IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 2 September 2024, accepted 23 September 2024, date of publication 1 October 2024, date of current version 9 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3471428

RESEARCH ARTICLE

Using Machine Learning and Simplified Functional Measures to Estimate Software Development Effort

LUIGI LAVAZZA[1], (Senior Member, IEEE), ANGELA LOCORO[2], AND ROBERTO MELI[3]
[1]Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria, 21100 Varese, Italy
[2]Dipartimento di Economia e Management, Università degli Studi di Brescia, 25100 Brescia, Italy
[3]DPO—Data Processing Organization, 00100 Rome, Italy

Corresponding author: Luigi Lavazza (luigi.lavazza@uninsubria.it)

This work was supported in part by the "Fondo di Ricerca d'Ateneo" funded by the Università degli Studi dell'Insubria.

ABSTRACT Functional size measures are often used as the basis for estimating development effort, because they are available in the early stages of software development. Several simplified measurement methods have also been proposed, both to decrease the cost of measurement and to make functional size measurement applicable when functional user requirements are not yet known in full detail. Lately, machine learning techniques have been successfully used for software development effort estimation, but the usage of machine learning techniques in combination with simplified functional size measures has not yet been empirically evaluated. This paper aims to fill this gap: it reports to what extent functional size measures can be simplified, without decreasing the accuracy of effort estimates obtained via machine learning techniques. The reported evaluation addresses separately the effort models concerning (i) new software developed from scratch, (ii) software extensions obtained by adding new functionality, and (iii) software modifications that required also changing and possibly removing functionalities. We carried out an empirical study, in which effort estimation models were built via multiple Machine Learning techniques, using both traditional full-fledged functional size measures and simplified measures. Our study shows that using simplified functional size measures in place of traditional functional size measures for effort estimation does not yield practically relevant differences in accuracy. Therefore, software project managers can consider analyzing only a small and specific part of functional user requirements to get measures that effectively support effort estimation.

INDEX TERMS Software effort estimation, functional size measurement, function point analysis, machine learning.

I. INTRODUCTION

Function Point Analysis (FPA) was introduced to yield a measure of software size based exclusively on functional requirements specifications [1]. Accordingly, functional size measures (FSMs) are widely used for estimating software development effort, mainly because they are available in the early stages of development, when effort estimates are most needed.

However, deriving FSMs requires that complete and detailed specifications are available; in addition, the measurement takes a relatively long time and requires highly qualified measurers. For all these reasons, a few "simplified" measures have been proposed. These measures are simpler and quicker, and applicable when fully detailed software specifications are not yet available. Among the simplified measures are Simple Function Points (SFP) [2] (formerly known as SiFP [3]) and the sheer number of transaction functions.

Simplified measures consider a smaller amount of information than traditional Function Points, hence they may be used to approximate traditional measures of functional

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

size. Previous research has shown that this approximation is relatively accurate [4]. However, in principle, even small inaccuracies in the measurement of functional size could negatively affect the accuracy of effort estimation. Some evidence against this hypothesis has been produced, but only when effort models are built using traditional statistical instruments [5], [6]. In addition, such evidence was incomplete, since it did not cover projects aiming to extend existing software.

In the recent past, machine learning (ML) has been increasingly used to successfully build effort models [7], [8], [9], [10], [11], [12], and is now a sort of *de-facto* standard technique for building effort estimation models.

The ever-increasing importance of machine learning techniques in software effort estimation challenges the conclusions of previous work concerning simplified FSMs: when using ML for effort estimation, is it still true that simplified FSMs can be used in place of full-fledged measures without any practically relevant loss of accuracy? Is this true for all types of projects (i.e., new developments, enhancements and extensions)?

In this paper, we address these problems, by means of an empirical study. Specifically, we address the following research questions:

RQ1 Is it possible to build ML models that are able to predict development effort using just a subset of the information needed to compute traditional full-fledged functional size measure?

RQ2 In case RQ1 provides a positive answer, is there a ML technique that appears to provide more accurate estimates than other techniques?

RQ3 In case RQ1 provides a positive answer, are there models (involving a specific ML technique and a simplified measure) that yield effort estimates as accurate as the estimates obtained from traditional full-fledged functional size measures?

RQ4 What is, if any, the most basic (i.e., most simplified) functional measure that supports estimation with acceptable accuracy, with respect to estimates obtained from traditional full-fledged functional size measures?

The study is based on the analysis of the ISBSG dataset [13], which has been widely used for studies concerning software functional size.

The main contribution of the paper does not consist only in studying the joint application of ML methods and simplified FSMs to build effort estimation models. In fact, the presented study evaluates the usage of FSMs that require different levels of knowledge of requirements specification: hence, the available FSMs are are not only evaluated, but also compared to each other, with respect to their ability to support accurate effort estimation. This comparison has a great practical importance for software project managers, who need to know how to build development effort estimation models having acceptable prediction accuracy in the early development stages.

The results of our empirical study show that—for all types of projects (new developments, extensions and enhancements)—Support Vector Regression appears to provide the most accurate estimates, and different types of metrics yield extremely similar effort estimation errors. These results support the idea that FSMs can be used in practice for effort estimation instead of full-fledged metrics, with no penalties with respect to accuracy. Since simplified FSMs are easier and cheaper to collect, and are available earlier than full-fledged FSMs, our results are potentially of great interest for project managers who need estimates of development effort even before functional requirements have been described in full detail.

The remainder of this paper is organized as follows. Section II provides some background on functional size measurement. The empirical study is described in Section III and its results are illustrated in Section IV. In Section V we answer the Research Questions. The threats to the validity of the empirical study are discussed in Section VI. Section VII accounts for related work. Section VIII illustrates the conclusions and outlines future work. Appendix provides further details about building prediction models via ML.

II. BACKGROUND

To make the paper as self-contained as possible, this section provides a brief introduction to functional size measurement.

A. FUNCTION POINT ANALYSIS

Function Point Analysis was conceived to measure the size of software systems from the end-users' point of view, with the goal of providing a basis for estimating the development effort [1].

The official documentation of FPA [15] is maintained by the IFPUG (International Function Points User Group): readers are referred to such documentation for further details, guidelines and examples.

Currently, the IFPUG promotes two FSMs: Function Points (from the original proposal by Albrecht [1]) and Simple Function Points (the simplified measure proposed by Meli [3]).

The basic idea of FPA is that the ''amount of functionality'' made available to users can be quantified by taking into account two aspects of software: data and 'transactions' (i.e., operations that involve data exchanged through the boundaries of the application). The data and transactions that are considered for measurement are those that are relevant to the user, i.e., no implementation-dependent factors are taken into account. In fact, IFPUG Function Points are counted on the basis of functional user requirements (FURs) specifications.

FURs are modeled as a set of measurable elements, named base functional components (BFCs). IFPUG BFCs are data functions (or 'logical files') and transaction functions. Data functions are distinguished in internal logical files (ILF), which are managed within the application, and external interface files (EIF), which are created and maintained

outside the considered application. Transaction functions are classified, based on some characteristics of the activities carried out within the transaction, into external outputs (EO), external inquiries (EQ), and external inputs (EI).

The functional size measured in unadjusted Function Points is the sum of the sizes of data and transaction functions. The size of each data function or transaction function is determined by its complexity, which is classified as high, medium or low. Readers interested in the evaluation of complexity are referred to the official documentation [15]. For our purposes, it is sufficient to know that the exact determination of complexity according to FPA requires a rather deep knowledge of functions. Specifically, complexity of data functions requires that data are specified at the level of detail provided by Entity-Relationship diagrams or UML class diagrams; complexity of transaction functions requires that transactions are specified at the level of detail of data-flow diagrams of UML sequence diagrams [16].

The core of FPA involves the following main activities:

1) Identifying data functions and transaction functions.
2) Classifying data functions as ILF or EIF and transactions functions as EO, EQ, or EI.
3) Evaluating the complexity of data functions and transaction functions.

The first two of these activities can be carried out even if the FURs have not yet been fully detailed. On the contrary, the last one requires a detailed specification of FURs. Simplified functional size measurement methods aim at providing estimates of FSMs by skipping one or more of the activities listed above. Specifically, simplified measurement methods tend to skip at least the evaluation of the complexity of functions, since these activities are time- and effort-consuming [17].

B. THE SIMPLE FUNCTION POINT METHOD

The Simple Function Point measurement method [2], [3] has been designed by Meli and subsequently evolved by IFPUG to be lightweight and easy to use. Like IFPUG FPA, it yields a size measure that does not depend on design, implementation or technology-related characteristics of the measured software.

SFP requires only the identification of Logical Files (LF) and Elementary Processes (EP), i.e., unclassified transactions, the basic assumption being that the size of a BFC does not account for details concerning its internal organization. As a consequence, measurement is quicker and easier, and can be performed before the analysis of requirements provides a completely detailed description of FURs.

SFP assigns a numeric value directly to BFCs, as follows:

$$Size_{SFP} = 7\ \#LF + 4.6\ \#EP$$

The weights for each BFC (i.e., 7 for #LF and 4.6 for #EP) were originally defined to achieve the best possible approximation of FPA. However, since SFP is a measurement method, those weights are constants, i.e., they are not subject to update or change for better approximating measures in Function Points. Instead, weights are now crystallized, thus making measures expressed in SFP stable, comparable and repeatable.

It is worth noting that #EP is the number of transactions, i.e., #EI+#EO+ #EQ, which is denoted as #TF in the following sections. Therefore, the transactional part of SFP differs from #TF only by the constant 4.6. Hence, using #TF for building effort prediction models is equivalent to using the transactional part of SFP.

As a final observation, functional transactions are defined identically in IFPUG FPA and SFP: accordingly, the identification and usage of #TF (alias #EP) exploits the existing standard methodologies and rules.

C. TRADITIONAL APPROACHES TO EFFORT ESTIMATION

1) COCOMO

The Constructive Cost Model (COCOMO) is one of the first and best known software effort estimation models. It was developed by Boehm in 1981 [23], by fitting a regression formula using historical data from 63 projects. The basic model relates Effort (expressed in PersonMonths) and Size (expressed in KLoC, i.e., thousands lines of code) as follows:

$$Effort = a\ (Size)^b$$

where the values of a and b depend on the type of project (organic, semi-detached, embedded). Since the basic model was not very accurate, Boehm introduced an effort adjustment factor (EAF) that accounts for 15 multipliers representing properties of the software product, the platform, the project and the involved developers: $Effort = a\ (Size)^b\ EAF$.

In 2000, COCOMO II was published as a replacement for COCOMO 81. COCOMO II aimed to better estimating modern software projects, accounting for more recent software development processes. It was derived from a dataset of over 160 projects.

COCOMO has inspired many software development models. Specifically, one of the main shortcoming of COCOMO was that it was based on data that could be hardly representative of the development process of organizations that needed to estimate the development effort of their projects. However, the relative simplicity of the COCOMO model allowed organizations that owned historical data to develop COCOMO-like models based on their own data, thus obtaining models that represented more faithfully the organizations' characteristics and capabilities.

2) STATISTICAL MODELS USING SIMPLIFIED FUNCTIONAL SIZE MEASURES

Regression-based analysis of historical data was used to build effort models that used simplified FSMs [6], on the strand of COCOMO and similar methodologies. Specifically, Lavazza and Meli found the following COCOMO-like models for new

development projects:

$$Effort = 34.84\ UFP^{0.77}$$
$$Effort = 214.9\ \#TF^{0.693}$$

Similar models were found for enhancement projects, but no statistically significant models could be found for extension projects.

