Linear Programming as a Baseline for Software Effort Estimation

Federica Sarro, University College London
Alessio Petrozziello, University of Portsmouth

Software effort estimation studies still suffer from discordant empirical results (i.e., conclusion instability) mainly due to the lack of rigorous benchmarking methods. So far only one baseline model, namely Automatically Transformed Linear Model (ATLM), has been proposed yet it has not been extensively assessed. In this paper, we propose a novel method based on Linear Programming (dubbed as Linear Programming for Effort Estimation, LP4EE) and carry out a thorough empirical study to evaluate the effectiveness of both LP4EE and ATLM for benchmarking widely used effort estimation techniques. The results of our study confirm the need to benchmark every other proposal against accurate and robust baselines. They also reveal that LP4EE is more accurate than ATLM for 17% of the experiments and more robust than ATLM against different data splits and cross-validation methods for 44% of the cases. These results suggest that using LP4EE as a baseline can help reduce conclusion instability. We make publicly available an open-source implementation of LP4EE in order to facilitate its adoption in future studies.

CCS Concepts: •**Software and its engineering** → **Software creation and management;**

Additional Key Words and Phrases: Software Effort Estimation, Linear Programming, Benchmarking

1. INTRODUCTION

Software effort estimation is the process of predicting the most realistic amount of effort (usually expressed in terms of person-hours or person-month) required to develop or maintain a software project.

Due to the strategic importance to software companies of getting accurate effort estimates, over the last 30 years researchers have focused on the construction of formal models to support the engineers in this process [Jorgensen and Shepperd 2007]. As a result a variety of promising approaches have been proposed ranging from the use of statistical models (e.g., [Briand and Wieczorek 2002]) and analogy-based techniques (e.g., [Kocaguneli et al. 2012b; Kocaguneli et al. 2013; Shepperd and Schofield 2000]) to the more recent use of machine learning (e.g., [Mair et al. 2000; Mendes and Mosley 2008]), search-based approaches (e.g., [Ferrucci et al. 2009; Ferrucci et al. 2010b; Ferrucci et al. 2014; Sarro et al. 2016]) and combinations of two or more of these methods (e.g., [Corazza et al. 2010; Kocaguneli et al. 2010; Corazza et al. 2013; Kocaguneli et al. 2012a]).

Although these techniques have pushed forward the state-of-the-art, it is also true that previous research has often shown discordant empirical evidence produced by a diversity of predictors, historical datasets and methods used for the evaluation, thus bringing to the well-known "conclusion instability" phenomenon in the software effort estimation research (i.e., different sets of best effort predictors exist under various different situations) [Keung et al. 2013].

Previous work highlighted that a central role to this problem is played by three important factors: (i) the lack of usage of baseline benchmarks [Whigham et al. 2015]; (ii) the validation method and subset of the data used for models' training and testing

Author's addresses: f.sarro@ucl.ac.uk (corresponding author), alessio.petrozziello@port.ac.uk

Table I: **Requirements of a Baseline Estimation Model.**

1. Be simple to describe, implement, and interpret.
2. Be deterministic in its outcomes.
3. Be applicable to mixed qualitative and quantitative data.
4. Offer some explanatory information regarding the prediction by representing generalised properties of the underlying data.
5. Have no parameters within the modelling process that require tuning.
6. Be publicly available via a reference implementation and associated environment for execution.
7. Generally be more accurate than a random guess or an estimate based purely on the distribution of the response variable.
8. Be robust to different data splits and validation methods.
9. Do not be expensive to apply.
10. Offer comparable performance to standard methods.

[Myrtveit et al. 2005]; (iii) the evaluation measures used for the comparison [Kitchenham and Mendes 2009; Myrtveit et al. 2005; Foss et al. 2003; Mittas and Angelis 2013].

For the above reasons, a thorough comparative assessment of effort prediction models has became necessary and selecting reliable baselines for comparative benchmarking is crucial to this purpose. However, no de facto baseline benchmark either in terms of estimation models or repository is available to effort estimation studies. Only recently [Whigham et al. 2015] have outlined seven requirements (no. 1–7 in Table 1) that a baseline model should possess and discussed the possible use of an Automatically Transformed Linear Model (ATLM) as a baseline.

In this paper, we propose (and show that it is important) considering other crucial requirements in addition to those previously suggested by [Whigham et al. 2015].

First of all, we claim that a baseline model should prove to be robust to different data splits under different cross-validation methods in order to mitigate conclusion instability (requirement no. 8 in Table 1). Let us support our claim with an example: Consider the scenario where a novel estimation method M is proposed and assessed with respect to a given baseline model B by using only one validation method V and one data split D, if B is sensitive to the selection of V and D, it may occur that M outperforms B only for this particular instance of V and D. Previous studies show that this scenario is not an exceptional one, but it is actually quite common [Sigweni et al. 2016; Rodriguez et al. 2010]. Therefore, we argue that this requirement must be met by a baseline, in addition to those previously proposed by [Whigham et al. 2015], in order to qualify (and be used) as a robust benchmark.

We also augment this requirement set with two additional requirements (no. 9 and 10 in Table 1) which were originally suggested by [Chen et al. 2018] for benchmarking search-based software engineering methods and are relevant to baseline effort estimation methods too. These requirements are: “do not be expensive to apply” (which can be measured, for example, in terms of required CPU or number of evaluations depending on the approach used) and “offer comparable performance to standard methods” (indeed while we do not expect a baseline model to outperform all state-of-the-art methods, it should offer a level of performance that often approaches existing standard methods in order to be insightful [Chen et al. 2018]).

In this paper, we investigate the suitability of ATLM with respect to all these requirements and also propose a novel baseline model, dubbed as Linear Programming for Effort Estimation (LP4EE), which is as simple as the one proposed by [Whigham et al. 2015] yet is based on a strong optimization framework such as Linear Programming, and has never been used for effort estimation before.

To this end we empirically assess the effectiveness of both models for benchmarking well-known and widely used regression and analogy-based estimation methods (i.e., Classification and Regression Tree, Random Forest, Support Vector Regression, K-Nearest Neighbour) for 10 publicly available industrial datasets by using four different cross-validation methods (i.e., k-fold with k=3, 5, 10, and leave-one-out) and repeating the evaluation 30 times with different data splits, following best practice to build and assess prediction systems [Shepperd and MacDonell 2012; Whigham et al. 2015; Langdon et al. 2016].

The results of our empirical study show that both approaches satisfy the requirements to be qualified as a baseline model for effort estimation yet LP4EE provides similar or more accurate estimates than ATLM and is much less sensitive than ATLM to multiple data splits and different cross-validation methods, therefore suggesting that using LP4EE as a baseline reduces conclusion instability.

In order to facilitate the adoption of LP4EE as an estimation method and baseline model in future studies we make available a free and open-source implementation for the R environment[1].

To summarise, the contributions of our work are:

— the suggestion of three important requirements for a baseline model, in addition to the ones proposed in literature by [Whigham et al. 2015].
— a novel estimation method for effort estimation based on Linear Programming (i.e., LP4EE), which satisfies all the requirements to qualify as a baseline;
— a thorough empirical study (based on best practice for evaluating prediction models) to assess the effectiveness of our LP4EE approach and the previously proposed baseline ATLM, both in terms of estimation accuracy and robustness to different data splits and validation methods;
— the results of our empirical study revealed that both LP4EE and ATLM provide better or comparable results with respect to state-of-the-art effort estimation techniques, thus confirming the need to compare any new technique against a well-known and robust baseline which represents a method of easy usage and public availability, besides being already approved and tested, allowing a fair and adequate assessment [Whigham et al. 2015];
— we also found that the use of different data splits and cross-validation methods can bring to significantly different results, however LP4EE is less sensitive than ATLM with respect to this issue, thus suggesting that using LP4EE as a baseline reduces conclusion instability yet provides similar or more accurate estimates than ATLM;
— a freely available implementation of LP4EE[1], which aims to facilitate its adoption and a more rigorous benchmarking in subsequent effort estimation studies.

The rest of the paper describes the mathematical Linear Programming model we propose for effort estimation together with its usability and implementation (Section 2). Then we present the design (Section 3) and results (Section 4) of the empirical study we carried out to assess our proposal. We conclude the paper presenting related work on baseline models for software effort estimation (Section 5) and our final remarks (Section 6).

2. OUR PROPOSAL: LINEAR PROGRAMMING FOR EFFORT ESTIMATION (LP4EE)

Linear Programming (LP) [Nash 2000] aims to achieve the best outcome from a mathematical model with a linear objective function subject to linear equality and inequality

[1]The source code of our R script (together with the data and the results of our study) is available on the accompanying website http://www0.cs.ucl.ac.uk/staff/F.Sarro/projects/LP4EE/index.html and GitHub project https://github.com/fedsar/LP4EE.

constraints. The feasible region is given by the intersection of the constraints and the Simplex (linear programming algorithm) is able to find a point in the polyhedron where the function has the smallest value (minimisation) in polynomial time.

The model proposed for the effort estimation problem minimises the Sum of Absolute Residual (SAE), subject to an inequality constraint imposing that the effort estimated for each of the projects in the training set has to fall in R_0^+, as follows:

$$\begin{aligned} \text{minimise} \quad & \sum_{i=1}^{n} |\sum_{j=1}^{m} a_{ij} x_j - \textit{ActualEffort}_i| \\ \text{subject to} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} \;\; x_j \geq 0 \\ & x_j \geq 0, \qquad\qquad j = 1, ..., m \end{aligned} \tag{1}$$

where a_{ij} represents the coefficient of the j^{th} feature for the i^{th} project, x_j is the value of the j^{th} feature, and $\textit{ActualEffort}_i$ is the actual effort of the i^{th} project.

Due to the non-linearity of the absolute value function, the above model has been linearised as follows:

$$\begin{aligned} \text{minimise} \quad & \sum_{i=1}^{n} t_i \\ \text{subject to} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} x_j \geq 0 \\ & \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} x_j - \textit{ActualEffort}_i - t_i \leq 0 \\ & \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} x_j - \textit{ActualEffort}_i + t_i \geq 0 \\ & x_j \geq 0, \qquad\qquad j = 1, ..., m \\ & t_i \text{ free}, \qquad\qquad i = 1, ..., n \end{aligned} \tag{2}$$

Let $X_i, \forall i$ be the part of Eq. (1) wrapped in the absolute value.$\forall i$, the slack variable t_i and the following two constraints have been added to the model:

$$X_i \leq t_i$$

$$-X_i \leq t_i$$

Therefore we can have one of the following cases:

$X_i > 0$: The second constraint, $-X_i \leq t_i$, is always fulfilled as $-X_i$ is negative and t_i is implicitly ≥ 0. Since t_i is minimised by the objective function and $0 \leq X_i \leq t_i$, the first constraint, $X_i \leq t_i$, is satisfied and t_i is $abs(X)$.

$X_i < 0$: The first constraint, $X_i \leq t_i$, is always fulfilled as X_i is negative and t_i is implicitly ≥ 0. Since t_i is minimised by the objective function and $0 \leq -X_i \leq t_i$, the second constraint, $-X_i \leq t_i$, is satisfied and t_i is $abs(X)$.

$X_i = 0$: Both constraints are always fulfilled since t_i is implicitly ≥ 0. Since t_i is minimised by the objective function, $0 = X_i = t_i$. So t_i is $abs(X)$.

When a new project is presented to the model, the following equation is used to predict its effort:

$$\textit{EstimatedEffort} = a_1 x_1 + \ ... \ + a_n x_n \tag{3}$$

where x represents the value of a given project feature and a represents the corresponding coefficient evaluated by LP.

2.1. LP4EE: Model usability and the R Package

Given a dataset of past software projects (or components), where each project P_i is characterised by the vector $[x_1, .., x_n, y]$, where $x_1, ..., x_n$ are the project features (independent variables) and y is the actual effort needed to realise the project P_i (dependent variable), LP4EE automatically builds the minimisation model as defined in Eq. (2) and finds the global optima. To solve this model we used the Simplex algorithm conceived by [Dantzig 1998] and implemented in the `R` packages `linprog` and `lpSolve`[2].

We realised LP4EE as an `R` script and made it free and open source to facilitate its adoption in future effort estimation studies[3]. This script offers a simple interface to use our proposed LP4EE approach in a fully automated way.

To run the script one has to call the function `lp(trainingsetDF, testsetDF, column)`, where `trainingsetDF` and `trainingsetDF` are the training and test set data tables, respectively, and `column` is a vector of strings containing the names of the independent variables used for the construction of the estimation model and the dependent variable (e.g., `c(‘‘varA", ‘‘varB", ‘‘Effort")`), as named in the data tables. The training set and test set data tables are formed by m rows (one per software project/component), n columns, each representing an independent variable, and the $n + 1$ column representing the dependent variable (in our case the effort).

