Software Technology*, 58:206–230.
- Idri, A., Hosni, M., and Abran, A. (2016). Systematic literature review of ensemble effort estimation. *Journal of Systems and Software*, 118:151–175.
- ISO/IEC (2006). International standard—iso/iec 14764 iec std 14764-2006 software engineering; software life cycle processes &; maintenance.
- Kraipeerapun, P. and Amornsamankul, S. (2015). Using stacked generalization and complementary neural networks to predict parkinson's disease. In *2015 11th International Conference on Natural Computation (ICNC)*, pages 1290–1294. IEEE.
- Martino, S., Ferrucci, F., Gravino, C., and Sarro, F. (2020). Assessing the effectiveness of approximate functional sizing approaches for effort estimation. *Information and Software Technology*, 123:106308.
- Priya, A. V., Varadarajan, V., et al. (2021). Estimating software development efforts using a random forest-based stacked ensemble approach. *Electronics*, 10(10):1195.
- Rehman, F., Maqbool, B., Riaz, M. Q., Qamar, U., and Abbas, M. (2018). Scrum software maintenance model: Efficient software maintenance in agile methodology. In *2018 21st Saudi computer society national computer conference (NCC)*, pages 1–5. IEEE.
- Sakhrawi, Z., Sellami, A., and Bouassida, N. (2019). Requirements change requests classification: an ontology-based approach. In *International Conference on Intelligent Systems Design and Applications*, pages 487–496. Springer.

- Sakhrawi, Z., Sellami, A., and Bouassida, N. (2021a). Software enhancement effort estimation using correlation-based feature selection and stacking ensemble method. *Cluster Computing*, pages 1–14.
- Sakhrawi, Z., Sellami, A., and Bouassida, N. (2021b). Software enhancement effort prediction using machine-learning techniques: A systematic mapping study. *SN Computer Science*, 2(6):1–15.
- Sakhrawi, Z., Sellami, A., and Bouassida, N. (2021c). Support vector regression for enhancement effort prediction of scrum projects from cosmic functional size. *Innovations in Systems and Software Engineering*, pages 1–17.
- Sampath Kumar, P. and Venkatesan, R. (2021). Improving accuracy of software estimation using stacking ensemble method. In *Advances in Machine Learning and Computational Intelligence*, pages 219–227. Springer.
- Sellami, A., Haoues, M., Borchani, N., and Bouassida, N. (2018). Towards an assessment tool for controlling functional changes in scrum process. In *IWSM-Mensura*, pages 34–47.
- Trzeciak, M. (2021). Sustainable risk management in it enterprises. *Risks*, 9(7):135.
- Ungan, E., Cizmeli, N., and Demirörs, O. (2014). Comparison of functional size based estimation and story points, based on effort estimation effectiveness in scrum projects. In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 77–80. IEEE.
- Usman, M., Britto, R., Damm, L.-O., and Börstler, J. (2018). Effort estimation in large-scale software development: An industrial case study. *Information and Software technology*, 99:21–40.
- Vyas, M., Bohra, A., Lamba, C., and Vyas, A. (2018). A review on software cost and effort estimation techniques for agile development process. *International Journal of Recent Research Aspects*, 5(1):1–5.
- Wang, L., Zhu, Z., Sassoubre, L., Yu, G., Liao, C., Hu, Q., and Wang, Y. (2021). Improving the robustness of beach water quality modeling using an ensemble machine learning approach. *Science of The Total Environment*, 765:142760.
- Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.