The obtained models supported the hypothesis that using simplified FSMs instead of full-fledged measures yielded similar accuracy: namely, simplified measures yielded slightly (3%) less accurate estimates for new developments and slightly better (less than 1%) estimates for enhancements.

However, the results described above were not conclusive, since they were limited to using statistical methods. In this paper, the usage of machine learning lets us complete the previous work and derive more general indications on the usability of simplified FSMs as replacements of the traditional full-fledged ones.

III. THE EMPIRICAL STUDY

This section describes the empirical study that was carried out to answer the research questions given in the introduction. Specifically, Section III-A describes the data used in the study, their characteristics and how they were split into three subsets, each corresponding to the different types of development—New development, Enhancement, Extension—that were studied. Section III-B describes the research method, namely the functional measures used as independent variables in effort estimation models, the machine learning techniques used (additional details are in Appendix), and the metrics used to evaluate the quality of the obtained estimation models.

A. THE DATASET

For the empirical study, we used the ISBSG dataset [13], [14], which has been widely used in studies involving Functional Size Measures.

The ISBSG dataset is usually released without FSM data at the fine level of granularity that we need.[1] Specifically, to carry out the analyses reported here, we needed the number of base functional components per type (EI, EO, EQ, ILF, EIF), per complexity (low, medium, high), and per activity (added, changed, deleted). Luckily, we were able to obtain a custom view of the ISBSG repository that includes the mentioned data. This view includes fewer records than the commercially released versions; namely, it contains data from 1,314 projects, while the "regular" ISBSG dataset includes several thousand records.

Among the data that characterize each project are the "Data quality rating" (concerning the completeness and reliability of the data) and "UFP rating" (concerning the trustworthiness of the UFP counting). Both are graded "A" (best) to "D" (worst), and ISBSG itself suggests to use only data rated "A" or "B". Following a consolidated practice [18], we used only the records rated "A" or "B."

The dataset includes data from real-life software development projects. Specifically, the dataset contains data from both projects addressing the development of new software products and projects addressing the enhancement of existing projects. For each project, we used the following data:

- The type of project, i.e., new development or enhancement.
- The effort spent, expressed in Person Hours.
- The size, expressed in UFP, split in added, deleted and changed size.
- The size of all ILF, EIF, EI, EO, and EQ.
- #ILF, #EIF, #EI, #EO, and #EQ, each split per complexity (high, medium, low) and activity type (added, changed, deleted).

1) THE TYPES OF PROJECTS

The relationship between effort and size depends on the type of activity performed; for instance, changing 200 FP in a 2000 FP program usually requires more effort than developing a 200 FP program from scratch. Therefore, in our study we considered three types of development activities:

- New developments, i.e., developments from scratch.
- Extensions, i.e., projects that are classified as 'enhancement' by the ISBSG, but involve no changes or deletions. These are projects that just add function to existing projects.
- Enhancement, i.e., projects that are classified as 'enhancement' by the ISBSG, and involve changes or deletions.

In practice, we split the set of projects that are classified as 'enhancement' by the ISBSG into extensions and proper enhancements, which we call simply 'enhancements' in what follows.

2) DESCRIPTIVE STATISTICS

Descriptive statistics concerning the size of projects in the ISBSG dataset are given in Table 1.

TABLE 1. Descriptive statistics of the ISBSG project sizes (values are rounded).

		New development (534 projects)	Extension (128 projects)	Enhancement (652 projects)
Size [UFP]	Mean	541	214	322
	Stdev	618	221	496
	Median	310	145	184
	Min	6	9	4
	Max	3968	1239	7134
Size [SFP]	Mean	545	216	313
	Stdev	613	227	488
	Median	319	148	174
	Min	7	9	5
	Max	4250	1405	7157

Fig. 1 illustrates the boxplot of projects' sizes, per type of development: New development ("NEW"), Enhancements ("ENH") and Extensions ("EXT"). Sizes are given in UFP

[1] A sample of the ISBSG dataset illustrating the available data can be obtained from https://481470c2.rocketcdn.me/wp-content/uploads/2022/01/ISBSG-DE-Data-ReleaseMay-2021-Sample.zip

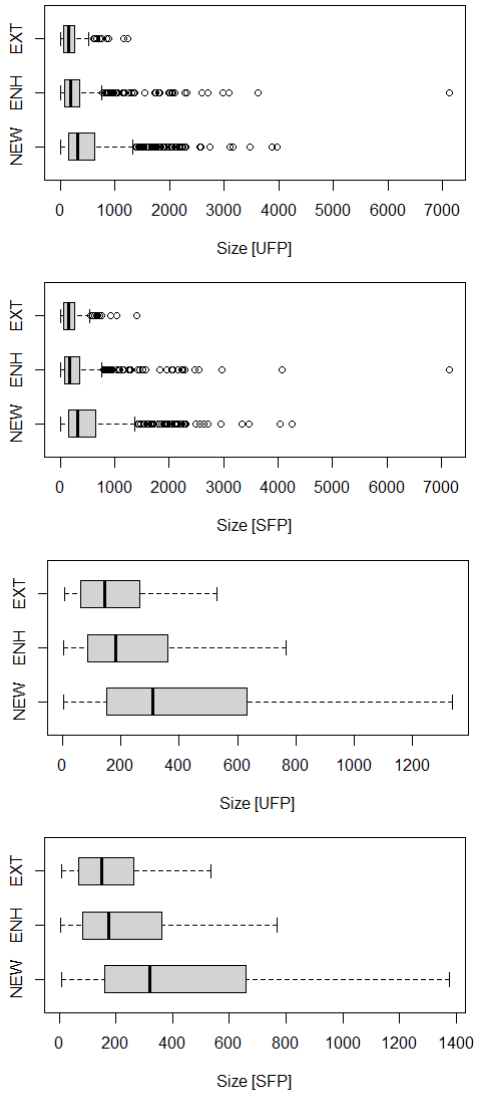

FIGURE 1. Distribution of projects per Size with (top) and without (bottom) outliers.

and SFP. For the sake of clarity, the boxplots are given both with and without outliers.

B. METHOD

Each type of project required using a different set of data: Section III-B1 describes the data used for each type of development project; Section III-B2 describes the types of ML models built; finally, Section III-B3 describes the metrics used to evaluate the accuracy of the obtained effort estimates.

1) MEASURES USED

To evaluate whether simplified measures can be used in place of traditional full-fledged measures, we have to compare effort estimates based on simplified measures with estimates based on traditional full-fledged measures. To this end, we built effort estimation models based on the two Functional Size Measures that are currently proposed by the IFPUG: Function Points (FP) and Simple Function Points (SFP).

Note that IFPUG defines both unadjusted FP (UFP) and adjusted FP. The former are a measure of functional requirements. The latter are obtained by correcting unadjusted FP to obtain an indicator that is better correlated to development effort. Noticeably, the ISO standardized only unadjusted FP, recognizing UFP as a proper measure of functional requirements [19]. Following the ISO, in this paper we deal only with UFP, even when we speak generically of Function Points or FP. Also, when we refer to "traditional full-fledged FSMs," we mean IFPUG UFP.

As described in Section II, FP and SFP are aggregated measures, obtained as weighted sums of more basic measures. In our study we consider the sets of basic measures that are used to compute FP and SFP: we use the entire set as well as subsets, looking for the minimum amount of information that supports effort estimation with acceptable accuracy.

For new developments, we tried the following sets of measures:

- {#ILF, #EIF, #EI, #EO, #EQ}
- {#EI, #EO, #EQ}
- {#TF, #DF}, where #TF = #EI+#EO+#EQ and #DF = #EIF+#ILF.
- {#TF}

For enhancements, we tried the following sets of measures (note that we retained ISBSG names, as far as possible):

- {#AddTF, #ChgTF, #DelTF, #AddDF, #ChgDF, #DelDF}
- {#AddTF, #ChgTF, #AddDF, #ChgDF}
- {#AddTF, #ChgTF, #DelTF}
- {#AddTF, #ChgTF}
- {#UpdTF, #UpdDF}
- {#UpdTF}
- {UpdUFP}
- {UpdSFP}
- {AddChgUFP}
- {AddChgSFP}
- {UpdSTF}

#AddTF, #ChgTF, #DelTF are the numbers of transaction functions (EI, EO, and EQ) that are added, changed or deleted, respectively. #AddDF, #ChgDF, #DelDF are the numbers of data functions (ILF and EIF) that are added, changed or deleted, respectively. #UpdTF is the number of transaction functions that are updated (i.e., added, changed or deleted); hence, #UpdTF = #AddTF+#ChgTF+#DelTF. #UpdDF is the number of data functions that are updated (i.e., added, changed or deleted); hence, #UpdDF = #AddDF+#ChgDF+#DelDF. UpdUFP is the measure in UFP of the updated (i.e., added, changed or deleted) FUR. UpdSFP is the measure in SFP of the updated FUR. AddChgUFP is the measure in UFP of the added or changed FUR. AddChgSFP is the measure in SFP of the added or changed FUR. UpdSTF is the measure in SFP of the updated transaction functions. Note that this measure is defined as UpdSTF=4.6 #TF, hence it is essentially equivalent to #TF.

For extensions, we tried the following sets of measures (we retained ISBSG names, as far as possible):

- {#AddILF, #AddEIF, #AddEI, #AddEO, #AddEQ}
- {#AddTF, #AddDF}
- {#AddTF}
- {AddSFP}
- {AddUFP}

#AddILF, #AddEIF, #AddEI, #AddEO and #AddEQ are the numbers of ILF, EIF, EI, EO, and EQ, respectively, that were added. #AddTF and #AddDF are the numbers of transaction and data functions that were added, respectively. AddSFP is the measure in SFP of the added functions (both transactions and data). AddUFP is the measure in UFP of the added functions (both transactions and data).

2) MODEL BUILDING

To address the research questions, we built models that predict development effort based on the measures described in Section III-B1. Specifically, we built models using a few machine learning techniques, namely Support Vector Regression (SVR), Neural Networks (NN), Random Forests (RF), and K-nearest neighbours (KNN). The choice of these techniques was made also considering the results of previous studies [10], which found that SVR and NN yield the most accurate effort models, in general. The details concerning the construction of ML models are given in Appendix.

Specifically, we used each of the aforementioned ML techniques to build models for each type of project, using all the measure combinations described in Section III-B1. Predictive models were evaluated via 10-times 10-fold cross-validation.