Given a training set, the training function returns the best weights found for each of the predictors on the training set. If a test set is provided, the best model found on the training set is automatically evaluated on the test set (i.e., to test its predictive ability) and the estimated and actual values for each of the target projects contained in the test set are returned together with the corresponding absolute residuals (i.e., $|\textit{ActualEffort} - \textit{EstimatedEffort}|$, which can be used by the user to compute other accuracy measures for further analysis (see Section 3.3 for the definition of some accuracy measures).

It is worth noting that the input data to LP4EE neither needs any pre-processing nor requires satisfying any assumptions; these are tasks that often imply manual effort and knowledge of some statistics which even if basic may negatively affect the predictions if not applied correctly (see e.g., [Kitchenham and Mendes 2009]). Therefore, this is a strenght of LP4EE which many other approaches do not have, in fact they require certain assumptions to hold in order to be used correctly. These approaches include simple statistical methods such as linear regression which assumes that the data satisfy at least four important criteria [Kitchenham and Mendes 2009] and ATLM which still requires manual pre-processing to handle multicollinearity although it applies automatic data transformation in order to comply with these criteria

3. EMPIRICAL STUDY DESIGN

This section presents the design of the empirical study we carried out to assess the suitability of Linear Programming as a baseline model for effort estimation: We describe the research questions, the data and techniques we experimented with to an-

[2]These packages are freely available from Cran-R (https://cran.r-project.org/web/packages/linprog/index.html, http://cran.r-project.org/src/contrib/packages.html\#lpSolve) and can be easily installed in the `R` environment by calling the functions `install.packages("linprog")` and `install.packages("lpSolve")`, and imported in your code (`require(linprog) require(lpSolve)`.

[3]The source code of our R script is available at https://github.com/fedsar/LP4EE

swer these questions, and the validation approach and evaluation criteria we used to assess the results.

3.1. Research Questions

Since we propose LP4EE as a novel estimation model, first of all we need to check whether it satisfies the seven requirements outlined by [Whigham et al. 2015] in order to qualify as a suitable baseline, together with the additional ones we have suggested herein (see Table 1 for the full list of requirements). This constitutes our first research question:

RQ1. Satisfying the Requirements of a Baseline Model: Does our proposed approach LP4EE meet the requirements of a baseline estimation model?

To answer this question we explain how LP4EE satisfies the first six quality requirements proposed by [Whigham et al. 2015] and empirically show that LP4EE meets requirement no. 7 (i.e., be able to outperform simple effort estimation methods) by comparing it with three simple methods recommended as sanity check [Mendes and Kitchenham 2004a; Shepperd and MacDonell 2012] (i.e., Mean and Median Effort and Random Guessing, which are explained in Section 3.5). Given the simplicity of these approaches (both in terms of definition/comprehensibility and ease of usage) it is clear that if LP4EE does not outperform them, it cannot qualify as a suitable baseline benchmark. Also, as suggested by [Chen et al. 2018] we empirically assess that LP4EE is not expensive to apply (requirement no. 9).

In order to establish whether our proposed approach (LP4EE) can be proposed as an accurate and robust baseline model for effort estimation, our second research question investigates its performance when used as a benchmark to asses widely used state-of-the art techniques:

RQ2. Benchmarking State-of-the-art Estimators: Can LP4EE be effectively used as a baseline model?

If we find that LP4EE is comparable, yet better than more sophisticated techniques, then we have scientific evidence to suggest that it can be adopted as the de facto estimation model for benchmarking novel proposals.

To answer RQ2 we compare the performance of LP4EE with respect to another baseline for effort estimation (i.e., ATLM), which to the best of our knowledge has been the first and (so far) the only approach recommended as a baseline model in the effort estimation community. To perform this comparison we take into account not only the estimation accuracy of both models (*requirement no. 10*), but also their stability across different data splits and validation methods (*requirement no. 8*) as it is well-known that different choices may lead to different results [Foss et al. 2003; Sigweni et al. 2016]. Thus, it is crucial that a baseline benchmark provides similar results across different data splits and cross-validation methods.

This motivates the following sub-questions. First of all we investigate to what extent LP4EE and ATLM are robust to different validation methods:

RQ2.1. Different Validation Methods: Is LP4EE (ATLM) robust to the use of different cross-validation methods?

To answer this question we repeat our experiments by using four cross-validation methods: leave-one-out (LOO), 3-fold, 5-fold, and 10-fold cross-validation. We use the Cliff's statistical test and effect size (see Section 3.4) to assess whether LP4EE and ATLM provide statistically different estimates when different validation methods are used.

Then, we investigate how robust (i.e., sensitive) these baseline models are against different data splits:

RQ2.2 Different Data Splits: Is LP4EE more robust than ATLM to the use of different data splits?:

To answer this question we compare all the algorithms in terms of the Variance Relative Error (RE*) measure (see Section 3.4) and a stability assessment is extensively performed, as suggested by [Whigham et al. 2015]. If our method will prove to be robust (i.e., provide similar results) across different runs and splits, it means that the it is less likely to be influenced by randomness, therefore fewer comparisons will be needed in future when comparing a new algorithm to our baseline model.

Last but not least we assess if our proposal provides more accurate estimates than ATLM:

RQ2.3 Accuracy: Is LP4EE more accurate than ATLM for benchmarking state-of-the-art estimation methods?

To answer this question we compare the estimates of both baseline models, LP4EE and ATLM, with those of four state-of-the art estimation methods (i.e., Classification and Regression Tree, K-Nearest Neighbour, Random Forest, Support Vector Regression) by using unbiased summary measures (i.e., Mean Absolute Error, Median Absolute Error, Standardized Accuracy), statistical significance test (i.e., Wilcoxon Test) and effect size (i.e., Vargha and Delaney's A_{12}), as detailed in Section 3.4.

3.2. Datasets

To empirically investigate our RQs we used 10 publicly available datasets (namely Albrecht+Kemerer (AK), China, Desharnais, Finnish, Kitchenam, Maxwell, Miyazaki, Nasa, Nasa93Coc, Telecom) containing a diverse sample of industrial software projects developed by a single company or several software companies [Menzies et al. 2017]. A detailed description of each of these datasets can be found in Appendix A, while in Table 2 we report the source of the data, the number of observations and the descriptive statistics of the variables used for each of the datasets. We can observe that these datasets exhibit a high degree of diversity: They differ for *number of observations* (from 18 to 499), *number and type of features* (from 1 to 17), *technical characteristics* (e.g., software projects developed in different programming languages and for different application domains, ranging from telecommunications to commercial information systems), *companies involved* (e.g., the Desharnais dataset is within-company (WC), the others are cross-company (CC)) and *geographical locations* (software projects coming from China, Canada, Finland, etc.). Furthermore, all these datasets have been widely used in several effort estimation studies (see e.g., [Sarro et al. 2016; Sarro et al. 2012b; Ferrucci et al. 2014; Kocaguneli et al. 2012a; Sigweni et al. 2016; Shepperd and Schofield 2000]).

3.3. Evaluation Criteria

Several measures have been proposed to evaluate the accuracy of a prediction model. Generally they are based on the Absolute Error (i.e., $|ActualEffort - EstimatedEffort|$). The most popular are MMRE and Pred(25) [Conte et al. 1986] but have been widely criticised [Foss et al. 2003; Kitchenham et al. 2001; Korte and Port 2008; Port and Korte 2008; Shepperd et al. 2000; Stensrud et al. 2003] for being biased towards underestimations and for behaving very differently when comparing prediction models. The use of other (more standardised) measures, such as the *Mean Absolute Error* (MAE) or *Median Absolute Error* (MdAE) and the *Standardized Accuracy* (SA) has been recommended to compare prediction models [Shepperd and MacDonell 2012; Langdon et al. 2016], while the use of the Variance Relative Error (RE*) has been suggested by [Whigham et al. 2015] to evaluate baseline models.

Table II: Descriptive statistics of the 10 datasets used in our study.

Dataset	Type	Variable	Min	Max	Mean	Std. Dev.
AK	CC	AdjFP	99.90	2307.00	782.80	549.75
(39 projects)		Effort	0.50	1107.00	97.79	188.27
China	CC	Input	0.00	9404.00	167.10	486.34
(499 projects)		Output	0.00	2455.00	113.60	221.27
		Enquiry	0.00	952.00	61.60	105.42
		File	0.00	2955.00	91.23	210.27
		Interface	0.00	1572.00	24.23	85.04
		Effort	26.00	54620 .00	3921.00	6481.00
Desharnais	WC	TeamExp	0.00	4.00	2.30	1.33
(77 projects)		ManagerExp	0.00	4.00	2.65	1.52
		Entities	7.00	386	121.54	86.11
		Transactions	9.00	661.00	162.94	146.09
		AdjustedFPs	73.00	1127.00	284.48	182.26
		Effort	546.00	23490.00	4903.95	4188.19
Finnish	CC	HW	1.00	3.00	1.26	0.64
(38 projects)		AR	1.00	5.00	2.24	1.50
		FP	65.00	1814.00	763.58	510.83
		CO	2.00	10.00	6.26	2.73
		Effort	460.00	26670.00	7678.29	7135.28
Kitchenham	CC	AFP	15.36	18140	527.70	1521.99
(145 projects)		Effort	219.00	113900.00	3113.00	9598.00
Maxwell	CC	SizeFP	48.00	3643.00	673.31	784.04
(62 projects)		Nlan	1.00	4.00	2.55	1.02
		T01	1.00	5.00	3.05	1.00
		T02	1.00	5.00	3.05	0.71
		T03	2.00	5.00	3.02	0.89
		T04	2.00	5.00	3.19	0.70
		T05	1.00	5.00	3.05	0.71
		T06	1.00	4.00	2.90	0.69
		T07	1.00	5.00	3.24	0.90
		T08	2.00	5.00	3.81	0.96
		T09	2.00	5.00	4.06	0.74
		T10	2.00	5.00	3.61	0.89
		T11	2.00	5.00	3.42	0.98
		T12	2.00	5.00	3.82	0.69
		T13	1.00	5.00	3.06	0.96
		T14	1.00	5.00	3.26	1.01
		T15	1.00	5.00	3.34	0.75
		Effort	583.00	63694.00	8223.20	10500.00
Miyazaki	CC	SCRN	0.00	281.00	33.69	47.24
(48 projects)		FORM	0.00	91.00	22.38	20.55
		FILE	2.00	370.00	34.81	53.56
		Effort	896.00	253760.00	13996.00	36601.56
Nasa	CC	Methodology	19.00	35.00	27.78	5.38
(18 projects)		Experience	6.00	21.00	15.83	3.36
		Effort	5.00	138.30	49.47	45.72
Nasa93Coc	CC	rely	0.88	1.40	1.11	0.13
(93 projects)		data	0.94	1.16	1.00	0.07
		cplx	0.85	1.65	1.18	0.15
		time	1.00	1.66	1.13	0.20
		stor	1.00	1.56	1.13	0.19
		virt	0.87	1.15	0.92	0.09
		turn	0.87	1.15	0.96	0.09
		acap	0.71	1.00	0.89	0.09
		aexp	0.82	1.13	0.93	0.06
		pcap	0.70	1.00	0.91	0.10
		vexp	0.90	1.21	1.00	0.08
		lexp	0.95	1.14	0.97	0.05
		modp	0.82	1.24	0.98	0.09
		tool	0.83	1.24	1.00	0.09
		sced	1.00	1.08	1.04	0.04
		kloc	0.90	980	94.02	133.60
		Effort	8.40	8211	624.41	1135.93
Telecom	CC	Files	3.00	284.00	110.33	91.33
(18 projects)		Effort	23.54	1115.54	284.34	264.71

MAE is unbiased (towards over or underestimation) and defined as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|ActualEffort_i - EstimatedEffort_i| \quad (4)$$

where N is the number of projects used for evaluating the performance, and $ActualEffort_i$ and $EstimatedEffort_i$ are the measured and estimated effort, respectively, for the project i. MdAE is the median of the above distribution and is generally less sensitive than MAE to extreme outliers.

SA is based on MAE and defined as follows:

$$SA = \left(1 - \frac{MAE_{P_j}}{MAE_{rguess}}\right) \cdot 100 \quad (5)$$

where MAE_{P_j} is the MAE of the approach P_j being evaluated and MAE_{rguess} is the MAE of a large number (e.g. 1,000) of random guesses (note that except for large datasets the MAE_{rguess} can be replaced by an exact MAE, $MARP0$, as suggested by [Langdon et al. 2016]). Thus, SA represents how much better P_j is than random guessing: A value close to zero means that the prediction model P_j is practically useless, performing little better than a mere random guess [Shepperd and MacDonell 2012].