3) ACCURACY MEASURES

The accuracy of the obtained estimates was evaluated via the mean of absolute residuals (MAR)—also known as the mean absolute error (MAE)—which is an unbiased indicator, recommended by several authors (e.g., [20]). Given a set of observations $Y = \{y_i\}$ with $i \in [1, n]$, the residual (or error) of the i^{th} estimate $\hat{y}_i$ is $y_i - \hat{y}_i$, where y_i is the i^{th} observation (i.e., the actual effort) and $\hat{y}_i$ is the estimate for y_i. The MAR is then computed as the mean of absolute residuals, as follows:

$$MAR = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

The MAR values obtained for different datasets are not comparable. To make our results comparable with those possibly obtained using different datasets, we also compute a normalized measure by dividing estimation errors by the mean actual value of effort. Specifically, we proceed as follows: given a set of observations Y,

- The residual of the i^{th} estimate $\hat{y}_i$ is $y_i - \hat{y}_i$, where y_i is the i^{th}'s project actual effort.
- The mean actual value $\bar{y}$ is $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, where n is the number of observations in Y, i.e., the number of projects in the datasets.
- We consider the ratio (RR) between absolute residuals and the mean of actuals: $RR_i = \frac{|y_i - \hat{y}_i|}{\bar{y}}$.
- Then, we compute MR, the mean of RR, as follows:

$$MR = \frac{1}{n}\sum_{i=1}^{n} RR_i = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{\bar{y}} = \frac{1}{\bar{y}}\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| = \frac{MAR}{\bar{y}}$$

In this way, we get MR values that are comparable across datasets. Unlike MMRE, i.e., the Mean Magnitude of Relative Errors, defined as $\frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i}$, MR is not biased, since in the computation of MR the absolute residuals of a given dataset are all divided by the same number ($\bar{y}$).

Now, we have to consider that our data tend to be skewed, since many of the projects that produced the data were fairly small. Therefore, to provide a more representative evaluation, we also computed indicators that are less sensitive to skewness than MAR and MR. Specifically, we computed the median of absolute residuals (MdAR) and the ratio MdR between MdAR and the median absolute value of effort.

We must also consider that we can find that two models obtain similar MAR values. In those cases, it is important to evaluate to what extent a prediction is better than the other one. To perform this type of evaluation, we also computed the effect size on absolute residuals. The effect size was computed via Hedges's g, i.e., via Cohen's d statistics [21] with Hedges correction [22].

C. OVERVIEW OF THE PROCESS

The process we followed to carry out the empirical study is schematically described in Fig. 2.

The first step concerned data preparation: we selected from the ISBSG dataset the subsets of data concerning each of the considered types of project (NEW, EXT, ENH): this resulted in three datasets, which were then analysed separately. Each dataset contained a different set of metrics, as described in detail in Section III-B1.

The second step was repeated for each type of project. An effort model was developed using each one of the considered machine learning methods, as specified in Section III-B2.

Finally, the third and last step consisted in applying the model obtained at step 2 to get effort estimates, which were then compared to the actual effort data. The obtained errors were evaluated using the accuracy metrics described in Section III-B3.

IV. RESULTS

In this section, we report about the models built to answer the research questions. Specifically, results concerning new development projects are given in Section IV-A; results concerning enhancement projects are given in Section IV-B; results concerning extension projects are given in Section IV-C.

A. RESULTS CONCERNING NEW DEVELOPMENT PROJECTS

As stated in Section III above, we tried building effort models using different sets of features (as specified in

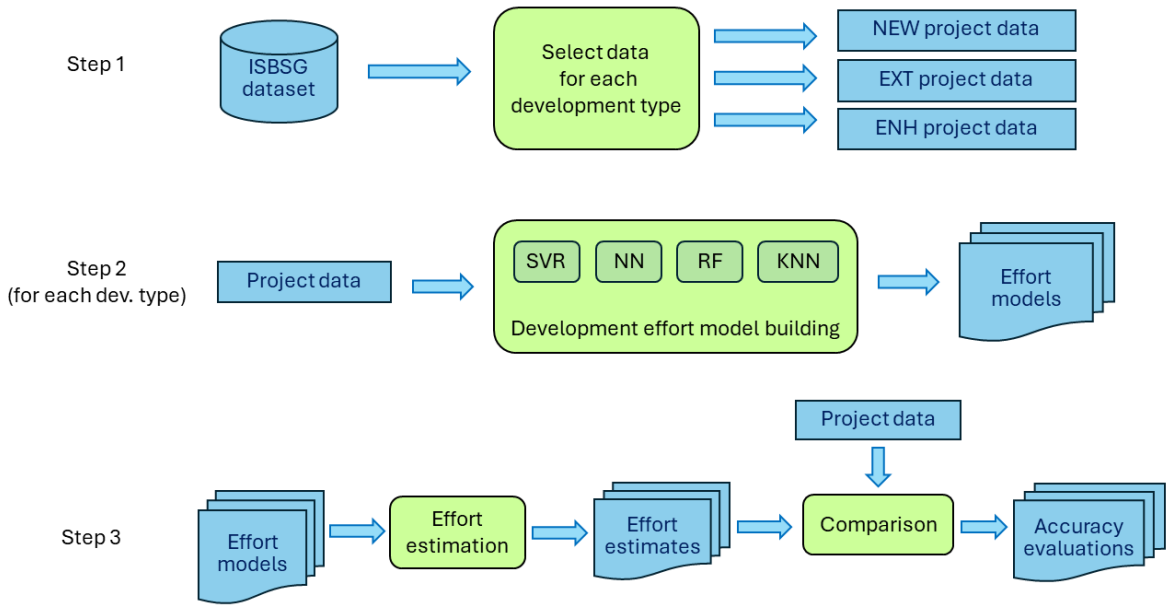

FIGURE 2. Overview of the evaluation process.

TABLE 2. MR of models for new developments, obtained using various sets of features and different ML techniques.

	FSn1	FSn2	FSn3	FSn4	FSn5	FSn6
SVR	0.635	0.645	0.667	0.673	0.647	0.657
NN	0.752	0.773	1.027	0.896	0.805	0.770
KNN	0.800	0.809	0.775	0.773	0.808	0.794
RF	0.781	0.804	0.708	0.726	0.743	0.795

TABLE 3. MdR of models for new developments, obtained using various sets of features and different ML techniques.

	FSn1	FSn2	FSn3	FSn4	FSn5	FSn6
SVR	0.542	0.544	0.579	0.550	0.550	0.545
NN	0.834	0.881	0.987	0.911	0.886	0.829
KNN	0.840	0.825	0.759	0.787	0.830	0.816
RF	0.790	0.804	0.798	0.798	0.844	0.814

Section III-B1) with a few different ML techniques (specified in Section III-B2).

All the used ML techniques were able to provide effort models with all the considered sets of features. As described in Section III-B2 above, we carried out 10-times 10-fold cross validations, to compute the accuracy of the effort models. The results are summarized in Tables 2 and 3. Specifically, Tables 2 provides the MR (i.e., the mean of RR, the ratio between absolute residuals and the mean of actuals: see Section III-B1), while Table 3 provides MdR (i.e., the median of RR), for all the combinations of ML techniques and features sets. To keep Tables and Figures readable, throughout this section features subsets are not given explicitly, instead they are labeled, as follows:

- FSn1: {UFP}
- FSn2: {SFP}
- FSn3: {#ILF, #EIF, #EI, #EO, #EQ}
- FSn4: {#EI, #EO, #EQ}
- FSn5: {#TF , #DF}
- FSn6: {#TF}

It can be noticed that the estimation errors illustrated in Tables 2 and 3 (as well as in the next sections) are not small. In fact, since the seminal work by Barry Boehm [23], we know that development effort depends on the size of the software to be developed as well as on many other factors, concerning the software product (complexity, non-functional requirements, etc.), the process (developers' experience and skill, methods and tools used, etc.) and the environment (schedule constraints, requirements changes, etc.). The projects represented in the ISBSG dataset have heterogeneous origins and characteristics, hence they are characterized by a variety of the aforementioned factors. However, these factors are not suitably represented in the ISBSG dataset: for instance, there is no indication of the complexity of the code being developed, of the experience and skill of developers, of non-functional requirements, etc. So, we could not use any of the mentioned features in building effort models. To this respect, it is important to notice that:

- Our analysis focuses on the ability of different size measures to affect effort estimation, thus considering other features might introduce a confounding effect.
- Our research questions concern the difference of accuracy achieved by different FSMs and model building techniques, rather the absolute magnitude of estimation errors.
- At any rate, the models we obtained are fairly accurate, considering that they are based on size only: COCOMO base mode, which also estimated effort based on size only [23], achieved pred(20%)=25%, while our models achieve pred(20%)=40%.

In practice, the results illustrated in Tables 2 and 3 are sufficient to answer RQ1 for new development projects, as discussed in Section V below.

Looking at Tables 2 and 3, it seems that SVR models are the most accurate. However, to evaluate which ML technique provides the best models, we compared ML techniques via the Wilcoxon signed rank test, applied to the absolute errors yielded by each pair of models. The obtained results are given in Table 4. The "<" (respectively, ">") sign indicates that the model in the row provides significantly smaller (respectively, greater) absolute errors than the model in column. All the presented results are significant at the usual $\alpha = 0.05$ level.

TABLE 4. Wilcoxon sign rank tests for new development effort models.

	Model using FSn1 (UFP)				Model using FSn2 (SFP)				Model using FSn6 (#TF)			
	SVR	NN	KNN	RF	SVR	NN	KNN	RF	SVR	NN	KNN	RF
SVR	–	<	<	<	–	<	<	<	–	<	<	<
NN	>	–	<	<	>	–	<	>	>	–	<	<
KNN	>	>	–	>	>	>	–	>	>	>	–	>
RF	>	>	<	–	>	<	<	–	>	>	<	–

Table 4 reports the results for models based on UFP, SFP and #TF. The results for the models using other sets of features are not given for space reasons; however, in all cases, SVR is more accurate than other ML models, according to Wilcoxon sign rank test.

The results illustrated in Table 4 are sufficient to answer RQ2 for new development projects, as discussed in Section V below.

The results illustrated above and Tables 2 and 3 suggest that all the considered sets of size features support very similar levels of accuracy, when used for effort estimation. To verify this hypothesis, we computed the effect size on absolute errors. The effect size was computed via Hedges's g, i.e., via Cohen's d statistics [21] with Hedges correction [22].

Table 5 reports the values of Hedges's g for the SVR models of new development projects' effort. It can be seen that all values are very close to zero, indicating negligible effect sizes.

TABLE 5. Hedges's g for the absolute residuals of new development SVR effort models.

	FSn1	FSn2	FSn3	FSn4	FSn5	FSn6
FSn1	–	-0.009	-0.028	-0.033	-0.011	-0.020
FSn2	0.009	–	-0.019	-0.024	-0.002	-0.011
FSn3	0.028	0.019	–	-0.005	0.017	0.008
FSn4	0.033	0.024	0.005	–	0.023	0.014
FSn5	0.011	0.002	-0.017	-0.023	–	-0.009
FSn6	0.020	0.011	-0.008	-0.014	0.009	–

The data in Table 5 are sufficient to answer RQ3 for new development projects, as discussed in Section V below.

As a complement to the analysis illustrated above, the boxplots or estimation errors, absolute errors and RR are given in Fig. 3 and 4.

The results given above are sufficient to answer RQ4 for new development projects, as discussed in Section V below.

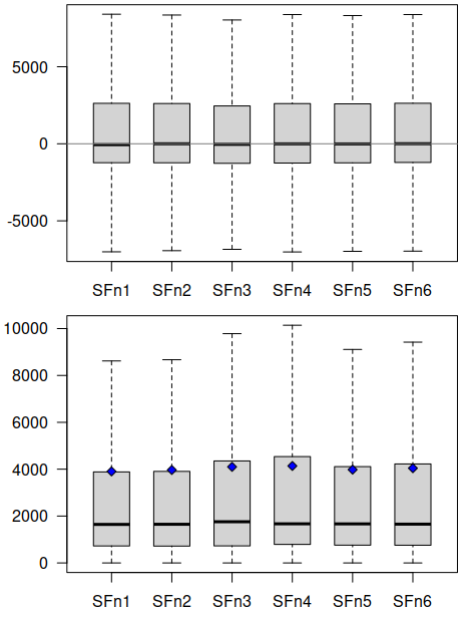

FIGURE 3. Boxplots of estimation errors and absolute estimation errors, for SVR models of new development (outliers not shown).