The RE* is defined as follows:

$$RE^* = \frac{var(EstimatedEffort - ActualEffort)}{var(ActualEffort)} \quad (6)$$

RE* is an appropriate baseline error measure since it gives a score of 1 to a prediction model with zero variance and a score less than 1 to any useful predictor. Any model producing an RE* greater than 1 would be considered poor, regardless of the dataset [Whigham et al. 2015]. In this study, we use MAE, MdAE and SA to evaluate the accuracy of the estimates provided by the approaches we considered, and RE* to assess their stability across different data splits and validation methods.

To establish if the estimations of one method are significantly better than those provided by another method, one can test whether there is a statistically significant difference between these estimates [Kitchenham et al. 2001; Mendes et al. 2003; Stensrud and Myrtveit 1996]. We perform 30 independent runs per algorithm, per validation approach, per dataset to allow for such statistical testing, correcting for multiple statistical tests. Specifically, to answer RQ1 and RQ2, we use the Wilcoxon Signed Rank Test [Cohen 1988] since the Shapiro test [Royston 1982] showed that many of our samples came from non-normally distributed populations, making the T-test unsuitable. The Wilcoxon test is a safe test to use (even for normally distributed data), since it raises the bar for significance, by making no assumptions about underlying data distributions. In particular, we test the following Null Hypothesis: "The mean (median) absolute errors provided by the prediction model P_i are not significantly less than those provided by the prediction model P_j for the dataset D", and set the confidence limit, α, at 0.05 and applied the standard Bonferroni correction (α/K, where K is the number of hypotheses) when multiple hypotheses were tested. To summarise the results of the Wilcoxon comparisons, we use the following win-tie-loss procedure as done in previous work [Kocaguneli et al. 2012a; Sarro et al. 2017; Sarro et al. 2018]: If the distribution i is statistically significantly better (less) than j according to the Wilcoxon test we update win_i and $loss_j$, otherwise we increment tie_i and tie_j .

Since it is inadequate to merely show statistical significance alone [Arcuri and Briand 2014], we also investigate whether the effect size is worthy of interest by using the Vargha and Delaney's A_{12} non-parametric effect size measure. Indeed, as suggested in recent best practice [Arcuri and Briand 2014; Shepperd and MacDonell

2012], it is better, in cases such as ours when not all samples are normally distributed, to use a standardised measure rather than a pooled one like the Cohen's d. Given a performance measure M, the A_{12} statistic measures the probability that running algorithm A yields better M-values than running another algorithm B, based on the following formula $\hat{A}_{12} = (R_1/m - (m+1)/2)/n$, where R_1 is the rank sum of the first data group we are comparing, and m and n are the number of observations in the first and second data sample, respectively. If the two algorithms are equivalent, then $\hat{A}_{12} = 0.5$. Given the first algorithm performing better than the second, $\hat{A}_{12}$ is considered small for $0.6 \leq \hat{A}_{12} < 0.7$, medium for $0.7 < \hat{A}_{12} < 0.8$, and large for $\hat{A}_{12} \geq 0.8$, although these thresholds are somewhat arbitrary. In this case, we are always interested in *any* improvement in predictive performance, so no transformation of the $\hat{A}_{12}$ metric is needed [Neumann et al. 2015].

To assess the difference in the results achieved by the estimation methods when using different data splits and validation methods (i.e., RQs 2.1-2.2) we used the Cliff's statistical test [Cliff 1996] together with the Hochbergs method [Hochberg 1988; Hochberg and Benjamini 1990] for controlling multiple tests available through the `R` function `cidmulv2` of the package `WRS`. The Cliff's test is an appropriate choice when dealing with non-normal data as in our case. Indeed this test provides a robust, non-parametric effect size and is reliable in presence of tied values[4].

3.4. Validation Method

A validation process is required to verify whether a method produces a useful estimation of the actual development effort. Indeed, when the accuracy of the model is computed using the same dataset employed to build the prediction model, the accuracy evaluation is considered optimistic [Briand and Wieczorek 2002]. Therefore we perform a multiple-fold (i.e., *k*-fold) cross-validation by partitioning the dataset in *k* disjoint test sets (the observations are sampled uniformly at random, without replacement) and considering the remaining observations as the training set.

To ensure that our experiments are not biased by the number of folds used, we execute them by using 3-fold, 5-fold, 10-fold and leave-one-out. The Cliff's statistical test and the Hochbergs method (see Section 3.3) have been used to investigate if there are statistically significant differences among the absolute residuals obtained by both LP4EE and ATLM using these validation methods (RQ 2.1). Since we observed a statistically significant difference only when comparing k-fold to LOO (see Section 4.2.1 for more details), we report herein the results obtained using both 3-fold and LOO and make all the results (including those obtained using 5-fold and 10-fold) available on the accompanying website[1].

3.5. Estimation Techniques

3.5.1. Random Guessing. Random Guessing (RG) is a naïve benchmark suggested to assess the usefulness of a prediction system [Shepperd and MacDonell 2012]. It randomly assigns the y value of another case to the target case. More formally, it is defined as: Predict a y for the target case t by randomly sampling (with equal probability) over all the remaining $n-1$ cases and take $y = r$ where r is drawn randomly from $1...n^r = t$ [Shepperd and MacDonell 2012]. Any prediction system should outperform random guessing since an inability to predict better than random implies that the prediction system is not using any target case information.

[4]The `WRS` package is available at https://cran.r-project.org/web/packages/WRS2/index.html, while a useful guide by Prof. Barbara Kitchenham can be found at http://crest.cs.ucl.ac.uk/cow/31/.

3.5.2. Mean Effort (Median) Effort. Mean Effort and Median Effort are two naïve baselines commonly used as a benchmark in previous effort estimation studies. Specifically, the mean (median) of the past project efforts is used as the predicted effort for a target project [Mendes and Kitchenham 2004a].

3.5.3. Automatically Transformed Linear Model. The Automatically Transformed Linear Model (ATLM) has been proposed by [Whigham et al. 2015] as a baseline effort estimation model. This approach is based on a multiple linear regression of the form $y_i = \beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_n x_{ni} + \epsilon_i$, where y_i is referred to as the quantitative response variable, x_i are explanatory variables, and β_i are determined using a least squares estimator [Neter et al. 1996]. Categorial explanatory variables are handled by using the standard contrasts approach of dummy variables for each qualitative x_i which is the default method provided by the `R` tool [R Development Core Team 2011]. The approach also performs an automatic data pre-processing[5], as it is recommended to transform skewed data when forming linear models [Kitchenham and Mendes 2009], by assessing the suitability of log and square-root transformations of the response and explanatory variables based on the underlying distribution of the data. This transformation step is calculated by comparing the skewness [Dimitriadou et al. 2008] for each of the response and explanatory variables. The transformation that results in the least skewed data for each variable is selected and used when constructing the linear model and predicting effort. An appropriate inverse transformation of the predictions is applied to make the model results meaningfully compared to the untransformed test data. The details of the transformation algorithm can be found elsewhere [Whigham et al. 2015].

3.5.4. Classification and Regression Trees. Classification and Regression Trees (CART) are machine learning methods that build prediction models by recursively partitioning the data and fitting a simple prediction model within each partition [Breiman et al. 1984]. The partitioning can be graphically represented with a decision tree. Decision trees where the dependent variable takes a finite set of values are called "classification trees", while decision trees where the dependent variable takes continuous values are called "regression trees". In our work, regression trees were generated using the `R` package `tree`, which is publicly available in the Cran-R repository [6]. Since CART does not have any seed associated with it (i.e., produces the same results each time it is executed under the same configuration) only one run per data split is required.

3.5.5. K-Nearest Neighbour. K-Nearest Neighbour (KNN) is an analogy-based approach that, given a target instance (e.g., a new software project characterized by a vector of n features), retrieves the instances relevant to this target from a case base of past projects. These relevant cases are identified by using the Euclidean distance as a similarity function, which measures the distance between the target case and the other cases based on the values for the n features of these projects. The average of the effort values of the k most similar past projects is then used as the effort estimate for the target project. If there are ties for the $k-th$ nearest vectors, all are used to compute the average. The choice of k is left to the user and has been a matter of some debate [Kadoda et al. 2001]. In this work we experimented KNN with different values of $k = 1, ..., 10$. Table 3 shows the results in terms of MAE produced by the 10 KNN configurations for each of the 10 datasets under investigation (the best MAE values are highlighted in bold, the worst ones in italic). We can observe that the configuration that exhibits the worst performance for almost all datasets is KNN1. On the other hand, KNN10 achieves the most accurate results for 6 out of 10 dataset when the 3-

[5]A manual pre-processing is still needed to handle multicollinearity if two or more variables are collinear.
[6]https://cran.r-project.org/web/packages/tree/tree.pdf

Table III: **KNN configurations**: Standardised Accuracy (SA) obtained by 10 different KNN configurations for each of the 10 datasets (best values in bold, worst in italic) using 3-fold (a) and LOO (b) cross-validation methods.

(a) 3-fold

	AK	China	Desharnais	Finnish	Kitchenam	Maxwell	Miyazaki	Nasa	Nasa93Coc	Telecom
KNN1 (worst)	*0.32*	*0.29*	*0.28*	*0.35*	*0.29*	*0.37*	*0.43*	0.10	*0.20*	0.36
KNN2	0.38	0.37	0.38	0.39	0.37	0.50	0.47	**0.15**	0.30	0.36
KNN3	**0.44**	0.40	0.42	0.42	0.41	0.53	**0.48**	**0.09**	0.36	0.38
KNN4	0.43	0.42	0.42	0.42	0.43	**0.54**	**0.48**	**0.11**	0.39	**0.41**
KNN5	0.43	0.43	0.42	0.41	0.44	**0.54**	0.47	0.11	0.41	0.40
KNN6	**0.44**	0.44	0.42	0.41	0.44	0.53	0.47	0.09	0.43	0.39
KNN7	0.43	0.44	0.42	0.40	**0.45**	0.52	0.46	0.08	0.44	0.38
KNN8	0.43	0.45	0.43	0.41	**0.45**	0.51	0.46	0.06	0.46	0.36
KNN9	**0.44**	0.45	0.43	0.42	**0.45**	0.51	0.46	*0.05*	**0.47**	0.33
KNN10 (best)	**0.44**	**0.46**	**0.44**	**0.43**	**0.45**	0.50	0.45	*0.05*	**0.47**	*0.30*

(b) LOO

	AK	China	Desharnais	Finnish	Kitchenam	Maxwell	Miyazaki	Nasa	Nasa93Coc	Telecom
KNN1 (worst)	0.09	*-0.13*	*0.15*	0.22	*0.34*	*0.75*	*0.53*	-0.33	*-0.27*	**0.28**
KNN2	*0.07*	0.01	0.26	*0.21*	0.39	0.80	0.56	**-0.20**	-0.15	**0.28**
KNN3	0.22	0.07	0.34	**0.30**	0.43	0.81	0.57	-0.31	-0.01	0.20
KNN4	**0.24**	0.10	0.36	**0.30**	0.48	0.82	0.57	-0.31	0.07	*0.14*
KNN5 (best)	0.20	0.13	**0.37**	0.27	0.48	**0.83**	**0.58**	-0.37	0.06	0.20
KNN6	0.22	0.15	0.34	0.28	0.48	**0.83**	0.56	-0.31	0.10	0.26
KNN7	0.23	0.15	0.34	0.27	0.49	0.82	0.56	-0.35	0.10	**0.28**
KNN8	0.22	**0.16**	0.34	0.27	0.49	0.82	0.55	-0.32	0.13	0.25
KNN9	0.21	**0.16**	0.34	0.27	**0.50**	0.82	0.56	*-0.40*	0.12	0.21
KNN10	0.22	**0.16**	0.34	0.23	0.49	0.82	0.56	-0.38	**0.15**	0.16

fold validation is used, and KNN5 provides the most accurate results for 3 out of 10 dataset when the LOO validation is used. Therefore, we use KNN1, KNN5 and KNN10 to answer all our RQs. The package used for the KNN method is the `R` package `caret`, which is publicly available from Cran-R[7].

3.5.6. Random Forest. Random Forest (RF) [Ho 1995] is an ensemble learning method for classification and regression tasks, which constructs multiple decision trees at training time and picks as a final model the one that is the mode of the classes (classification) or the mean prediction (regression) of the individual trees. In our work, RF was trained and configured using the `train` function of the `R` package `caret`, which is publicly available in the Cran-R repository[7]. Specifically we use the `train` function performing 30 times a simple grid search with a nested 2-fold to mitigate the learner bias due to the intrinsic randomness of RF. The best model is selected on the training set according to this function and then used to predict the effort of the unseen projects contained in the test set.

3.5.7. Support Vector Regression. Support Vector Regression (SVR) is a machine learning method able to map non-linear separable patterns into a higher feature space where points of different categories are divided by a clear gap that is as wide as possible. For a regression task the aim is to minimise a loss function, maximising the support vector bounds. In our work, SVR is configured and trained using the `R` package `caret`, which is publicly available in the Cran-R repository[7]. Through this function we automatically apply a simple grid search for hyper-parameters tuning[8] and run a nested 2-fold cross-validation 30 times to handle the intrinsic SVR randomness. The best SVR prediction model built on the training set over the 30 runs is used to estimate the effort for the previously unseen test set projects.

[7]https://cran.r-project.org/web/packages/caret/caret.pdf

[8]Grid-search is one of the simplest way to tune machine learners and alternative approaches have been investigated in the context of effort estimation [Corazza et al. 2010; Corazza et al. 2013] and defect prediction [Sarro et al. 2012a; Fu et al. 2016; Tantithamthavorn et al. 2018]

3.6. Threats to Validity

Several factors can bias the validity of empirical studies. In this section we discuss the construct, conclusion and external validity threats that may affect our study.

To satisfy construct validity a study has "to establish correct operational measures for the concepts being studied" [Kitchenham et al. 1995]. This means that the study should represent to what extent the predictor and response variables precisely measure the concepts they claim to measure [Mendes et al. 2003]. Thus, the choice of the features and the way they are collected are crucial aspects. We mitigated such a threat by using real-world datasets widely used to empirically evaluate effort estimation models and excluding from these datasets all the independent variables that are not known at prediction time and therefore cannot be used for prediction purposes. To this end we read all the papers were the data was originally released/described rather than simply relying on the information available from public repositories or subsequent papers reusing this data as this might be misleading. A detailed description of this aspect can be found in Appendix A.

With regards to the conclusion validity, we carefully applied the statistical tests, verifying all the required assumptions and correcting for multiple statistical testing. To reduce conclusion instability [Menzies and Shepperd 2012], we followed recent best practice to assess prediction systems [Shepperd and MacDonell 2012; Whigham et al. 2015; Langdon et al. 2016]. Moreover, we used datasets of different sizes to mitigate the threats related to the number of projects and features in each dataset. We also compared our approach to traditional techniques using publicly available tools to allow for replications and comparisons.

To mitigate external validity threats we used a large number of software projects covering different contexts and domains, however we cannot claim that our results generalise beyond the subjects studied herein.

4. RESULTS

4.1. RQ1. Does LP4EE Satisfy the Requirements for a Baseline Model?

Linear Programming (LP) has been largely used in the optimization field [Rardin 1998] primarily due to the fact that it offers a strong mathematical framework along with an easy implementation and results interpretability. To the best of our knowledge this is the first time that LP is used for effort estimation.

In the following we describe how our proposed method, LP4EE, satisfies all the requirements a baseline estimation model should meet (as listed in Table 1).

LP4EE describes the effort estimation problem as a simple and elegant constrained linear mathematical problem (see Section 2), which is easily readable and even geometrically representable (*requirement 1*). Moreover, LP4EE performs an automatic feature selection at learning time and explicitly gives as outputs the coefficients for each of the features for the dataset under evaluation. This feature of LP not only gives easy interpretability and straightforward application of the model to new test points (*requirement 1*), but also provides users with an insight into the most important variables of the projects (*requirement 4*). Furthermore, LP4EE is deterministic in its outcome (*requirement 2*) and can be applied to real, integer, and categorical variables (*requirement 3*). It is also hyper-parameters free (*requirement 5*) and a reference implementation for the `R` environment is publicly available[1] (*requirement 6*).

In order to show that LP4EE satisfies *requirement 7* (i.e., be more accurate than a random guess or an estimate based purely on the distribution of the response variable) we empirically assessed whether it is more accurate than a random guess and also than estimates based on the Mean and Median effort of past projects (as detailed in Section 3.5). The analysis of the SA values (see Table 4) suggests that the

estimations obtained using LP4EE are always better than those achieved by using Mean ($SA_{LP4EE} > SA_{Mean}$), Median ($SA_{LP4EE} > SA_{Median}$), and Random estimates ($SA_{LP4EE} > 0$) for all the 10 datasets considered in our study. These observations are confirmed by the tests we used to assess if any statistically significant difference arises between the accuracy of the estimates provided by LP4EE and the other methods considered and the magnitude of this difference (i.e., effect size). Table 5 shows the win-tie-loss outcome of the Wilcoxon test (for both the MAE and MdAE distributions) summarised by counting the number of times LP4EE scored a $p - value < 0.001$ (win), $p - value > 0.99$ (loss) and $0.001 <= p - value <= 0.99$ (tie). We can observe that LP4EE achieves the best win-tie-loss for balance across all the measures and datasets considered. Indeed, the MAE and MdAE distributions obtained by LP4EE are always statistically significantly better than those provided by other methods with a large effect size for 21 out of 27 cases (78%) and a medium effect size for the remaining six cases (22%). For completeness, the same tests have been carried out for ATLM and the results are reported in the last three columns of Table 5. We can observe that ATLM does not always score a win and provides a worse win-tie-loss outcome with respect to LP4EE. This inferential statistical analysis confirms that our approach significantly outperforms the baselines, thereby passing the sanity check set by *requirement 7* (see Table 1).

To check if *requirement 9* holds (i.e., do not be expensive to apply), we have analysed the running time spent by LP4EE and ATLM to build an estimation model for each of the 10 datasets using four different cross-validation methods. We ran all the experiments on a notebook with an Intel Core i5 2Ghz CPU and memory of 8GB. Table VI shows the average running time over 30 runs expressed in seconds[9]: We can observe that both LP4EE and ATLM build the prediction model in less than a second (usually milliseconds) for all the datasets and validation methods we considered and therefore both approaches satisfy *requirement 9*.

Finally, we have assessed that LP4EE is robust to different data splits and validation methods (*requirement 8)* and offers comparable performance to standard methods (*requirement 10)* by carrying out a thorough empirical study, whose results are discussed in Section 4.2.

Therefore, we can positively answer our first research question RQ1: **LP4EE satisfies the requirements for a baseline estimation model**.

4.2. RQ2: Is LP4EE Effective to Benchmark State-of-the-art Estimators?

4.2.1. RQ2.1: Is LP4EE (ATLM) Sensitive to Different Cross-validation Methods?. To check to what extend our proposed method, LP4EE, is sensitive to the validation method we ran the same experiments with four different methods widely used in effort estimation, i.e., leave-one-out (LOO), 3-fold, 5-fold, and 10-fold. The Cliff's statistical test and the Hochberg's method have been used to investigate if there are statistical differences among the absolute residuals obtained by both LP4EE and ATLM using these validation methods. The resulting p-values, corrections, and effect sizes are shown in Table 7. First of all we observe that the p-values obtained by LP4EE for the comparisons among 3-, 5- and 10-fold are all above the significance threshold, so we cannot state that there are statistically significant differences in the use of these validation methods using LP4EE, while there are differences in two cases using ATLM. On the other hand, we found statistical significant differences between the use of LOO and all the considered k-fold for 25% of the comparisons using LP4EE and 27% of the cases us-

[9]For completeness we report the running time of the state-of-the-art approaches and observe that CART and KNN build method in less than a second, while RF and SVR take always more than a second and SVR takes up to 16 seconds to build a prediction model for the largest dataset (i.e., China) in our empirical study.

Table IV: **RQs1-2 (requirements 7 & 10):** Standard Accuracy (SA) obtained by the sanity check, baseline and state-of-the-art methods using 3-fold (a) and LOO (b) cross-validation for the 10 datasets considered in our study.

(a) 3-fold

		AK	China	Desharnais	Finnish	Kitchenam	Maxwell	Miyazaki	Nasa	Nasa93Coc	Telecom
Sanity Check	Mean	0.27	0.25	0.24	0.22	0.19	0.27	0.16	0.11	0.22	0.24
	Median	0.37	0.36	0.31	0.23	0.34	0.33	0.39	0.09	0.34	0.20
Baseline	LP4EE	0.48	**0.49**	**0.47**	0.41	**0.66**	**0.54**	**0.54**	**0.11**	**0.58**	0.38
	ATLM	**0.51**	0.32	0.43	0.46	0.20	0.50	0.49	-0.35	**0.58**	0.37
State-of-the-art	CART	0.35	0.39	0.36	0.43	0.22	0.46	0.20	0.06	0.44	0.32
	KNN1	0.32	0.29	0.28	0.35	0.29	0.37	0.43	0.10	0.20	0.36
	KNN10	0.44	0.46	0.44	0.43	0.45	0.50	0.45	0.05	0.47	0.30
	RF	0.41	0.45	0.43	**0.48**	0.40	0.51	0.53	0.11	0.53	0.41
	SVR	0.42	0.44	0.41	0.41	0.38	0.49	0.40	0.09	0.51	**0.42**

(b) LOO

		AK	China	Desharnais	Finnish	Kitchenam	Maxwell	Miyazaki	Nasa	Nasa93Coc	Telecom
Sanity Check	Mean	-0.01	-0.15	0.13	0.03	0.25	0.72	0.31	-0.34	-0.23	-0.03
	Median	0.14	0.03	0.21	-0.02	0.39	0.75	0.49	-0.23	-0.04	-0.25
Baseline	LP4EE	0.28	**0.22**	**0.37**	0.26	**0.69**	0.82	**0.65**	**-0.23**	0.35	0.27
	ATLM	**0.31**	-0.03	0.36	0.33	0.53	**0.84**	0.57	-0.61	**0.43**	0.17
State-of-the-art	CART	0.10	0.06	0.30	**0.47**	0.27	0.81	0.44	-0.52	0.15	0.16
	KNN1	0.09	-0.13	0.15	0.22	0.34	0.75	0.53	-0.33	-0.27	0.28
	KNN5	0.20	0.13	**0.37**	0.27	0.48	0.83	0.58	-0.37	0.06	0.20
	RF	0.18	0.15	0.35	0.39	0.44	0.82	0.62	-0.28	0.31	**0.30**
	SVR	0.23	0.14	0.35	0.29	0.44	0.82	0.53	-0.26	0.25	0.29

Table V: **RQ1. Sanity Check (requirement 7)**: Results of the Wilcoxon test and A_{12} effecti size obtained by comparing the MAE and MdAE distributions of our proposed method, LP4EE (left), with those achieved by the sanity check estimators Random, MeanEffort and MedianEffort for the 10 datasets considered in our study. For completeness, sanity check estimators are also included for ATLM (right).

(a) Wilcoxon test (3-fold)

	MAE			MdAE				MAE			MdAE		
LP4EE vs.	win	loss	tie	win	loss	tie	*ATLM vs.*	win	loss	tie	win	loss	tie
Random	10	0	0	10	0	0	Random	9	0	1	9	0	1
Mean	9	0	1	10	0	0	Mean	9	1	0	9	0	1
Median	9	0	1	9	0	1	Median	8	1	1	9	1	0
Total	28	0	2	28	0	1	Total	26	2	2	27	1	2

(b) Wilcoxon test (LOO)

	MAE			MdAE				MAE			MdAE		
LP4EE vs.	win	loss	tie	win	loss	tie	*ATLM vs.*	win	loss	tie	win	loss	tie
Random	9	1	0	10	0	0	Random	8	2	0	9	1	0
Mean	10	0	0	10	0	0	Mean	9	1	0	10	0	0
Median	10	0	0	9	1	0	Median	8	2	0	8	2	0
Total	29	1	0	29	1	0	Total	25	5	0	27	3	0

(c) Vargha and Delaney's A_{12} Effect Size (3-fold)

	MAE			MdAE				MAE			MdAE		
LP4EE vs.	large	med	small	large	med	small	*ATLM vs.*	large	med	small	large	med	small
Random	9	1	0	9	1	0	Random	9	0	1	9	0	1
Mean	6	3	1	9	1	0	Mean	5	4	1	9	0	1
Median	6	3	1	5	4	1	Median	2	6	2	6	3	1
Total	21	7	2	23	6	1	Total	16	10	4	24	3	3

(d) Vargha and Delaney's A_{12} Effect Size (LOO)

	MAE			MdAE				MAE			MdAE		
LP4EE vs.	large	med	small	large	med	small	*ATLM vs.*	large	med	small	large	med	small
Random	9	0	1	10	0	0	Random	8	0	2	9	0	1
Mean	10	0	0	10	0	0	Mean	9	0	1	10	0	0
Median	10	0	0	9	0	1	Median	8	0	2	8	0	2
Total	29	0	1	29	0	1	Total	25	0	5	27	0	3

Table VI: **RQ1. Do not be expensive to apply (requirement 9):** Running time (average seconds over 30 runs) taken by LP4EE, ATLM, CART, KNN, RF and SVR to build an estimation model for each of the 10 datasets and four cross-validation (CV) methods considered in our study.