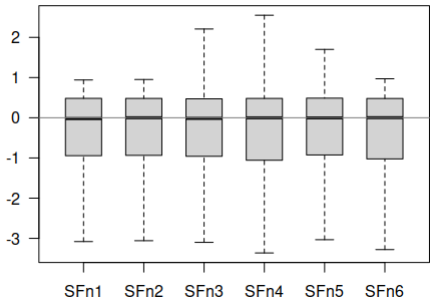

FIGURE 4. Boxplots of RR (the ratio between absolute estimation errors and mean actuals), for SVR models of new developments (outliers not shown).

B. RESULTS CONCERNING ENHANCEMENT PROJECTS

In this section, we report about the effort models for projects involving the enhancement of existing software. As already mentioned, we consider as enhancement projects those involving changes or deletions, and possibly extensions.

All the considered ML techniques provided effort models with all the considered sets of features. The results of 10-times 10-fold cross validations are summarized in Tables 6 and 7, which report the MR and MdR, respectively, for all the combinations of ML techniques and features sets.

To keep Tables and Figures readable, throughout this section features subsets are not given explicitly, instead they are labeled as follows:

- FSe1: {UFP}
- FSe2: {#AddTF, #ChgTF, #DelTF, #AddDF, #ChgDF, #DelDF}
- FSe3: {#AddTF, #ChgTF, #AddDF, #ChgDF}
- FSe4: {#AddTF, #ChgTF, #DelTF}
- FSe5: {#AddTF, #ChgTF}
- FSe6: {#UpdTF, #UpdDF}
- FSe7:] {#UpdTF}

TABLE 6. MR of models for enhancements, obtained using various sets of features and different ML techniques.

	FSe1	FSe2	FSe3	FSe4	FSe5	FSe6	FSe7	FSe8	FSe9	FSe10	FSe11
SVR	0.668	0.650	0.644	0.659	0.665	0.663	0.663	0.675	0.669	0.654	0.661
NN	0.929	1.130	1.096	0.976	0.901	0.891	0.826	0.926	0.933	0.920	0.907
KNN	0.817	0.766	0.767	0.781	0.789	0.763	0.784	0.819	0.814	0.800	0.801
RF	0.797	0.702	0.703	0.709	0.734	0.716	0.797	0.816	0.796	0.790	0.797

TABLE 7. MdR of models for enhancements, obtained using various sets of features and different ML techniques.

	FSe1	FSe2	FSe3	FSe4	FSe5	FSe6	FSe7	FSe8	FSe9	FSe10	FSe11
SVR	0.583	0.572	0.562	0.552	0.553	0.582	0.567	0.591	0.584	0.592	0.595
NN	1.007	1.020	0.999	0.933	0.922	0.934	0.883	0.977	1.010	1.034	1.010
KNN	0.786	0.730	0.747	0.729	0.754	0.712	0.743	0.778	0.776	0.795	0.768
RF	0.770	0.734	0.755	0.753	0.745	0.730	0.760	0.794	0.765	0.750	0.733

- FSe8: {AddChgUFP}
- FSe9: {UpdUFP}
- FSe10: {UpdSFP}
- FSe11: {AddChgSFP}

The results illustrated in Tables 6 and 7 are sufficient to answer RQ1 for enhancement projects, as discussed in Section V below. Tables 6 and 7 seem to confirm that SVR models are the most accurate, as with new developments. However, to evaluate which ML technique provides the best models, we compared ML techniques via the Wilcoxon signed rank test, applied to the absolute errors yielded by each pair of models. The obtained results are given in Table 8.

TABLE 8. Wilcoxon sign rank tests for enhancement effort models.

	Model using FSe1 (UFP)				Model using FSe7 (#UpdTF)			
	SVR	NN	KNN	RF	SVR	NN	KNN	RF
SVR	–	<	<	<	–	<	<	<
NN	>	–	>	>	>	–	>	>
KNN	>	<	–	>	>	<	–	<
RF	>	<	<	–	>	<	>	–
	Model using FSe9 (UpdUFP)				Model using FSe10 (UpdSFP)			
	SVR	NN	KNN	RF	SVR	NN	KNN	RF
SVR	–	<	<	<	–	<	<	<
NN	>	–	>	>	>	–	>	>
KNN	>	<	–	>	>	<	–	>
RF	>	<	<	–	>	<	<	–

Table 8 reports the results for models based on UFP, #UpdTF, UpdUFP and UpdSFP. The results for the models using other sets of features are not given for space reasons; however, in all cases, SVR appears more accurate than other ML models, according to Wilcoxon sign rank test.

The results illustrated in Table 8 are sufficient to answer RQ2 for enhancement projects, as discussed in Section V below.

As for new developments, the results given in Tables 6 and 7 suggest that all the considered sets of size features support very similar levels of accuracy, when used for effort estimation. To verify this hypothesis, we computed the effect size on absolute errors. Table 9 reports the values of Hedges's g for the SVR models of enhancement projects' effort. It can be seen that all values are very close to zero, indicating negligible effect sizes.

The data in Table 9 are sufficient to answer RQ3 for enhancement projects, as discussed in Section V below.

TABLE 9. Hedges's g for the absolute residuals of enhancement SVR effort models.

	FSe1	FSe2	FSe3	FSe4	FSe5	FSe6	FSe7	FSe8	FSe9	FSe10	FSe11
FSe1	—	0.017	0.023	0.009	0.003	0.005	0.005	-0.006	-0.001	0.014	0.007
FSe2	-0.017	—	0.005	-0.008	-0.014	-0.012	-0.012	-0.024	-0.018	-0.004	-0.010
FSe3	-0.023	-0.005	—	-0.014	-0.019	-0.017	-0.017	-0.029	-0.023	-0.009	-0.016
FSe4	-0.009	0.008	0.014	—	-0.006	-0.004	-0.004	-0.015	-0.010	0.005	-0.002
FSe5	-0.003	0.014	0.019	0.006	—	0.002	0.002	-0.010	-0.004	0.010	0.004
FSe6	-0.005	0.012	0.017	0.004	-0.002	—	0.000	-0.012	-0.006	0.008	0.002
FSe7	-0.005	0.012	0.017	0.004	-0.002	0.000	—	-0.011	-0.006	0.009	0.002
FSe8	0.006	0.024	0.029	0.015	0.010	0.012	0.011	—	0.006	0.020	0.013
FSe9	0.001	0.018	0.023	0.010	0.004	0.006	0.006	-0.006	—	0.014	0.008
FSe10	-0.014	0.004	0.009	-0.005	-0.010	-0.008	-0.009	-0.020	-0.014	—	-0.007
FSe11	-0.007	0.010	0.016	0.002	-0.004	-0.002	-0.002	-0.013	-0.008	0.007	—

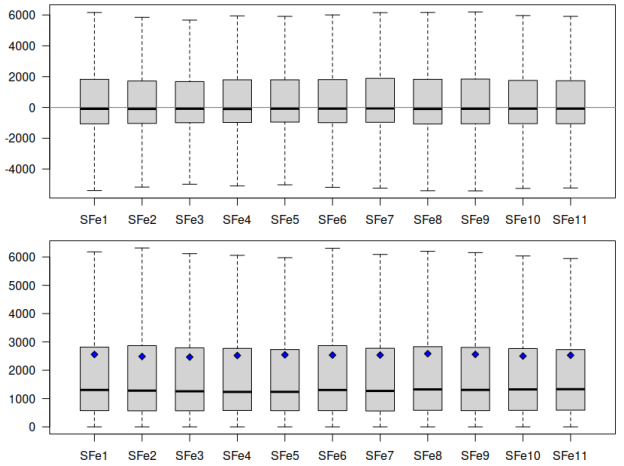

FIGURE 5. Boxplots of estimation errors and absolute estimation errors, for SVR models of enhancements (outliers not shown).

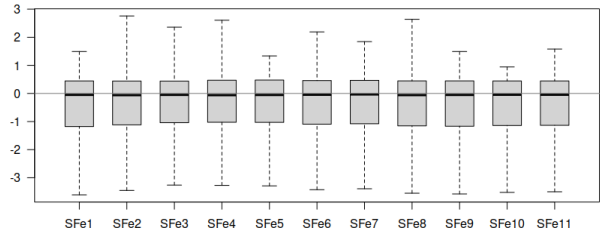

FIGURE 6. Boxplots of RR, for SVR models of enhancements (outliers not shown).

As a complement to the analysis illustrated above, the boxplots or estimation errors, absolute errors and RR are given in Fig. 5 and 6, respectively.

C. RESULTS CONCERNING EXTENSION PROJECTS

In this section, we report about the effort models for projects involving the extension of existing software, without any change or deletion of existing functionality.

All the considered ML techniques yielded effort models with all the considered sets of features. The results of 10-times 10-fold cross validations are summarized in Tables 10 and 11, which report the MR and MdR, respectively, for all the combinations of ML techniques and features sets.

To keep Tables and Figures readable, subsets are not given explicitly, instead they are labeled as follows:

- FSx1: {AddUFP}
- FSx2: {AddSFP}
- FSx3: {#AddILF, #AddEIF, #AddEI, #AddEO, #AddEQ}
- FSx4: {#AddTF, #AddDF}
- FSx5: {#AddTF}

TABLE 10. MR of models for extensions, obtained using various sets of features and different ML techniques.

	FSx1	FSx2	FSx3	FSx4	FSx5
SVR	0.797	0.745	0.815	0.755	0.769
NN	1.041	1.116	4.065	2.443	1.162
KNN	0.977	0.960	0.932	1.071	1.031
RF	0.910	0.958	0.850	0.938	0.937

TABLE 11. MdR of models for extensions, obtained using various sets of features and different ML techniques.

	FSx1	FSx2	FSx3	FSx4	FSx5
SVR	0.599	0.566	0.636	0.637	0.553
NN	0.872	1.041	1.790	1.130	1.073
KNN	0.740	0.755	0.684	0.828	0.825
RF	0.716	0.717	0.873	0.921	0.787

The results illustrated in Tables 10 and 11 are sufficient to answer RQ1 for extension projects, as discussed in Section V below. Tables 10 and 11 seem to confirm that SVR models are the most accurate, as with new developments and enhancements. However, to evaluate which ML technique provides the best models, we compared ML techniques via the Wilcoxon signed rank test, applied to the absolute errors yielded by each pair of models. The obtained results are given in Table 12.

TABLE 12. Wilcoxon sign rank tests for enhancement effort models.

	Model using FSx1 (AddUFP)				Model using FSx2 (AddSFP)				Model using FSx5 (#AddTF)			
	SVR	NN	KNN	RF	SVR	NN	KNN	RF	SVR	NN	KNN	RF
SVR	—	<	<	<	—	<	<	<	—	<	<	<
NN	>	—	>	>	>	—	>	>	>	—	>	>
KNN	>	<	—	>	>	<	—	>	>	<	—	>
RF	>	<	<	—	>	<	<	—	>	<	<	—

Table 12 reports the results for models based on AddUFP, AddSFP, and #AddTF. The results for the models using other sets of features are not given for space reasons; however, in all cases, SVR appears more accurate than other ML models, according to Wilcoxon sign rank test. The results illustrated in Table 12 are sufficient to answer RQ2 for extension projects, as discussed in Section V below.