Dataset	CV	LP4EE	ATLM	CART	KNN	RF	SVR
AK	3-fold	0.0092	0.0046	0.0024	0.0004	2.3554	1.4460
	5-fold	0.0081	0.0044	0.0021	0.0003	2.0595	1.3949
	10-fold	0.0088	0.0039	0.0022	0.0003	2.3192	1.5658
	LOO	0.0078	0.0003	0.0023	0.0004	2.6043	1.5395
China	3-fold	0.2298	0.0075	0.0036	0.0013	2.9212	16.3439
	5-fold	0.2805	0.0067	0.0034	0.0009	2.7005	18.2760
	10-fold	0.3545	0.0079	0.0039	0.0007	3.0747	22.7024
	LOO	0.4356	0.0003	0.0040	0.0005	3.6875	26.5700
Desharnais	3-fold	0.0240	0.0085	0.0031	0.0005	2.3753	3.0256
	5-fold	0.0259	0.0067	0.0028	0.0004	2.3922	3.3034
	10-fold	0.0257	0.0059	0.0026	0.0004	2.2447	3.4593
	LOO	0.0360	0.0003	0.0033	0.0005	3.0007	5.0737
Finnish	3-fold	0.0121	0.0062	0.0024	0.0004	2.2098	2.0035
	5-fold	0.0129	0.0058	0.0026	0.0004	2.3855	2.2649
	10-fold	0.0114	0.0052	0.0024	0.0003	2.1703	2.2422
	LOO	0.0874	0.0004	0.0046	0.0005	3.4779	3.2748
Maxwell	3-fold	0.0444	0.0150	0.0040	0.0007	2.8332	3.3576
	5-fold	0.0463	0.0124	0.0037	0.0006	2.7612	3.6881
	10-fold	0.0448	0.0118	0.0038	0.0006	2.4820	3.8507
	LOO	0.0635	0.0004	0.0041	0.0007	3.4517	5.0363
Miyazaki	3-fold	0.0129	0.0056	0.0026	0.0004	2.3288	1.8966
	5-fold	0.0126	0.0050	0.0025	0.0004	2.2354	1.9051
	10-fold	0.0119	0.0048	0.0023	0.0003	2.1711	2.0773
	LOO	0.0150	0.0003	0.0029	0.0004	2.9094	2.4808
Kitchenam	3-fold	0.0255	0.0043	0.0026	0.0004	2.4366	2.0846
	5-fold	0.0278	0.0036	0.0023	0.0004	2.1405	1.9956
	10-fold	0.0307	0.0038	0.0024	0.0004	2.2541	2.2662
	LOO	0.0376	0.0002	0.0026	0.0003	2.6469	2.5188
Nasa	3-fold	0.0079	0.0050	0.0028	0.0004	2.3331	1.3351
	5-fold	0.0087	0.0050	0.0026	0.0004	2.7048	1.4499
	10-fold	0.0065	0.0044	0.0021	0.0003	2.2017	1.3005
	LOO	0.0067	0.0004	0.0038	0.0004	2.5745	1.4574
Nasa93Coc	3-fold	0.0503	0.0122	0.0039	0.0007	2.4017	3.9508
	5-fold	0.0518	0.0109	0.0036	0.0006	2.4506	4.4908
	10-fold	0.0631	0.0122	0.0040	0.0007	2.7360	5.7340
	LOO	0.0652	0.0002	0.0031	0.0005	2.8387	5.3400
Telecom	3-fold	0.0077	0.0042	0.0022	0.0003	2.2351	1.3014
	5-fold	0.0077	0.0049	0.0023	0.0004	2.3310	1.4240
	10-fold	0.0058	0.0039	0.0019	0.0003	2.0089	1.2193
	LOO	0.0056	0.0003	0.0027	0.0003	2.1835	1.2479

ing ATLM[10]. The above results suggest that both LPEE and ATLM can be sensitive to different validation methods when LOO and k-fold are compared, but we did not observe any statistical significant difference when using LP4EE with different k-fold (i.e., k=3,5,10). Therefore our answer to RQ2.1 is: **LP4EE exhibits a slightly better stability than ATML against the use of different validation methods**.

4.2.2. RQ2.2: Is LP4EE More Robust than ATLM to Different Data Splits?. In order to assess to what extend LP4EE and ATLM are sensitive (i.e., robust) to different data splits we performed 30 independent cross-validation runs using each time a different data split, obtained uniformly at random, and observed if the results remain similar across these runs.

Table 8 shows the best estimators for each of the 30 splits when state-of-the-art approaches are benchmarked against ATLM (column a) or LP4EE (column b). If we count the number of times a state-of-the-art approach is the best estimator we can observe that this happens for 119 out of 300 cases (39%) when ATLM is used as a baseline, while if we use LP4EE this number is much lower, i.e. 50 cases out of 270 (16%), therefore showing that in 23% of the cases we would get a wrong answer to the

[10]Given the difference observed in the results between the k-fold and the LOO cross-validation methods, we report in this paper the results obtained using both the 3-fold and the LOO methods for all the RQs. All results (including those with 5-fold and 10-fold) are available on the accompanying website to allow for replications of our study http://www0.cs.ucl.ac.uk/staff/F.Sarro/projects/LP4EE/index.html.

Table VII: **RQ2.1. Robustness to Different Validation Methods (requirement 8)**: Results of the Cliff's statistical significance test reported in the form `p-value (p.crit) (P(X>Y))` where `p.crit` is the threshold below which the `p-value` can be considered significant according to the Hochbergs method, and `P(X>Y)` indicates the effect size (the leading zero is omitted, e.g. 0.05 is reported as .05).

(a) MAE

Technique	Dataset	LOO vs. 3-fold	LOO vs. 5-fold	LOO vs. 10-fold	3-fold vs. 5-fold	3-fold vs. 10-fold	5-fold vs. 10-fold
LP4EE	AK	.03 (.01) (.70)	.99 (.05) (.50)	.16 (.01) (.63)	.83 (.02) (.52)	.11 (.01) (.65)	.46 (.02) (.56)
	China	.48 (.01) (.57)	.72 (.02) (.53)	.72 (.01) (.53)	.68 (.01) (.53)	.99 (.05) (.50)	.89 (.02) (.49)
	Desharnais	.03 (.01) (.70)	.07 (.01) (.67)	.48 (.02) (.57)	.47 (.01) (.56)	.58 (.02) (.55)	.72 (.05) (.53)
	Finnish	.72 (.05) (.53)	.01 (.01) (.73)	.29 (.01) (.60)	.09 (.01) (.63)	.45 (.02) (.56)	.54 (.02) (.45)
	Kitchenam	.16 (.01) (.63)	.48 (.01) (.57)	.72 (.01) (.53)	.87 (.02) (.49)	.82 (.02) (.52)	.96 (.05) (.50)
	Maxwell	.29 (.01) (.60)	.48 (.01) (.57)	.48 (.01) (.57)	.62 (.02) (.54)	.54 (.02) (.55)	.83 (.05) (.52)
	Miyazaki	.00 (.01) (.13)	.29 (.02) (.40)	.03 (.01) (.70)	.35 (.02) (.41)	.03 (.01) (.70)	.40 (.05) (.57)
	Nasa	.16 (.01) (.37)	.48 (.02) (.43)	.29 (.01) (.40)	.44 (.01) (.56)	.88 (.05) (.51)	.71 (.02) (.47)
	Nasa93Coc	.01 (.01) (.27)	.29 (.01) (.40)	.72 (.02) (.53)	.62 (.02) (.46)	.48 (.01) (.56)	.79 (.05) (.52)
	Telecom	.07 (.01) (.33)	.72 (.05) (.47)	.29 (.02) (.60)	.16 (.01) (.61)	.16 (.01) (.63)	.51 (.02) (.55)
ATLM	AK	.48 (.02) (.57)	.16 (.01) (.63)	.29 (.02) (.6)	.23 (.01) (.61)	.23 (.01) (.61)	.61 (.05) (.54)
	China	.72 (.01) (.53)	.99 (.05) (.50)	.99 (.02) (.50)	.83 (.02) (.52)	.83 (.01) (.52)	.77 (.01) (.52)
	Desharnais	.00 (.01) (.23)	.29 (.01) (.60)	.48 (.02) (.57)	.08 (.01) (.64)	.36 (.02) (.58)	.87 (.05) (.51)
	Finnish	.29 (.01) (.40)	.07 (.01) (.67)	.72 (.02) (.47)	.07 (.01) (.64)	.98 (.05) (.50)	.32 (.02) (.42)
	Kitchenam	.16 (.01) (.63)	.72 (.02) (.47)	.03 (.01) (.70)	.87 (.05) (.48)	.02 (.01) (.71)	.35 (.02) (.58)
	Maxwell	.00 (.01) (.20)	.29 (.02) (.40)	.72 (.05) (.53)	.11 (.01) (.62)	.01 (.01) (.70)	.25 (.02) (.59)
	Miyazaki	.16 (.02) (.63)	.03 (.01) (.70)	.03 (.01) (.70)	.21 (.02) (.60)	.04 (.01) (.68)	.26 (.05) (.59)
	Nasa	.99 (.05) (.50)	.16 (.01) (.63)	.29 (.02) (.60)	.24 (.01) (.59)	.29 (.01) (.58)	.94 (.02) (.51)
	Nasa93Coc	.29 (.02) (.40)	.72 (.02) (.47)	.16 (.01) (.63)	.83 (.05) (.52)	.03 (.01) (.68)	.06 (.01) (.64)
	Telecom	.29 (.01) (.40)	.29 (.01) (.60)	.72 (.02) (.53)	.40 (.01) (.57)	.59 (.02) (.55)	.79 (.05) (.52)

(b) MdAE

Technique	Dataset	LOO vs. 3-fold	LOO vs. 5-fold	LOO vs. 10-fold	3-fold vs. 5-fold	3-fold vs. 10-fold	5-fold vs. 10-fold
LP4EE	AK	.03 (.01) (.70)	.99 (.05) (.50)	.48 (.02) (.43)	.51 (.02) (.45)	.09 (.01) (.36)	.32 (.01) (.42)
	China	.00 (.01) (.77)	.01 (.01) (.73)	.99 (.05) (.50)	.59 (.02) (.54)	.47 (.02) (.44)	.31 (.01) (.42)
	Desharnais	.01 (.01) (.73)	.48 (.01) (.57)	.72 (.02) (.53)	.48 (.01) (.44)	.50 (.02) (.44)	.86 (.05) (.49)
	Finnish	.07 (.02) (.33)	.00 (.01) (.23)	.00 (.01) (.17)	.80 (.05) (.48)	.06 (.01) (.35)	.07 (.02) (.35)
	Kitchenam	.72 (.02) (.47)	.48 (.01) (.43)	.48 (.01) (.43)	.41 (.01) (.43)	.94 (.05) (.51)	.75 (.02) (.53)
	Maxwell	.00 (.01) (.07)	.00 (.01) (.13)	.00 (.01) (.17)	.70 (.05) (.47)	.13 (.02) (.38)	.21 (.02) (.40)
	Miyazaki	.01 (.01) (.27)	.07 (.01) (.33)	.07 (.01) (.33)	.34 (.02) (.57)	.65 (.02) (.54)	.86 (.05) (.49)
	Nasa	.00 (.01) (.00)	.00 (.01) (.07)	.00 (.01) (.07)	.94 (.05) (.49)	.09 (.02) (.37)	.17 (.02) (.39)
	Nasa93Coc	.00 (.01) (.00)	.00 (.01) (.17)	.03 (.01) (.30)	.65 (.02) (.54)	.83 (.05) (.48)	.73 (.02) (.47)
	Telecom	.00 (.01) (.03)	.00 (.01) (.13)	.00 (.01) (.17)	.95 (.05) (.51)	.03 (.02) (.33)	.04 (.02) (.34)
ATLM	AK	.72 (.05) (.47)	.07 (.01) (.33)	.00 (.01) (.23)	.13 (.02) (.38)	.02 (.01) (.30)	.22 (.02) (.40)
	China	.99 (.05) (.50)	.29 (.01) (.40)	.16 (.01) (.37)	.53 (.02) (.45)	.28 (.01) (.41)	.64 (.02) (.46)
	Desharnais	.00 (.01) (.00)	.00 (.01) (.23)	.01 (.01) (.27)	.42 (.02) (.56)	.69 (.05) (.46)	.60 (.02) (.46)
	Finnish	.00 (.01) (.07)	.00 (.01) (.07)	.00 (.01) (.10)	.63 (.05) (.54)	.02 (.02) (.32)	.01 (.02) (.30)
	Kitchenam	.03 (.01) (.70)	.29 (.02) (.60)	.16 (.02) (.37)	.88 (.05) (.49)	.04 (.01) (.32)	.12 (.01) (.38)
	Maxwell	.16 (.01) (.63)	.29 (.02) (.40)	.48 (.05) (.43)	.27 (.01) (.41)	.23 (.01) (.40)	.46 (.02) (.44)
	Miyazaki	.00 (.01) (.23)	.03 (.01) (.30)	.07 (.01) (.33)	.29 (.02) (.41)	.22 (.02) (.40)	.50 (.05) (.45)
	Nasa	.00 (.01) (.07)	.00 (.01) (.23)	.00 (.01) (.10)	.99 (.05) (.50)	.14 (.02) (.38)	.11 (.02) (.38)
	Nasa93Coc	.00 (.01) (.01)	.07 (.01) (.33)	.16 (.02) (.37)	.03 (.01) (.67)	.23 (.02) (.60)	.60 (.05) (.46)
	Telecom	.00 (.01) (.10)	.00 (.02) (.23)	.00 (.01) (.13)	.51 (.05) (.45)	.00 (.01) (.26)	.03 (.02) (.33)

question "Does the approach x outperform the baseline model?" if we benchmark it against ATLM rather than LP4EE.

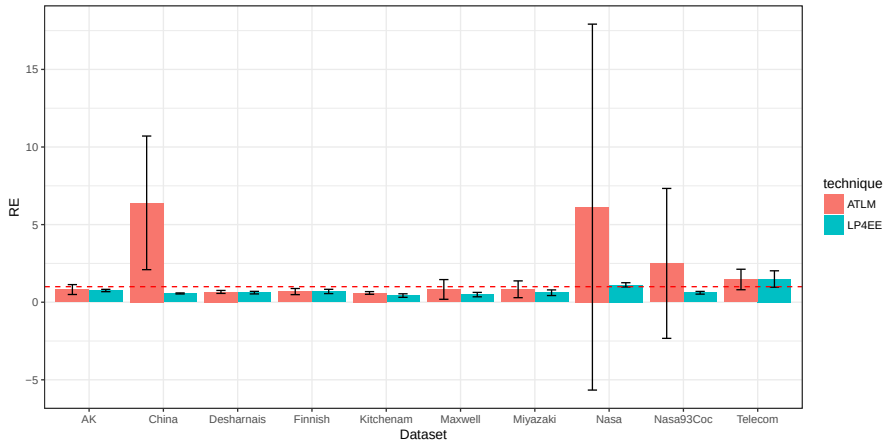
Figure 1 shows the mean RE* values and the standard deviation achieved by both ATLM and LP4EE over 30 runs for the 10 datasets considered in this study. We can observe that ATLM achieves an average RE* less than 1 and a small standard deviation only for three datasets, while its average RE* for the China, Nasa and Nasa93Coc datasets is over 5 with a high standard deviation. Differently, LP4EE achieves an average RE* value less than 1 and a small standard deviation for eight out of 10 datasets, and for the remaining two datasets (i.e., Nasa and Telecom) this value is still very close to 1 with a small standard deviation. Thus, LP4EE exhibits a much lower variability across different data splits with respect to ATLM.

The analysis of the RE* values (Table 9) confirms that our proposed method, LP4EE, is a very good estimator for almost all the splits scoring an RE$* < 1$ for 243 out of 300 cases (81%) and 9 out of 10 cases (90%) when we use the 3-fold validation and the LOO validation, respectively. Such a high number of positive scores highlights that LP4EE produces very similar results through different runs, hence it is less inclined to be good

Table VIII: **RQ2.2: Best Estimation Method** per dataset when ATLM (a) or LP4EE (b) is used as a benchmark for each of the 30 data splits (3-fold cross-validation).

	AK		China		Desharnais		Finnish		Maxwell		Miyazaki		Kitchenam		Nasa		Nasa93Coc		Telecom	
	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)
Split1	ATLM	LP4EE	KNN10	LP4EE	KNN10	KNN10	RF	RF	ATLM	LP4EE	ATLM	LP4EE	RF	RF	KNN1	KNN1	ATLM	LP4EE	KNN1	KNN1
Split2	ATLM	LP4EE	RF	LP4EE	ATLM	LP4EE	ATLM	CART	ATLM	LP4EE	ATLM	LP4EE	RF	LP4EE	KNN1	LP4EE	ATLM	LP4EE	RF	RF
Split3	ATLM	LP4EE	KNN10	LP4EE	RF	RF	RF	RF	ATLM	LP4EE	ATLM	LP4EE	ATLM	RF	SVR	SVR	ATLM	LP4EE	RF	RF
Split4	ATLM	LP4EE	KNN10	LP4EE	KNN10	KNN10	ATLM	LP4EE	ATLM	LP4EE	KNN10	KNN10	RF	RF	RF	RF	RF	LP4EE	RF	LP4EE
Split5	ATLM	KNN10	KNN10	LP4EE	KNN10	KNN10	ATLM	RF	ATLM	LP4EE	ATLM	LP4EE	RF	LP4EE	SVR	LP4EE	ATLM	LP4EE	SVR	SVR
Split6	ATLM	LP4EE	KNN10	LP4EE	KNN10	LP4EE	SVR	SVR	ATLM	LP4EE	RF	LP4EE	RF	LP4EE	KNN1	LP4EE	RF	LP4EE	RF	LP4EE
Split7	ATLM	LP4EE	KNN10	LP4EE	ATLM	LP4EE	CART	CART	ATLM	LP4EE	ATLM	RF	ATLM	LP4EE	RF	RF	ATLM	RF	KNN1	KNN1
Split8	ATLM	KNN10	KNN10	LP4EE	RF	LP4EE	RF	RF	ATLM	LP4EE	KNN10	LP4EE	ATLM	RF	RF	RF	ATLM	RF	RF	LP4EE
Split9	ATLM	LP4EE	KNN10	LP4EE	ATLM	LP4EE	RF	RF	ATLM	LP4EE	ATLM	LP4EE	RF	LP4EE	RF	RF	ATLM	LP4EE	ATLM	KNN1
Split10	ATLM	LP4EE	KNN10	LP4EE	ATLM	LP4EE	ATLM	CART	ATLM	LP4EE	ATLM	LP4EE	RF	LP4EE	KNN1	KNN1	ATLM	LP4EE	SVR	SVR
Split11	SVR	SVR	KNN10	LP4EE	RF	LP4EE	ATLM	RF	ATLM	LP4EE	KNN10	LP4EE	RF	LP4EE	SVR	LP4EE	RF	LP4EE	ATLM	RF
Split12	ATLM	LP4EE	KNN10	LP4EE	KNN10	LP4EE	RF	RF	ATLM	LP4EE	RF	LP4EE	ATLM	LP4EE	KNN1	LP4EE	ATLM	LP4EE	SVR	SVR
Split13	ATLM	LP4EE	KNN10	LP4EE	KNN10	LP4EE	ATLM	KNN10	ATLM	LP4EE	ATLM	LP4EE	ATLM	LP4EE	KNN1	KNN1	ATLM	RF	SVR	SVR
Split14	ATLM	LP4EE	KNN10	LP4EE	ATLM	LP4EE	RF	RF	ATLM	LP4EE	ATLM	LP4EE	ATLM	LP4EE	KNN10	LP4EE	RF	RF	RF	LP4EE
Split15	ATLM	LP4EE	RF	LP4EE	ATLM	LP4EE	RF	RF	ATLM	LP4EE	ATLM	RF	RF	LP4EE	KNN1	KNN1	ATLM	LP4EE	KNN1	KNN1
Split16	ATLM	LP4EE	RF	LP4EE	KNN10	LP4EE	RF	RF	ATLM	LP4EE	ATLM	LP4EE	ATLM	LP4EE	RF	RF	RF	LP4EE	RF	RF
Split17	ATLM	LP4EE	KNN10	LP4EE	RF	RF	RF	RF	KNN10	LP4EE	CART	CART	ATLM	LP4EE	RF	RF	ATLM	LP4EE	SVR	SVR
Split18	ATLM	LP4EE	KNN10	LP4EE	RF	LP4EE	RF	RF	ATLM	LP4EE	ATLM	LP4EE	ATLM	LP4EE	SVR	LP4EE	ATLM	LP4EE	RF	LP4EE
Split19	ATLM	LP4EE	KNN10	LP4EE	KNN10	LP4EE	RF	RF	ATLM	LP4EE	KNN10	LP4EE	ATLM	LP4EE	KNN10	LP4EE	SVR	LP4EE	SVR	SVR
Split20	ATLM	LP4EE	KNN10	LP4EE	KNN10	LP4EE	ATLM	RF	ATLM	LP4EE	SVR	LP4EE	RF	LP4EE	KNN1	KNN1	SVR	LP4EE	SVR	SVR
Split21	ATLM	LP4EE	RF	LP4EE	ATLM	LP4EE	ATLM	SVR	ATLM	LP4EE	CART	LP4EE	RF	LP4EE	KNN1	LP4EE	SVR	LP4EE	KNN1	KNN1
Split22	ATLM	LP4EE	KNN10	LP4EE	ATLM	LP4EE	RF	RF	ATLM	LP4EE	ATLM	LP4EE	RF	LP4EE	KNN1	LP4EE	ATLM	LP4EE	RF	RF
Split23	ATLM	LP4EE	KNN10	LP4EE	KNN10	KNN10	RF	RF	ATLM	LP4EE	ATLM	LP4EE	RF	RF	SVR	SVR	ATLM	LP4EE	SVR	SVR
Split24	ATLM	LP4EE	KNN10	LP4EE	KNN10	LP4EE	RF	RF	ATLM	LP4EE	CART	CART	ATLM	LP4EE	RF	LP4EE	RF	RF	RF	RF
Split25	ATLM	LP4EE	RF	LP4EE	ATLM	LP4EE	CART	CART	ATLM	LP4EE	ATLM	RF	ATLM	RF	RF	LP4EE	ATLM	LP4EE	SVR	SVR
Split26	ATLM	LP4EE	KNN10	LP4EE	KNN10	LP4EE	ATLM	RF	ATLM	LP4EE	RF	LP4EE	RF	LP4EE	KNN1	KNN1	RF	LP4EE	SVR	LP4EE
Split27	ATLM	LP4EE	KNN10	LP4EE	KNN10	LP4EE	RF	RF	ATLM	LP4EE	ATLM	LP4EE	ATLM	RF	RF	RF	ATLM	LP4EE	SVR	LP4EE
Split28	ATLM	LP4EE	KNN10	LP4EE	KNN10	LP4EE	RF	RF	ATLM	LP4EE	CART	LP4EE	ATLM	LP4EE	SVR	LP4EE	ATLM	LP4EE	SVR	SVR
Split29	ATLM	LP4EE	KNN10	LP4EE	RF	LP4EE	RF	RF	ATLM	LP4EE	KNN10	LP4EE	ATLM	RF	SVR	SVR	ATLM	LP4EE	ATLM	RF
Split30	ATLM	LP4EE	KNN10	LP4EE	KNN10	LP4EE	ATLM	SVR	ATLM	LP4EE	RF	RF	RF	RF	RF	RF	ATLM	LP4EE	CART	LP4EE

Fig. 1: **RQ 2.2. Robustness to Different Data Splits (requirement 8)**: Mean (height) and standard deviation (bar) of RE* (over 30 runs) achieved by ATLM and LP4EE per dataset. The red dotted line represents the RE* threshold value 1.



due to the randomness of the data splits[11]. On the other hand, ATLM achieved a total score of 187 out of 300 (62%) and 7 out of 10 (70%) using 3-fold and LOO, respectively. Therefore, we can claim that LP4EE is more robust than ATLM for 39% of the cases (i.e., 19% and 20% for 3-fold and LOO, respectively).

These observations are confirmed by the inferential statistical analysis, whose the results are shown in Table 10 in form of win-loss-tie summary and large-medium-small summary for the Wilcoxon statistical test of the RE* distributions and the A_{12} effect size, respectively. The improvement of LP4EE over the five state-of-the-art techniques is significant ($p < 0.001$) for 78 out of 100 cases (78%) and worse for only one (1%), with effect sizes large (60%), medium (23%) and small (22%). On the other hand, the RE*

[11]It is not our aim to compare LP4EE (ATLM) with the state-of-the-art methods considered in our study, however for completeness we also report their RE* values in Table 9 and it is interesting to observe that LP4EE is also more robust than all the state-of-the-art methods but KNN10 and SVR when a 3-fold validation is used and provides always same or better RE* values when a LOO validation is used.

Table IX: **RQ2.2: Robustness to Different Data Splits (requirement 8)**: RE^* results. Columns represent estimation techniques, rows represent datasets. The value in each cell counts how many times (out of 30 independent runs for 3-fold, and out of one run for LOO) $RE* < 1$ for a given estimator (i.e., since $RE* \geq 1$ indicates a poor estimator, we count how many times an approach is actually good).