As for new developments and enhancements, the results given in Tables 10 and 11 suggest that all the considered sets of size features support very similar levels of accuracy, when used for effort estimation. To verify this hypothesis, we computed the effect size on absolute errors. Table 13 reports the values of Hedges's g for the SVR models of extension projects' effort. It can be seen that all values are very close to zero, indicating negligible effect sizes.

TABLE 13. Hedges's g for the absolute residuals of extension SVR effort models.

	FSx1	FSx2	FSx3	FSx4	FSx5
FSx1	–	0.029	-0.010	0.023	0.016
FSx2	-0.029	–	-0.040	-0.006	-0.013
FSx3	0.010	0.040	–	0.035	0.026
FSx4	-0.023	0.006	-0.035	–	-0.007
FSx5	-0.016	0.013	-0.026	0.007	–

The data in Table 13 are sufficient to answer RQ3 for extension projects, as discussed in Section V below.

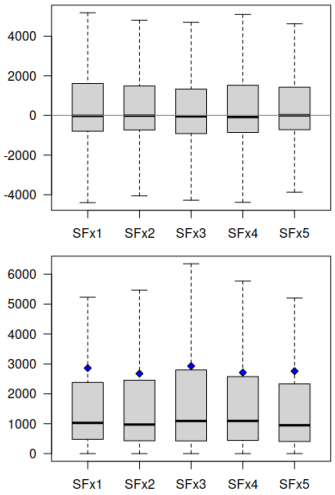

FIGURE 7. Boxplots of estimation errors, for SVR models of extensions (outliers not shown).

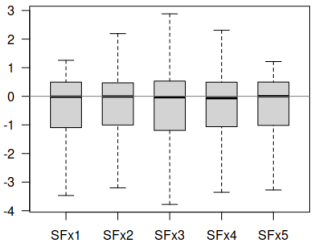

FIGURE 8. Boxplots of RR, for SVR models of extensions (outliers not shown).

As a complement to the analysis illustrated above, the boxplots or estimation errors, absolute errors and RR are given in Fig. 7 and 8.

D. SUMMARY OF RESULTS

The results obtained through Section IV are summarized in Table 14: for each type of projects (NEW, EXT, ENH) the most accurate effort estimates achieved by different types of FSMs are given (the most accurate is in boldface). Specifically, Table 14 shows the best results achieved with UFP (i.e., the traditional IFPUG standard full-fledged metric), SFP (i.e., the IFPUG standard simplified metric), and combinations of basic unweighted measures. For each type of metric, the machine learning method that yielded the best results and the MR measure of errors are given. In addition, the specific subset of basic unweighted measures that supported the most accurate effort estimation is given.

It is worth noting that the best results were always obtained via Support Vector Regression. It is also quite interesting that i) for a given type of project, different types of metrics supported extremely similar effort estimation errors, and ii) for each project type, a different metric supports the most accurate estimation: for instance, UFP supported the most accurate model for NEW projects, while SFP yielded the best estimates for EXT projects. These observations support the idea that FSMs can be used in practice for effort estimation

TABLE 14. Summary of results: best ML method and metrics class for each project type.

	UFP		SFP		Extremely simplified metrics		
	ML model	MR	ML model	MR	Metrics	ML model	MR
NEW	SVR	**0.635**	SVR	0.645	{#TF, #DF}	SVR	0.647
EXT	SVR	0.797	SVR	**0.745**	{#AddTF, #AddDF}	SVR	0.755
ENH	SVR	0.668	SVR	0.654	{#AddTF, #ChgTF, #AddDF, #ChgDF}	SVR	**0.644**

instead of full-fledged metrics with no penalties with respect to accuracy.

V. EVALUATION OF RESULTS

In this section, we provide answers to the research questions and we discuss the consequences of our findings.

A. ANSWER TO RESEARCH QUESTION RQ1

RQ1 asks whether it is possible to build ML models that are able to predict development effort, based on a subset of the information needed to compute traditional full-fledged FSM.

Based on the results given in Section IV, the answer to RQ1 is definitely positive. In fact, we obtained models for all types of projects, with all the considered ML techniques and with all the considered measure subsets.

B. ANSWER TO RESEARCH QUESTION RQ2

RQ2 asks if there is any ML technique that yields substantially better estimates than other techniques.

Based on the results given in Section IV, we can state that SVR models provide estimates that are more accurate than those yielded by RF, NN and KNN models. Noticeably, this is true for all the types of projects considered (New developments, Enhancements and Extensions) and for all the considered sets of measures.

C. ANSWER TO RESEARCH QUESTION RQ3

RQ3 asks whether there are models using simple measures that yield effort estimates not substantially worse than estimates obtained from traditional, full-fledged FSMs.

Based on the MR, MdR and effect size evaluations given in Section IV, we can state that all effort models, i.e., those using traditional IFPUG measures and those using subsets of simpler size measures, yield effort estimates having substantially equivalent accuracy. Specifically, the differences in MR and MdR are so small to be deemed irrelevant by most project managers, while effect size confirms that the observed differences are of negligible magnitude.

D. ANSWER TO RESEARCH QUESTION RQ4

RQ4 asks what is, if any, the most basic feature that supports estimation with acceptable accuracy, with respect to estimates obtained from traditional, full-fledged FSMs.

For all types of projects, we built effort models based exclusively on the number of transaction functions. Namely, we used #TF for new development projects, #UpdTF for enhancement projects, and #AddTF for extension projects: these are by far the simplest measures we used.

According to the evaluations reported in Section IV, the models based on the number of transaction functions proved as accurate as the models based on other measures, including those accounting for all the available features of FUR. It is noticeable that for enhancements and extensions, the model based on the number of transaction functions were, though marginally, more accurate than those based on IFPUG UFP.

E. DISCUSSION OF RESULTS

1) IMMEDIATELY USEFUL RESULTS

As already mentioned in the introduction, functional size measurement is useful because it can be applied in the early stages of development, based on requirements specifications, and it provides the basis for effort estimation, which is also an activity carried out at the beginning of software projects.

In many cases, project managers need a rough estimate of development effort even before functional requirements have been described in full detail. This led to the proposal of many simplified measurement methods (also known as approximate measurement methods) [3], [4], [24], [25], [26], [27], [28], [29], [30]. It is worth mentioning that simplified measures not only anticipate the availability of functional measures; they also make the measurement process faster and less expensive, a fact that is clearly appreciated by software project managers, as long as the inherent approximation of simplified measures does not decrease too much the accuracy of effort estimates.

In this paper, we have described a study that shows that even "extreme" simplification of functional size measurement seems not to affect effort estimation accuracy. This is good news for software project managers, who can start using (or at least can start experimenting with) the number of transactions as the measure of functional size.

2) FUNCTIONAL MEASUREMENT FOR AGILE PROCESSES

Standard functional size measurement methods—like IFPUG UFP and SFP—define precisely the element to be considered (e.g., the elementary processes), and yield measures that are independent on technological and environmental factors. Therefore, in principle these methods could be beneficial also in agile processes, where the identification, description and measurement of requirements are often performed in ad-hoc manners: so, for instance, the granularity of user stories can vary, even in the same organization.

However, traditional full-fledged FSMs are hardly used in agile contexts [31]. In fact, traditional FSM methods (like

FPA or the COSMIC method [32]) require that functional requirements are documented in detail, which is seldom the case in agile processes. Even when FUR have been specified completely and at the needed detail level, performing the measurement process is perceived as a "heavy" activity that does not fit well in agile processes.

However, simplified measures could play a role in agile processes; in fact, the usage of SiFP in agile processes has already been reported [33]. Also the number of transactions #TF could be effectively employed in agile processes. In fact, usually a sprint addresses a piece of the product backlog, named the sprint backlog, concerning the realization of a set of stories. Implementing a story usually involves adding some functionalities, i.e., in FPA terms, adding some transactions, and possibly changing the related data and transactions as well. Now, predicting what data needs to be changed while implementing the new functionality is not always obvious, and finding that information would need a detailed analysis that is out of the scope of agile processes. Instead, characterizing the work to be done in a sprint in terms of transaction functions is possible and easy, because transactions can usually be identified from user stories.

The projects from the ISBSG dataset are mostly carried out using traditional, not agile, processes; hence the quantitative results presented in the previous sections are not directly applicable to agile processes, especially concerning estimation accuracy. However, our results show that it is possible to estimate effort using #TF as the only measure of functional size; hence, in principle #TF is usable to estimate the effort for implementing functional stories at the sprint level in agile contexts. Such practice is definitely worth exploring.

3) TOWARDS MORE SOPHISTICATED EFFORT MODELS

Development effort does not depend exclusively on the size of the software to be developed: even the earliest effort estimation methods accounted for additional factors, involving, for instance, the development environment, the characteristics of developers, etc. [23].

However, all effort estimation models include the size of software as one of the independent variables. Earliest methods used the number of lines of code (LOC) as the preferred size metric, while more recent methods tend to use FSMs. Unfortunately, FSMs represent effectively only a few aspects of "functionality": both FPA and COSMIC [32] measure the amount of data used to implement a functionality, as well as the amount of data exchanged with the user, but they do not address the complexity or amount (extension) of elaboration required to implement the required functionality. It has been shown that the amount and complexity of elaboration can affect the effort required for development to a large extent [34].

These considerations, together with the results of this study, suggest that traditional functional size measurement methods are unnecessarily complex, since a very straightforward measure like #TF provides equivalent performance. Nonetheless, they do not capture essential information concerning the amount or complexity of elaboration required to implement FURs; specifically, neither IFPUG UFP nor COSMIC Function Points represent the difference between the functionality needed to compute a chess move or a tic-tac-toe move. Therefore, it is advisable that researchers address the problem of measuring the amount and complexity of elaboration involved in user requirements, since such information in combination with even very simple functional measures is likely to improve estimation accuracy, possibly to a large extent.

VI. THREATS TO VALIDITY

In this section we discuss the threats to the validity of the study. Note that we follow the indications by Verdecchia et al. [35]: accordingly, instead of providing a comprehensive list of all the threats that could possibly affect a study like this, we focus on those threats that are relevant in our specific case, and that we have addressed and mitigated as possible.

A. INTERNAL VALIDITY

A possibly relevant threat concerns the correctness of the used data. In fact, in empirical studies like ours, the quality of data determines the quality of results. To mitigate the risks deriving from the usage of bad quality data, we selected from the ISBSG dataset only the data having high quality ratings (see details in Section III-A), as is common practice [18].

B. CONSTRUCTION VALIDITY

The main issues with construction validity may concern 1) the methods through which the machine learning models were obtained and used, and 2) the way effort estimation errors were measures and evaluated.

Concerning the first threat, we used widely adopted methods via thoroughly tested implementations (as described in Section III-B2 and Appendix). To mitigate the possible residual threats, particular attention was put in configuring the models via hyper-parameters, as described in Appendix.

Concerning the evaluation of the obtained effort estimates, we addressed potential threats by using multiple complementary metrics:

- *MAR* and *MdAR* were used to evaluate absolute estimation errors;
- *MR* and *MdR* were used to evaluate relative estimation errors. Note that these metrics were preferred to *MMRE* and *MdMRE* because the latter are biased [20].
- Wilcoxon signed rank test was used to compare effort estimation errors obtained by different models.
- Hedges's g was used to evaluate the importance of differences between errors.