(a) 3-fold

	LP4EE	ATLM	CART	KNN1	KNN10	RF	SVR
AK	30	26	6	4	17	17	19
China	30	0	29	4	30	30	30
Desharnais	30	30	23	3	30	30	30
Finnish	29	27	27	17	28	30	29
Kitchenam	30	29	0	0	30	26	26
Maxwell	30	23	26	13	30	29	30
Miyazaki	28	24	0	22	29	29	27
Nasa	0	0	2	1	0	7	11
Nasa93Coc	30	20	20	1	28	29	30
Telecom	6	8	15	18	25	13	27
Total	243	187	148	83	247	240	259

(b) LOO

	LP4EE	ATLM	KNN1	KNN5	CART	RF	SVR
AK	1	1	1	1	1	1	1
China	1	1	0	1	1	1	1
Desharnais	1	1	1	1	1	1	1
Finnish	1	1	1	1	1	1	1
Kitchenam	1	0	1	1	1	1	1
Maxwell	1	1	1	1	1	1	1
Miyazaki	1	1	1	1	1	1	1
Nasa	0	0	0	0	0	0	0
Nasa93Coc	1	1	1	1	1	1	1
Telecom	1	1	1	1	1	1	1
Total	9	8	8	9	9	9	9

values achieved by ATLM are statistically significant better than the others for only 58 of 100 cases (58%) and also statistically significantly worse in 30 of 100 cases (30%) with large (38%), medium (20%) and small (42%) effect sizes, therefore suggesting that ATLM's estimates are much more sensitive than LP4EE to the composition of the data splits, which is an unwanted behaviour for a baseline benchmark.

The above results allow us to positively answer RQ2.2: **LP4EE is more robust than ATLM against the use of different data splits**.

4.2.3. RQ2.3: Is LP4EE More Accurate than ATLM?. The analysis of the SA measure (Table 4) reveals that our proposed algorithm LP4EE provides better results than ATLM for 8 out of 10 (3-fold validation) and 6 out of 10 (LOO validation) datasets (i.e., 70% of the cases) and is comparable for the remaining ones. LP4EE also outperforms almost all the state-of-the-art techniques against which we compare it, in fact the SA values provided by LP4EE are the highest in 60% of the cases.

These observations are confirmed by the inferential statistical analysis we have carried out, whose results are summarised in Table 11 in the form of "win-loss-tie" for the Wilcoxon test results and "large-medium-small" for the A_{12} effect sizes (for both the 3-fold and LOO validation methods). We can observe that LP4EE produces estimates significantly more accurate ($p < 0.001$) than the five state-of-the-art techniques both in terms of MAE and MdAE for 143 out of 200 cases (71%) and worse only for 20 out of 200 cases (10%); no statistical difference was found in the remaining 19% of the cases. The same analysis for ATLM reveals that it is significantly better than the state-of-the-art only for 126 out of 200 cases (63%) and worse for 40 out of 200 (20%); no significant difference was found in the remaining 48% of the cases. The practical significance of these differences is overall higher for LP4EE than ATLM, indeed LP4EE achieved 55% large, 18% medium and 28% small effect sizes, while ATLM achieved 47% large, 16% medium and 37% small effect sizes.

Table X: **RQ2.2 Robustness to Different Data Splits (requirement 8):** Results of the Wilcoxon test and A_{12} effect size obtained by comparing the RE* distributions of LP4EE (left) and ATLM (right) with those of the state-of-the-art techniques.

(a) Wilcoxon Test (3-fold)

LP4EE vs.	win	loss	tie	*ATLM vs.*	win	loss	tie
CART	7	0	3	CART	5	2	3
KNN1	9	0	1	KNN1	7	1	2
KNN10	8	0	2	KNN10	6	2	2
RF	5	1	4	RF	4	3	3
SVR	6	0	4	SVR	6	2	2
Total	35	1	14	Total	28	10	12

(b) Wilcoxon Test (LOO)

LP4EE vs.	win	loss	tie	*ATLM vs.*	win	loss	tie
CART	9	1	0	CART	6	4	0
KNN1	10	0	0	KNN1	7	3	0
KNN5	8	2	0	KNN5	6	4	0
RF	8	2	0	RF	5	5	0
SVR	8	2	0	SVR	6	4	0
Total	43	7	0	Total	20	20	0

(c) Vargha and Delaney's A_{12} effect size (3-fold)

LP4EE vs.	large	med	small	*ATLM vs.*	large	med	small
CART	3	4	3	CART	2	3	5
KNN1	6	3	1	KNN1	4	3	3
KNN10	0	8	2	KNN10	0	6	4
RF	1	4	5	RF	1	3	6
SVR	2	4	4	SVR	1	5	4
Total	12	23	15	Total	8	20	22

(d) Vargha and Delaney's A_{12} effect size (LOO)

LP4EE vs.	large	med	small	*ATLM vs.*	large	med	small
CART	9	0	1	CART	6	0	4
KNN1	10	0	0	KNN1	7	0	3
KNN5	8	0	2	KNN5	6	0	4
RF	8	0	2	RF	5	0	5
SVR	8	0	2	SVR	6	0	4
Total	43	0	7	Total	30	0	20

The above results suggest that our proposed method LP4EE has increased the overall performance over ATLM in 28% of the cases when benchmarking widely used effort estimation approaches. Therefore, we can positively answer to RQ2.3: **LP4EE is more accurate than ATLM when benchmarking state-of-the art estimators**.

5. RELATED WORK: BASELINE MODELS FOR SOFTWARE EFFORT ESTIMATION

Many researchers in the artificial intelligence community have strongly advocated comparing novel prediction systems against simpler existing alternatives [Cohen 1995], however the use of baseline models in effort estimation studies is not a common practice yet. This may be due to the fact that very few studies have proposed baseline/benchmarks and the majority focused on how to compare prediction systems and how to use suitable statistic measures (see e.g., [Kitchenham et al. 2001; Mittas and Angelis 2013; Shepperd and MacDonell 2012; Langdon et al. 2016]).

Previous work proposing regression models [Mendes and Kitchenham 2004a; 2004b; Mittas and Angelis 2012] and model comparison frameworks [Mittas et al. 2015] for effort estimation used the mean (or median) of past project efforts as a naive benchmark arguing that if a novel estimation method is not able to outperform these two benchmarks it cannot be transferred to industry [Mendes and Kitchenham 2004a; 2004b].

Table XI: **RQ2.3 Accuracy (requirement 10):** Results of the Wilcoxon test and A_{12} effect sizes obtained by comparing the MAE and MdAE distributions of LP4EE (left) and ATLM (right) with those of the state-of-the-art techniques.

(a) Wilcoxon test (3-fold)

	MAE			MdAE				MAE			MdAE		
LP4EE vs.	win	loss	tie	win	loss	tie	*ATLM vs.*	win	loss	tie	win	loss	tie
CART	8	0	2	7	0	3	CART	5	1	4	6	0	4
KNN1	7	0	3	7	0	3	KNN1	8	1	1	7	2	1
KNN10	5	0	5	6	0	4	KNN10	1	1	8	7	0	3
RF	4	1	5	5	1	4	RF	3	2	5	5	1	4
SVR	5	0	5	7	0	3	SVR	5	2	3	7	1	2
Total	29	1	20	32	1	17	Total	22	7	21	32	4	14

(b) Wilcoxon test (LOO)

	MAE			MdAE				MAE			MdAE		
LP4EE vs.	win	loss	tie	win	loss	tie	*ATLM vs.*	win	loss	tie	win	loss	tie
CART	9	1	0	8	2	0	CART	7	3	0	8	2	0
KNN1	9	1	0	8	2	0	KNN1	8	2	0	8	2	0
KNN5	8	2	0	8	2	0	KNN5	5	5	0	9	1	0
RF	8	2	0	6	4	0	RF	5	5	0	7	3	0
SVR	8	2	0	10	0	0	SVR	7	3	0	8	2	0
Total	42	8	0	40	10	0	Total	32	18	0	40	11	0

(b) Vargha and Delaney's A_{12} effect size (3-fold)

	MAE			MdAE				MAE			MdAE		
LP4EE vs.	large	med	small	large	med	small	*ATLM vs.*	large	med	small	large	med	small
CART	2	6	2	3	4	3	CART	2	3	5	4	2	4
KNN1	5	2	3	5	2	3	KNN1	1	7	2	6	1	3
KNN10	0	5	5	2	4	4	KNN10	0	1	9	3	4	3
RF	1	3	6	3	2	5	RF	0	3	7	2	3	5
SVR	1	4	5	3	4	3	SVR	0	5	5	4	3	3
Total	9	20	21	16	16	18	Total	3	19	28	19	13	18

(d) Vargha and Delaney's A_{12} effect size (LOO)

	MAE			MdAE				MAE			MdAE		
LP4EE vs.	large	med	small	large	med	small	*ATLM vs.*	large	med	small	large	med	small
CART	9	0	1	8	0	2	CART	7	0	3	8	0	2
KNN1	9	0	1	8	0	2	KNN1	8	0	2	8	0	2
KNN5	8	0	2	8	0	2	KNN5	5	0	5	9	0	1
RF	8	0	2	6	0	4	RF	5	0	5	7	0	3
SVR	8	0	2	10	0	0	SVR	7	0	3	8	0	2
Total	42	0	8	40	0	10	Total	32	0	18	40	0	10

Lately [Shepperd and MacDonell 2012] proposed the use of random guessing as a naïve benchmark to assess the usefulness of a prediction system. [Whigham et al. 2015] have been the first ones to outline a preliminary set of requirements (no. 1 to 7 in Table 1) for a baseline estimation model. Based on these criteria [Whigham et al. 2015] proposed the use of Automatically Transformed Linear Model (ATLM) as a baseline, and empirically showed that more sophisticated effort estimation approaches published in literature were outperformed by this simpler approach yet they were not benchmarked against any baseline when proposed.

In this paper we have refined the guidelines of [Whigham et al. 2015] by suggesting an additional requirement, i.e., *be robust to different data splits and validation methods*, which we argue is crucial to mitigate the conclusion instability in effort estimation studies, and also including two requirements originally suggested for search-based baseline methods [Chen et al. 2018] as they are also relevant to effort estimation models (i.e., *do not be expensive to apply* and *offer comparable performance to standard methods.*). Moreover, we have extended the study of [Whigham et al. 2015] by empirically evaluating the effectiveness of ATLM for 10 publicly available industrial datasets and by proposing a novel baseline model based on Linear Programming (see Section 2), which has proved to be more robust and accurate than ATLM in benchmarking state-of-the-art estimation methods according to the results discussed in Section 4.

6. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we have proposed a novel baseline model for effort estimation based on Linear Programming (dubbed as LP4EE) and validated its effectiveness by carrying out a thourough empirical study, which follows best practice for a robust assessment of estimation models and includes a comparison with a previous baseline model, namely ATLM [Whigham et al. 2015]. In particular, we assessed the effectiveness of both LP4EE and ATLM when benchmarking widely used regression and analogy-based estimation methods (i.e., CART, KNN, RF, SVR) by using 10 publicly available industrial datasets, 30 independent runs (i.e., using different data splits) and four cross-validation methods (i.e., 3-fold, 5-fold, 10-fold, LOO), for a total of 2,000 ($10 * 30 * 4$) scenarios.

The results of our study show that both ATLM and LP4EE provide comparable or better results than the more sophisticated state-of-the-art techniques for all the datasets considered, therefore confirming the need to benchmark every other proposals against robust baseline benchmarks. Moreover, our results reveal that LP4EE is more accurate and also more robust (i.e., it provides stable results against different data splits and validation methods) than ATLM.

The positive results we have found for our novel approach to predictive modelling benchmarking are very encouraging and suggest that using LP4EE as a baseline could reduce the conclusion instability still widely observed in effort estimation studies.

To facilitate the adoption of LP4EE, we made a reference implementation freely available for the `R` environment[1], which is a very popular and widely used free software environment for statistical computing.

We believe that the results presented in this paper together with a ready-to-use baseline tool can foster more rigorous benchmarking in future effort estimation studies. Moreover, future work could investigate the effectiveness of LP for different effort estimation scenarios (e.g., cross- vs. within-company [Mendes et al. 2014; Minku et al. 2015], chronological estimation [MacDonell and Shepperd 2003; Ferrucci et al. 2014; Sigweni et al. 2016], web effort estimation [Di Martino et al. 2011; Ferrucci et al. 2012; Di Martino et al. 2016]) as well as for different estimation tasks in software engineering (e.g., predicting apps' rating [Sarro et al. 2018] and size [Ferrucci et al. 2015a; 2015b], bug-fixing time [Bhattacharya and Neamtiu 2011; Zhang et al. 2013]) or in other domains. Besides, future studies could extend the LP model we proposed herein, for example to consider different and multiple optimisation functions [Ferrucci et al. 2010a; Sarro et al. 2016].