The fact that all the mentioned metrics provided mutually coherent results increases our confidence that the evaluations are correct.

A possible concern with the representativeness of the set of projects that supplied ISBSG data is with the size and effort

distribution. As shown by Fig. 1, most projects are relatively small. This is a quite common characteristic of software engineering datasets, and it is not actually worrying, because in practice smaller projects are definitely more frequent than larger ones.

C. EXTERNAL VALIDITY

Concerning external validity, we need to consider whether we can generalize the results of our study. To this end, we can observe that—as described in Section III-A2—the ISBSG dataset we used includes data from projects that have different characteristics. Specifically, the datasets we analyzed accounts for projects developed in 16 different industrial sectors, in a 25 year range. The projects have various size: e.g., our enhancement project involve functionality in the [4, 7134] UFP range. Accordingly, the considered projects required quite different effort (from 21 PersonHours to over 300 PersonMonths) and were carried out by teams having various sizes (1 to 49 members). The multiplicity of development situations represented in the used data supports some confidence that the presented results are generalizable.

At any rate, there are some possible limitations to the generalizability of the obtained results. For instance, our data did not account for any very large project (10,000 UFP or more): accordingly, our results could not apply to very large projects. Nonetheless, where evidence based on data is lacking, conceptual considerations can be used. For instance, when reasoning about very large projects, we can note that simple measures are as good as full-fledged measures in representing economies of scale. Similarly, larger project size is generally due to the number of transactions and data functions, which are the base of simple measures. Simple measures could fall short in estimating development effort of very large projects only if those projects were characterized by higher complexity of transactions and data functions: an unlikely situation that can be easily checked by analyzing in detail a function sample.

It can be noted that several of the mentioned threats derive from the nature and characteristics of the the ISBSG dataset. In this respect, it must be observed that the ISBSG dataset is the largest publicly available dataset providing data concerning software development and functional sizes. It is also the most widely used. As a consequence, several of the aforementioned threats are shared with the other scientific papers that analyzed ISBSG data.

VII. RELATED WORK

Effort estimation has always been a hot topic in software engineering. Thousands of papers have been published, many concerning the usage of FSMs and simplified measures. However, of these many papers, only a small subset addresses topics that are somewhat related to our research. Specifically, we have identified the following categories:

- Papers that use simplified FSMs for estimating development effort;
- Papers that use alternative size measures for estimating development effort in the early stages of development;
- Paper that deal with using machine learning techniques for building effort estimation models;
- Papers that deal with using (not necessarily simplified) FSMs for effort estimation in agile processes.

We retrieved the relevant papers in the aforementioned categories and selected the most relevant ones using popular techniques used for systematic literature reviews (SLR) [36], [37]. The papers recognized as relevant and related to our research are discussed in the following sections.

A. EFFORT ESTIMATION WITH SIMPLIFIED FUNCTIONAL SIZE MEASURES

Although FSMs can be used for several purposes, their most common application is for effort estimation. When a new simplified functional size measurement method is introduced, its effectiveness for effort prediction is not known; therefore, empirical evaluations are necessary to verify if the approximate sizing approach fails to capture factors affecting software development effort.

In 2011, Simple Function Points (SiFP) were introduced as a simple alternative to IFPUG Function Points [3]. In 2014, Lavazza and Meli [5] tested the ability of SiFP to support estimates as accurate as those obtained with IFPUG UFP, using a version of the ISBSG dataset. Effort models were built using linear regression after log-log transformation. The obtained results showed that SiFP and IFPUG UFP achieved the same level of accuracy.

In 2016, the study by Lavazza and Meli was replicated by Ferrucci et al. [38], who used a dataset that accounted for of 25 Web applications developed by a single software company. The study showed that SiFP and IFPUG UFP are equally accurate in predicting the development effort of Web applications.

In 2020, Lavazza and Meli replicated the study they performed in 2014, using a more recent version of the ISBSG dataset (dated 2019) [6]. They performed the evaluation separately for new development, enhancement and extension projects. For the first time, they used the sheer number of transactions as an effort estimator. The study showed that the difference in accuracy is negligible for new development projects and extensions, while no statistically significant conclusion could be drawn for enhancement projects.

While SFP (formerly known as SiFP) is a simple FSM that was proposed as an alternative to IFPUG UFP, some methods were proposed to estimate the size of software expressed in IFPUG UFP. Among these methods, the most popular is probably the NESMA estimated method [26], also known as High-Level FPA [40].

An early evaluation by Ohiwa et al. [41] found a good linear correlation between NESMA estimated function point count and software development effort. For the evaluation, they used the software development data of 36 projects (dated 2008 through 2012) extracted from a larger software repository maintained by the Economic Research Association.

Ohiwa et al. did not provide any measure of the effort estimation accuracy, though.

In 2012, Popović and Bojić used several size measures to estimate effort in various phases of software development [42]. They analyzed a set of 50 small and medium size real-world Web based projects, all carried out from 2004 to 2010 by a company at CMMI level 2. They built effort estimation models using regression, after log-log transformation. They found that the NESMA estimated method featured good accuracy in the elaboration phase. Specifically, MMRE was 13–16%, depending on the model used, slightly worse than the MMRE in the 10–12% range, achieved when using IFPUG FP.

In 2020, Di Martino et al. [43] evaluated measures obtained via the NESMA estimate method as effort predictors. They used a dataset accounting for 25 Web applications form a single company, and built regression models. They found that effort models based on measures obtained via the NESMA estimated method are effective, providing a prediction accuracy comparable to the one of IFPUG measures.

In all the aforementioned studies, effort estimation was performed via regression models.

In 2023, a new study [44] addressed the problem, using elementary effort estimation models: $Estimated\ effort = \frac{Size}{Productivity}$, where *Productivity* is the mean $\frac{Size}{Effort}$ ratio in the considered dataset, excluding the project to be estimated. *Size* and *Productivity* were evaluated using IFPUG UFP, SFP and #TF; the ISBSG dataset was used. As before, estimates based on IFPUG UFP and SFP appear similar. Some difference in the effort estimates was found only when #TF is used, or transaction complexity at the projects level is taken into account.

In conclusion, the literature seems to support the idea that using simplified FSMs instead of IFPUG UFP does not cause the accuracy of effort estimates to decrease. However, none of the papers published so far performed this kind of evaluation using ML to build effort models.

B. EFFORT ESTIMATION WITH "ALTERNATIVE" SIZE MEASURES

Functional size measurement methods can be applied independently from the formalisms and methods used to specify software requirements. Hence, they can be applied in any software development process, in principle.

Other size measures have been proposed, based on concepts and constructs that are specific of the considered software requirements specification formalisms and methods. These measures are conceived to be relatively easy to obtain; however, they are applicable only to software specifications obtained through the corresponding method.

Two process-specific measures are fairly popular: use-case points [45] and story points [46]. From our point of view, it is interesting to know is simplified FSMs could be used for effort estimation in place of use-case points or story points without loss of accuracy.

1) EFFORT ESTIMATION VIA USE-CASE POINTS

Use-case points (UCP) [45] were proposed to estimate the resources needed to develop a software system with the Objectory process [47]. The size of the specified system is first computed as a weighted sum of actors and use cases, where the weight is the "complexity" of each actor and use case. The resulting number is then "adjusted" considering technical complexity factors (mainly non-functional characteristics of the system) and environmental factors (characteristics of the developers and the development process). Parts of the Objectory process, namely the description of the responsibilities of the system via use cases, were later integrated in UML, thus contributing to the popularity of use cases and UCP.

The software systems specified via use case diagrams can be measured via Function Point Analysis, as shown by Fetcke et al. [48], who mapped the elements of UCP measurement onto the elements of FPA. As for functional size measurement, the main usage of UCP is for effort estimation; hence, it would be interesting to know whether UCP-based effort estimation is as accurate as IFPUG UFP-based estimation.

Recently, Azzeh et al. surveyed the papers dealing with the usage of UCP for effort estimation in a Systematic Literature Review [49]. According to this SLR, UCP are reported to perform better than expert judgment, and better than algorithmic models at early stage of software development. Also, several papers addressed the construction of UCP-based effort estimation models using ML techniques. However, very little was done to compare the accuracy of UCP-based effort estimates with those obtained using other measures.

In 2009, Braz and Vergilio [50] proposed a new classification of complexity for computing UCP, and the usage of fuzzy theory for gradually changing complexity. In their paper, they also compared the effort estimates obtained with their method with those obtained with UCP and with IFPUG FP. Their results show that IFPUG FP appear to support more accurate effort estimation than UCP. However, the evaluation was based on a dataset of only 5 software modules, hence it is hardly generalizable. In fact, Sholiq et al. [51] reached opposite conclusions, by analyzing a dataset of only 4 projects.

In conclusion, we were not able to find reliable evidence concerning the effort estimation accuracy level that can be achieved via UCP, especially in comparison with IFPUG UFP.

2) EFFORT ESTIMATION VIA STORY POINTS

"*Story points are units of measure for expressing an estimate of the overall effort required to fully implement a product backlog item*" [52]. That is, Story Points are not a proper size measure. They are used to represent a dimension of software that correlates to development effort, but only within a specific development organization, or even within a single team. Therefore, measures expressed in Story Points are not comparable among different organizations. Nonetheless,

TABLE 15. Overview and classification of related work.

Paper topic	References	Deals with simplified FSM	Uses ML	Compares with simplified FSM	SLR
Evaluate simple FSMs for effort estimation	[5], [6], [38], [41]–[44]	√		√	
Evaluate UCP for effort estimation	[48], [50]				
Evaluate UCP for effort estimation	[49]		√		√
Evaluate SP for effort estimation	[33]	√	√		√
Evaluate SP for effort estimation	[53]				
Survey ML for effort estimation	[10]	N/A	√		√
Survey approaches to effort estimation	[55]		√		√
Survey approaches to effort estimation	[56]	N/A	√		
Evaluate FSM for effort est. in agile processes	[31], [33]	√	√		√

it could be interesting to know if, in a given organization, Story Points are better effort predictors than FSMs.

Fernandez et al. surveyed effort estimation in agile environments via a SLR [33]. They found that Planning poker and Story Points are by far the most used effort estimation method and size metric, respectively. According to Fernandez et al., when planning poker and Story Points are used for effort estimation, the mean MMRE reported by primary studies is 41%. However, no comparison with estimates based on FSMs was reported.

Salmanoglu et al. [53] compared effort estimation activities carried out using Story Points and COSMIC Function Points, and found that the two considered measure achieve similar estimate accuracy. Although they did not consider IFPUG FP, COSMIC FP are a FSM that is quite well correlated with IFPUG FP [54], hence their results could be of some interest for our purposes. However, Salmanoglu et al. performed their evaluation on three projects only, thus it is not guaranteed that their conclusions can be generalized.

In conclusion, UCP and Story Points can be used for effort estimation in specific cases, but there is no reliable evidence-based evaluation of the difference of accuracy obtained by using these metrics instead of FSMs.

C. MACHINE LEARNING MODELS FOR EFFORT ESTIMATION

A great deal of research has been dedicated to ML-based effort estimation. Ali and Gravino performed a Systematic Literature Review of articles concerning software effort estimation using ML techniques, published up to December 2017 [10]. They observed that the most frequently used ML approaches are neural networks and SVM, while also regression techniques appear to be widely used. Overall, NN and SVM performed better than other ML approaches. In this respect, our study confirms that SVM provided good performance, while NN were not as accurate as could be expected. Although the SLR by Ali and Gravino did not mention the measures used by ML models, they investigated the datasets used for ML model building: none of these datasets includes simplified FSMs.

More recently, Rashid et al. surveyed the current approaches in effort estimation [55]. Even though they did not focus specifically on ML techniques, they accounted for multiple studies that used ML techniques. Like Ali and Gravino, Rashid et al. did not address directly the measures used to build effort estimation models; nonetheless they reported about the datasets used for model building: none of these datasets includes simplified FSMs. This situation is further supported by a SLR by Halimawan et al. [56].

D. EFFORT ESTIMATION WITH FUNCTIONAL SIZE MEASURES IN AGILE CONTEXTS

Functional size measurement is generally considered a rather heavyweight activity, which is not suitable for agile software development. This belief was confirmed by Hacaloglu and Demirors [31]: in 2018, they reported that among the reasons that prevent the wide adoption of FSMs in agile processes are i) measurement processes are cumbersome to follow, and ii) there is a mismatch between agility and the need for detailed requirements. These issues do not apply to simplified functional measures, which are perceived as lightweight, hence particularly suitable for usage in an agile context. In fact, Fernández-Diego et al. [33] report that SiFP have been used in agile processes.

E. CONCLUSIONS ABOUT RELATED WORK

Table 14 presents the most relevant related work, classified according to the dimensions that are relevant to answer our research questions: using ML techniques, dealing with simplified FSMs, performing a comparison of simplified FSMs and full-fledged FSMs with respect to effort estimation accuracy. In Table 14, we have also highlighted what papers are systematic literature reviews (SLR).

Table 15 shows that there are many papers that dealt with topics that are related—to some extent—to the goals of our work. However, none of the identified papers covers all the topics that are necessary to answer our research questions. Therefore, we can state that, to the best of our knowledge, the study reported here is the first one that (i) uses ML techniques to build effort models using full-fledged FSMs (IFPUG UFP), simplified FSMs (IFPUG SFP) and sets of extremely simple, unweighted measures (the building blocks of FSMs), and (ii) performs a comparison of the accuracy of the obtained estimates.

VIII. CONCLUSION AND FUTURE WORK

Simplified functional size measures (FSMs) have several advantages over traditional full-fledged FSMs, like IFPUG

Function Points. Specifically, they are available earlier than full-fledged metrics, because less details concerning requirements specifications are needed; because of the same reason, simplified metrics are also easier and cheaper to collect. However, it is possible that simplified FSMs neglect some information that affects development effort, so that estimates based on simplified measures are less accurate than those based on full-fledged functional measures. So, before using simplified measures for effort estimation, we need some evidence based on experimental data that we do not risk to get excessively inaccurate effort estimates.

Earlier work [6] evaluated the effort estimation accuracy that can be achieved using simplified FSMs. However, such work considered only effort models built via statistical technique. In this paper, we addressed such evaluation via an empirical study, in which effort estimation models were built using multiple Machine Learning. The study addressed three types of projects: developments from scratch (NEW), enhancements (ENH) and extensions, i.e., additions to existing software that did not requirechanges (EXT).

Using ML techniques let us extend our knowledge: for instance, we built ML models of effort for extension projects, which was not possible using plain regression [6].

Among the ML technique used (Support Vector Regression, Random Forests, Neural Networks and K-nearest Neighbors), SVR provided the most accurate models, with all the metrics and for all project types.

The results we obtained indicate that all the FSMs that were used—UFP (i.e., the traditional IFPUG standard full-fledged metric), SFP (i.e., the IFPUG standard simplified metric), and sets of basic unweighted measures—appear equally accurate. More precisely: the observed differences (which for ENH and EXT projects were in favor of simplified metrics) are very small, and practically irrelevant.

Based on our results, software project managers can consider analyzing only a small and specific part of Functional User Requirements, to get measures that support effort estimation with no appreciable differences in prediction accuracy.

The study reported in this paper considers effort models based exclusively on the size of the software to be developed or maintained. This choice was motivated by the need to limit the possible confounding effect deriving from the usage of multiple parameters. Nonetheless, it is well-known that development effort depends on several other factors, besides size, including the characteristics of software (complexity, non-functional requirements, etc.), of developers (experience, domain knowledge, code knowledge, etc.) and the development environment (tools, methods, processes, etc.). Accordingly, we plan to extend the work reported here by building more comprehensive effort models, depending on the availability of data.

We also plan to explore how Large Language Models (LLMs) can contribute to functional size measurement and effort estimation.

APPENDIX
DETAILS ON THE CONSTRUCTION OF ML MODELS

Data analysis was carried out using the R programming language and environment [57]. Specifically, we used the `e1071` library (`https://cran.r-project.org/web/packages/e1071/index.html`).

A. SUPPORT VECTOR REGRESSION (SVR) MODELS

Since the used datasets contain predictors and a dependent variable, and values are all continuous, we used a supervised and computationally not demanding method like SVR. Also the small size of the datasets was a criterion for choosing a robust approach like SVR. Finally, considering the problem at hand, we used a radial kernel.

To build models, a fundamental step was the configuration of the model with proper parameters (i.e., the so-called hyperparameters of the model). To this end, we exploited the `tune.svm` function of the `e1071` library. The `tune.svm` function was designed to find the best set of parameters for the data in a ranged or full parameter space for each parameter; we used this function passing a proper hyperparameter range for each tuning parameter:

- **cost** is a regularization parameter used when transforming mathematically the problem into a Lagrangian formulation. We provided the tuning function with the $[2^{-3},2^{6}]$ range.
- **epsilon** is the margin of tolerance for not penalizing errors. We provided the tuning function with the set of values $\{0.1, 0.01, 0.001\}$.
- **gamma** controls the distance of the influence of a single training point. Low (respectively, large) values of gamma indicate a large (respectively, small) similarity radius which results in more (respectively, fewer) points being grouped together. We provided the tuning function with the $[2^{-1},2^{3}]$ range, which does not include large gamma values that could cause overfitting.

In addition, the `tune.svm` function has a `tune.control` argument, which enables the choice of common parameters like the sampling method, the size of the bootstrap samples, the returning of the error measure, and the returning of the performance of all the parameters combined at each tuning iteration. Among the `tune.control` arguments, `cross` allows the programmer to instruct the `tuning` function to look for the best parameters via an internal `cross`-fold cross validation: we set `cross`=5.

Since the `tune.svm` function explores only a subset of the parameters space, we executed it ten times for each dataset, computing the resulting MAR; we then selected the parameters that obtained the lowest MAR.

B. NEURAL NETWORK (NN) MODELS

We used Neural Networks because this method is considered general purpose and it often outperforms other methods in common tasks such as machine translation and image

recognition, as well as in effort prediction [10]. Furthermore, it is robust to small samples, and it is sophisticated and flexible enough to provide meaningful tradeoffs between overfitting vs. underfitting model configuration. In particular, we chose this method with our dataset, to obtain independent variables weights and take their weighted combination to train an NN and see how independent variables contribute to the choice of the dependent variable value.

We used the `tune.nn` function of the `e1071` library to get the best NN model for our problem. In particular, the R `nnet package`, upon which the function `tune.nn` is built, models a single hidden layer neural network. To train the best NN for our problem We set the hyperparameters of the NN as follows:

linout for yielding a linear output instead of logistic output, which must be set to `true`.

rang is the initial random weights interval and was set to the interval [-rang, rang], with value rang=0.1.

The returning model was queried for the best parameter space of the hyperparameter `size`, which quantifies the ideal number of units in the hidden layer to be considered in the final model, and `decay`, a factor by which the minimization of the loss function procedure is affected, in that it "regularizes" the value of the node weights at each step.

As for the SVR method, also the NN method presents a `tune.control` argument, which enables the choice of common parameters like the sampling method, the size of the bootstrap samples, the returning of the error measure, and the returning of the performance of all the parameters combined at each tuning iteration. Among the `tune.control` arguments, `cross` allows the programmer to instruct the `tuning` function to look for the best parameters via an internal `cross`-fold cross validation: we set `cross`=5.

C. RANDOM FOREST (RF) MODELS

Random Forest is an evolution of the models based on Decision Trees. We chose a method of this kind for our dataset, as the independent variables values can be used as decision rules to obtain the final value of the dependent variable. In order to yield the best model based on RF for our problem we used the `tune.rf` function of the `e1071` library. As usual, the hyperparameters space was configured without any restriction, whereas the best parameter space considered for exploiting the best RF model on our dataset were:

nodesize is the number of minimum nodes in each tree to be considered.

mtry is the number of predictors considered at each node split (decision).

ntree is the number of trees in the model.

Also the RF method presents a `tune.control` argument. Among the `tune.control` arguments, we set `cross`=5.

D. GENERALIZED K-NEAREST NEIGHBOURS (GKNN) MODELS

The last method used was the so called Generalized K-Nearest Neighbour model. This model is an implementation of the k-nearest neighbour algorithm making use of general distance measures among neighbours. This value can be, for example, the average of the values of k nearest neighbors. A technique is the assignment of weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. The neighbors are taken from a set of objects for which the object property value is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. A peculiarity of the k-NN algorithm is that it is more sensitive to the local structure of the data with respect to other ML models.

The `e1071` library provided a `tune.gknn` function for configuring hyperparameters and yielding the best KNN model for the problem at hand. The function was used with the following configuration:

k is the number of neighbours to be considered. We set this value as the interval [1,3].

use.all controls handling of ties. If true, all distances equal to the k^{th} largest are included. If false, a random selection of distances equal to the k^{th} is chosen to use exactly k neighbours. We set `use.all` to `true`.

FUN is the function used to aggregate the k nearest target values in case of regression. We chose the `mean`.

Also the gKNN method presents a `tune.control` argument. Among the `tune.control` arguments, we set `cross`=5.

REFERENCES

[1] A. J. Albrecht, "Measuring application development productivity," in *Proc. Joint SHARE/GUIDE/IBM Appl. Develop. Symp.*, vol. 10, 1979, pp. 83–92.

[2] IFPUG, *Simple Function Point (SFP) Counting Practices Manual Release 2.1*, 2021.

[3] R. Meli, "Simple function point: A new functional size measurement method fully compliant with IFPUG 4.x," in *Software Measurement European Forum*, 2011, pp. 145–152.

[4] L. Lavazza, A. Locoro, G. Liu, and R. Meli, "Estimating software functional size via machine learning," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 5, pp. 1–27, Sep. 2023.

[5] L. Lavazza and R. Meli, "An evaluation of simple function point as a replacement of IFPUG function point," in *Proc. Joint Conf. Int. Workshop Softw. Meas. Int. Conf. Softw. Process Product Meas.*, Oct. 2014, pp. 196–206.

[6] L. Lavazza, G. Liu, and R. Meli, "Using extremely simplified functional size measures for effort estimation: An empirical study," in *Proc. 14th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Oct. 2020, pp. 1–9.

[7] 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.

[8] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, "Neural network models for software development effort estimation: A comparative study," *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2369–2381, Nov. 2016.

[9] P. Sharma and J. Singh, "Systematic literature review on software effort estimation using machine learning approaches," in *Proc. Int. Conf. Next Gener. Comput. Inf. Syst. (ICNGCIS)*, Dec. 2017, pp. 43–47.

[10] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods," *J. Softw., Evol. Process*, vol. 31, no. 10, p. e2211, Oct. 2019.
[11] H. D. P. De Carvalho, R. Fagundes, and W. Santos, "Extreme learning machine applied to software development effort estimation," *IEEE Access*, vol. 9, pp. 92676–92687, 2021.
[12] P. B. Ritu, "Software effort estimation with machine learning—A systematic literature review," in *Agile Software Development: Trends, Challenges and Applications*. Hoboken, NJ, USA: Wiley, 2023.
[13] *Worldwide Software Development: The Benchmark*, International Software Benchmarking Standards Group, Melbourne, VIC, Australia, 2019.
[14] A. Abran, "Data collection and industry standards: The ISBSG repository," in *Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers*. Hoboken, NJ, USA: Wiley, 2015.
[15] International Function Point Users Group (IFPUG), *Function Point Counting Practices Manual*, Release 4.3.1, 2010.
[16] L. A. Lavazza, V. Del Bianco, and C. Garavaglia, "Model-based functional size measurement," in *Proc. 2nd ACM-IEEE Int. Symp. Empirical Softw. Eng. Meas.*, Oct. 2008, pp. 100–109.
[17] L. Lavazza, "'On the effort required by function point measurement phases," *Int. J. Adv. Softw.*, vol. 10, nos. 1–2, pp. 119–178, 2017.
[18] F. González-Ladrón-de-Guevara, M. Fernández-Diego, and C. Lokan, "The usage of ISBSG data fields in software effort estimation: A systematic mapping study," *J. Syst. Softw.*, vol. 113, pp. 188–215, Mar. 2016.
[19] International Standardization Organization (ISO), *Software Engineering—IFPUG 4.1 Unadjusted Functional Size Measurement Method—Counting Practices Manual*, Standard ISO/IEC 20926:2003, 2003.
[20] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Inf. Softw. Technol.*, vol. 54, no. 8, pp. 820–827, Aug. 2012.
[21] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. New York, NY, USA: Academic, 2013.
[22] L. V. Hedges and I. Olkin, *Statistical Methods for Meta-Analysis*. New York, NY, USA: Academic, 2014.
[23] B. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1981.
[24] G. Horgan, S. Khaddaj, and P. Forte, "Construction of an FPA-type metric for early lifecycle estimation," *Inf. Softw. Technol.*, vol. 40, no. 8, pp. 409–415, Aug. 1998.
[25] R. Meli and L. Santillo, "Function point estimation methods: A comparative overview," in *Proc. FESMA*, vol. 99, 1999, pp. 6–8.
[26] NESMA—The Netherlands Software Metrics Association, *Definitions and Counting Guidelines for the Application of Function Point Analysis*, NESMA Functional Size Measurement Method Compliant to ISO/IEC 24570, version 2.1, 2004.
[27] International Standards Organisation, *Software Engineering—NESMA Functional Size Measurement Method Version 2.1—Definitions and Counting Guidelines for the Application of Function Point Analysis*, Standard ISO/IEC 24570:2005, 2005.
[28] L. Bernstein and C. M. Yuhas, *Trustworthy Systems Through Quantitative Software Engineering*, vol. 1. Hoboken, NJ, USA: Wiley, 2005.
[29] L. Santillo, M. Conte, and R. Meli, "Early & quick function point: Sizing more with less," in *Proc. 11th IEEE Int. Softw. Metrics Symp. (METRICS)*, Sep. 2005, p. 41.
[30] T. Iorio, R. Meli, and F. Perna, "Early & quick function points v3.0: Enhancements for a publicly available method," in *Proc. Softw. Meas. Eur. Forum (SMEF)*, 2007, pp. 179–198.
[31] T. Hacaloğlu and O. Demirörs, "Challenges of using software size in agile software development: A systematic literature review," in *Academic Papers at IWSM Mensura*, 2018.
[32] Common Software Measurement International Consortium (COSMIC), *COSMIC Measurement Manual for ISO 19761*, Version 5.0, 2021.
[33] M. Fernández-Diego, E. R. Méndez, F. González-Ladrón-De-Guevara, S. Abrahão, and E. Insfran, "An update on effort estimation in agile software development: A systematic literature review," *IEEE Access*, vol. 8, pp. 166768–166800, 2020.
[34] L. Lavazza, S. Morasca, and D. Tosi, "A study on the difficulty of accounting for data processing in functional size measures," *Int. J. Adv. Softw.*, vol. 8, nos. 1–2, pp. 276–287, 2015.
[35] R. Verdecchia, E. Engström, P. Lago, P. Runeson, and Q. Song, "Threats to validity in software engineering research: A critical reflection," *Inf. Softw. Technol.*, vol. 164, Dec. 2023, Art. no. 107329.
[36] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele Univ., Keele, U.K., Tech. Rep. EBSE-2007-01, 2007.
[37] C. Wohlin, M. Kalinowski, K. Romero Felizardo, and E. Mendes, "Successful combination of database search and snowballing for identification of primary studies in systematic literature studies," *Inf. Softw. Technol.*, vol. 147, Jul. 2022, Art. no. 106908.
[38] F. Ferrucci, C. Gravino, and L. Lavazza, "Simple function points for effort estimation: A further assessment," in *Proc. 31st Annu. ACM Symp. Appl. Comput.*, 2016, pp. 1428–1433.
[39] (Oct. 1, 2024). *NESMA*. [Online]. Available: https://nesma.org/themes/sizing/function-point-analysis/early-function-point-counting/?highlight=early-function-point-counting
[40] A. Timp, *uTip—Early Function Point Analysis and Consistent Cost Estimating, uTip # 03–(Version 1.0 2015/07/01)*, IFPUG, 2015.
[41] S. Ohiwa, T. Oshino, S. Kusumoto, and K. Matsumoto, "Towards an early software effort estimation based on the NESMA method (Estimated FP)," in *Proc. IT Confidence Conf. 2nd Int. Conf. IT Data Collection, Anal. Benchmarking*, 2014.
[42] J. Popovic and D. Bojic, "A comparative evaluation of effort estimation methods in the software life cycle," *Comput. Sci. Inf. Syst.*, vol. 9, no. 1, pp. 455–484, 2012.
[43] S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, "Assessing the effectiveness of approximate functional sizing approaches for effort estimation," *Inf. Softw. Technol.*, vol. 123, Jul. 2020, Art. no. 106308.
[44] L. Lavazza, A. Locoro, and R. Meli, "Software development effort estimation using function points and simpler functional measures: A comparison," in *Proc. Joint Conf. 32nd Int. Workshop Softw. Meas. (IWSM) 17th Int. Conf. Softw. Process Product Meas. (MENSURA)*, 2023, pp. 1–19.
[45] G. Karner, "Resource estimation for objectory projects," *Objective Syst. SF AB*, vol. 17, no. 1, p. 9, 1993.
[46] M. Cohn, *Agile Estimating and Planning*, London, U.K.: Pearson, 2005.
[47] I. Jacobson, *Object-Oriented Software Engineering: A Use Case Driven Approach*. London, U.K.: Pearson, 1993.
[48] T. Fetcke, A. Abran, and T.-H. Nguyen, "Mapping the OO-Jacobson approach into function point analysis," in *Proc. Int. Conf. Technol. Object Oriented Syst. Lang.*, Aug. 1997, pp. 192–202.
[49] M. Azzeh, A. Bou Nassif, and I. B. Attili, "Predicting software effort from use case points: A systematic review," *Sci. Comput. Program.*, vol. 204, Apr. 2021, Art. no. 102596.
[50] M. R. Braz and S. R. Vergilio, "Software effort estimation based on use cases," in *Proc. 30th Annu. Int. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1, Sep. 2006, pp. 221–228.
[51] R. S. Dewi and A. P. Subriadi, "A comparative study of software development size estimation method: UCPabc vs function points," *Proc. Comput. Sci.*, vol. 124, pp. 470–477, Jan. 2017.
[52] Altassian. (2020). *Story Points and Estimation*. [Online]. Available: https://www.atlassian.com/agile/project-management/estimation
[53] M. Salmanoglu, T. Hacaloglu, and O. Demirors, "Effort estimation for agile software development: Comparative case studies using COSMIC functional size measurement and story points," in *Proc. 27th Int. Workshop Softw. Meas. 12th Int. Conf. Softw. Process Product Meas.*, 2017, pp. 41–49.
[54] A. Z. Abualkishik, F. Ferrucci, C. Gravino, L. Lavazza, G. Liu, R. Meli, and G. Robiolo, "A study on the statistical convertibility of IFPUG function point, COSMIC function point and simple function point," *Inf. Softw. Technol.*, vol. 86, 1–19, Jun. 2017.
[55] C. H. Rashid, I. Shafi, J. Ahmad, E. B. Thompson, M. M. Vergara, I. de la Torre Diez, and I. Ashraf, "Software cost and effort estimation: Current approaches and future trends," *IEEE Access*, vol. 11, pp. 99268–99288, 2023.
[56] N. Halimawan, F. W. P. Dharma, and W. N. Surantha, "Model and dataset trend in software project effort estimation—A systematic literature review," *ICIC Exp. Lett.*, vol. 15, no. 10, p. 1109, 2021.
[57] R Core Team, *R: A Language and Environment for Statistical Computing*, 2015.

LUIGI LAVAZZA (Senior Member, IEEE) received the Laurea degree in electronic engineering from the Politecnico di Milano, in 1984.

From 1985 to 1990, he worked in industry. In 1990, he joined Cefriel (https://www.\linebreakcefriel.com/?lang=en). From 1996 to 2005, he was a Research Assistant with the Politecnico di Milano. Since 2005, he has been an Associate Professor with the University of Insubria, Varese, Italy. His research interests include empirical software engineering, software metrics and software quality evaluation, software project management and effort estimation, software process modeling, measurement and improvement, and open source software. He was involved in several international research projects and he also served as a reviewer for EU projects. He has served on the PC for a several of international software engineering conferences and in the editorial board for international journals. He is the co-author of over 180 scientific articles, published in international journals, and in the proceedings of international conferences or in books.

Prof. Lavazza is a member of the ACM and a fellow of the IARIA.

ANGELA LOCORO received the degree in modern literature, the degree in computer science, and the Ph.D. degree in electronic engineering, informatics, robotics and telecommunications from the University of Genoa. She is currently an Associate Professor with the Department of Economics and Management, University of Brescia. She is the author of over 60 publications, including international journals, book chapters, and international and national conference proceedings. Her research interests include data visualization, human–computer interaction, human–AI interaction, knowledge management, and information processing. She has served on the PC of a several for HCI and IS conferences.

ROBERTO MELI has been a Consultant and a Teacher in ICT, since 1987. He is currently the CEO of DPO, since 2003. He is also an Expert in project management, requirements management, and software measurement & estimation. He has been the Coordinator of the Directive Board of Italian Software Metrics Association GUFPI-ISMA for about ten years. He has also participated in the Core Team that created the COSMIC functional software measurement method. He was the Inventor of the Early & Quick Function Point estimation method and of the simple function point measurement method. He is the co-author of over 90 papers published in international journals, book chapters, and international and national conference proceedings.

Open Access funding provided by 'Università degli Studi dell'Insubria' within the CRUI CARE Agreement