REFERENCES

A. J. Albrecht and J. E. Gaffney. 1983. Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE TSE* SE-9, 6 (1983), 639–648.

A. Arcuri and L. Briand. 2014. A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *STVR* 24, 3 (2014), 219–250.

J. W Bailey and V. R Basili. 1981. A meta-model for software development resource expenditures. In *Proc. of ICSE'81*. IEEE Press, 107–116.

P. Bhattacharya and I. Neamtiu. 2011. Bug-fix Time Prediction Models: Can We Do Better?. In *Proc. of MSR'11*. ACM, 207–210.

B. W. Boehm. 1981. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Wadsworth Publishing Company, Belmont, California, U.S.A.

L. C. Briand and I. Wieczorek. 2002. Software esource estimation. *Encyclopedia of Software Engineering* (2002), 1160–1196.

J. Chen, V. Nair, R. Krishna, and T. Menzies. 2018. "Sampling" as a Baseline Optimizer for Search-based Software Engineering. *IEEE TSE* (2018). DOI:http://dx.doi.org/10.1109/TSE.2018.2790925

N. Cliff. 1996. *Ordinal Methods for Behavioral Data Analysis*. L. Erlbaum Associates Inc, New Jersey.

J. Cohen. 1988. *Statistical power analysis for the behavioral sciences* (2nd edition ed.). L. Earlbaum Associates.

P. R. Cohen. 1995. *Empirical Methods for Artificial Intelligence*. MIT Press, Cambridge, MA, USA.

D. Conte, H.E. Dunsmore, and V.Y. Shen. 1986. *Software engineering metrics and models*. The Benjamin/Cummings Publishing Company, Inc.

A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, F. Sarro, and E. Mendes. 2010. How Effective is Tabu Search to Configure Support Vector Regression for Effort Estimation?. In *Proc. of PROMISE'10*.

A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, F. Sarro, and E. Mendes. 2013. Using tabu search to configure support vector regression for effort estimation. *ESE* 18, 3 (2013), 506–546.

G. B. Dantzig. 1998. *Linear programming and extensions*. Princeton university press.

J. M. Desharnais. 1989. *Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction*. Ph.D. Dissertation. Unpublished Masters Thesis, University of Montreal.

S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro. 2011. Using Web Objects for Development Effort Estimation of Web Applications: A Replicated Study. In *Proc. of PROFES'11*. 186–201.

S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro. 2016. Web Effort Estimation: Function Point Analysis vs. COSMIC. *IST* 72 (2016), 90 – 109. DOI:http://dx.doi.org/https://doi.org/10.1016/j.infsof.2015.12.001

E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, and A. Weingessel. 2008. Misc functions of the Department of Statistics (e1071), TU Wien. *R package* 1 (2008), 5–24.

F. Ferrucci, C. Gravino, R. Oliveto, and F. Sarro. 2009. Using Tabu Search to Estimate Software Development Effort. In *Proc. of MENSURA'09*. LNCS 5891 Springer, 307–320.

F. Ferrucci, C. Gravino, R. Oliveto, and F. Sarro. 2010a. Genetic Programming for Effort Estimation: An Analysis of the Impact of Different Fitness Functions. In *Proc. of SSBSE'10*. 89–98.

F. Ferrucci, C. Gravino, R. Oliveto, F. Sarro, and E. Mendes. 2010b. Investigating Tabu Search for Web Effort Estimation. In *Proc. of EUROMICRO-SEAA'10*. 350–357.

F. Ferrucci, C. Gravino, P. Salza, and F. Sarro. 2015a. Investigating Functional and Code Size Measures for Mobile Applications. In *Proc. of EUROMICRO-SEAA'15*. 365–368. DOI:http://dx.doi.org/10.1109/SEAA.2015.23

F. Ferrucci, C. Gravino, P. Salza, and F. Sarro. 2015b. Investigating Functional and Code Size Measures for Mobile Applications: A Replicated Study. In *Proc. of PROFES'15*. 271–287. DOI:http://dx.doi.org/10.1007/978-3-319-26844-6_20

F. Ferrucci, C. Gravino, and F. Sarro. 2014. Exploiting Prior-phase Effort Data to Estimate the Effort for the Subsequent Phases: A Further Assessment. In *Proc. of PROMISE'14*. ACM, 42–51. DOI:http://dx.doi.org/10.1145/2639490.2639509

F. Ferrucci, M. Harman, and F. Sarro. 2014. Search-Based Software Project Management. In *Software Project Management in a Changing World*, Gnther Ruhe and Claes Wohlin (Eds.). Springer Berlin Heidelberg, 373–399.

F. Ferrucci, E. Mendes, and F. Sarro. 2012. Web Effort Estimation: The Value of Cross-company Data Set Compared to Single-company Data Set. In *Proc. of PROMISE'12*. ACM, New York, NY, USA, 29–38. DOI:http://dx.doi.org/10.1145/2365324.2365330

T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. 2003. A Simulation Study of the Model Evaluation Criterion MMRE. *IEEE TSE* 29, 11 (2003), 985–995.

W. Fu, T. Menzies, and X. Shen. 2016. Tuning for software analytics: Is it really necessary? *IST* 76 (2016), 135–146.

T. Kam Ho. 1995. Random decision forests. In *Proc. of 3rd Int. Conf. on Document Analysis and Recognition*, Vol. 1. 278–282. DOI:http://dx.doi.org/10.1109/ICDAR.1995.598994

Y. Hochberg. 1988. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika* 75, 4 (1988), 800. DOI:http://dx.doi.org/10.1093/biomet/75.4.800

Y. Hochberg and Y. Benjamini. 1990. More powerful procedures for multiple significance testing. *Statistics in Medicine* 9, 7 (1990), 811–818. DOI:http://dx.doi.org/10.1002/sim.4780090710

M. Jorgensen and M. Shepperd. 2007. A Systematic Review of Software Development Cost Estimation Studies. *IEEE TSE* 33, 1 (2007), 33–53.

G. Kadoda, M. Cartwright, and M. Shepperd. 2001. Issues on the Effective Use of CBR Technology for Software Project Prediction. In *Proc. of ICCBR'01*. LNCS, Vol. 2080. 276–290.

G. Kadoda and M. Shepperd. 2001. Using Simulation to Evaluate Predictions Techniques. In *Proc. of METRICS'01*. IEEE press, 349–358.

C. F. Kemerer. 1987. An empirical validation of software cost estimation models. *Communications of ACM* 30, 5 (1987), 416–429.

J. Keung, E. Kocaguneli, and T. Menzies. 2013. Finding conclusion stability for selectingthe best effort predictor in software effort estimation. *ASE* 20, 4 (2013), 543–567.

B. Kitchenham and E. Mendes. 2009. Why comparative effort prediction studies may be invalid. In *Proc. of PROMISE'09*. 1–4.

B. Kitchenham, S. L. Pfleeger, B. McColl, and S. Eagan. 2002. An Empirical Study of Maintenance and Development Estimation Accuracy. *JSS* 64, 1 (2002), 57–77.

B. Kitchenham, L. Pickard, and S.L. Pfleeger. 1995. Case studies for method and tool evaluation. *IEEE Software* 12, 4 (1995), 52–62.

B. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd. 2001. What accuracy statistics really measure. *IEEE Proc. Software* 148, 3 (2001), 81–85.

E. Kocaguneli, T. Menzies, A. Bener, and J.W. Keung. 2012b. Exploiting the Essential Assumptions of Analogy-Based Effort Estimation. *IEEE TSE* 38, 2 (2012), 425–438.

E. Kocaguneli, T. Menzies, J. Keung, D. Cok, and R. Madachy. 2013. Active learning and effort estimation: Finding the essential content of software effort estimation data. *IEEE TSE* 39, 8 (2013), 1040–1053.

E. Kocaguneli, T. Menzies, and J. W. Keung. 2012a. On the Value of Ensemble Effort Estimation. *IEEE TSE* 38, 6 (2012), 1403–1416.

E. Kocaguneli, A. Tosun, and A. Bener. 2010. AI-Based Models for Software Effort Estimation. In *Proc. of EUROMICRO-SEAA'10*. 323–326.

M. Korte and D. Port. 2008. Confidence in Software Cost Estimation Results Based on MMRE and Pred. In *Proc. of PROMISE'08*. 63–70.

W. B. Langdon, J. Dolado, F. Sarro, and M. Harman. 2016. Exact Mean Absolute Error of Baseline Predictor, MARP0. *IST* 73 (2016), 16 – 18.

Stephen G. MacDonell and Martin J. Shepperd. 2003. Using Prior-Phase Effort Records for Re-estimation During Software Projects. In *Proc. of METRICS '03*. IEEE Computer Society, 73–86.

C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, and S. Webster. 2000. An Investigation of Machine Learning Based Prediction Systems. *JSS* 53, 1 (2000), 23–29.

K. Maxwell. 2002. *Applied Statistics for Software Managers*. Software Quality Institute Series, Prentice Hall.

E. Mendes, S. Counsell, N. Mosley, C. Triggs, and I. Watson. 2003. A Comparative Study of Cost Estimation Models for Web Hypermedia Applications. *ESE* 8, 23 (2003), 163–196.

E. Mendes, M. Kalinowski, D. Martins, F. Ferrucci, and F. Sarro. 2014. Cross- vs. Within-company Cost Estimation Studies Revisited: An Extended Systematic Review. In *Proc. of EASE'14*. ACM, 12:1–12:10. DOI:http://dx.doi.org/10.1145/2601248.2601284

E. Mendes and B. Kitchenham. 2004a. A Comparison of Cross-company and Within-company Effort Estimation Models for Web Applications. In *Proc. of EASE'04*. 47–55.

E. Mendes and B. Kitchenham. 2004b. Further Comparison of Cross-company and Within-company Effort Estimation Models for Web Applications. In *Proc. of METRICS'04*. 348–357.

E. Mendes and N. Mosley. 2008. Bayesian Network Models for Web Effort Prediction: A Comparative Study. *IEEE TSE* 34, 6 (2008), 723–737. DOI:http://dx.doi.org/10.1109/TSE.2008.64

T. Menzies, R. Krishna, and D. Pryor. 2017. The SEACRAFT Repository of Empirical Software Engineering Data. (2017). https://zenodo.org/communities/seacraft

T. Menzies, D. Port, Z. Chen, and J. Hihn. 2005. Validation methods for calibratingsoftware effort models. In *Proc. of ICSE'05*. 587–595. DOI:http://dx.doi.org/10.1109/ICSE.2005.1553605

T. Menzies and M. Shepperd. 2012. Special issue on repeatable results in software engineering prediction. *ESE* 17, 1 (2012), 1–17.

L. Minku, F. Sarro, E. Mendes, and F. Ferrucci. 2015. How to Make Best Use of Cross-Company Data for Web Effort Estimation?. In *Proc. of ESEM'15*. 1–10. DOI:http://dx.doi.org/10.1109/ESEM.2015.7321199

N. Mittas and L. Angelis. 2012. A Permutation Test Based on Regression Error Characteristic Curves for Software Cost Estimation Models. *ESE* 17, 1 (2012), 34–61.

N. Mittas and L. Angelis. 2013. Ranking and Clustering Software Cost Estimation Models through a Multiple Comparisons Algorithm. *IEEE TSE* 39, 4 (2013), 537–551.

N. Mittas, I. Mamalikidis, and L. Angelis. 2015. A framework for comparing multiple cost estimation methods using an automated visualization toolkit. *IST* 57, Supplement C (2015), 310 – 328. DOI:http://dx.doi.org/https://doi.org/10.1016/j.infsof.2014.05.010

Y. Miyazaki, M. Terakado, K. Ozaki, and H. Nozaki. 1994. Robust Regression for Developing Software Estimation Models. *JSS* 27, 1 (1994), 3–16. DOI:http://dx.doi.org/10.1016/0164-1212(94)90110-4

I. Myrtveit, M. Shepperd, and E. Stensrud. 2005. Reliability and Validity in Comparative Studies of Software Prediction Models. *IEEE TSE* 31, 5 (2005), 380–39.

J. C. Nash. 2000. The (Dantzig) simplex method for linear programming. *Computingin Science & Engineering* 2, 1 (2000), 29–31.

J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman. 1996. *Applied Linear Statistical Models*. McGraw-Hill Irwin.

G. Neumann, M. Harman, and S. M. Poulding. 2015. Transformed Vargha-Delaney Effect Size. In *Proc. of SSBSE'15*. 318–324.

D. Port and M. Korte. 2008. Comparative Studies of the Model Evaluation Criterions MMRE and Pred in Software Cost Estimation Research. In *Proc. of ESEM'08*. 51–60.

R Development Core Team. 2011. R: A language and environment for statistical computing. (2011). http://www.r-project.org/foundation/

R. L Rardin. 1998. *Optimization in operations research*. Vol. 166. Prentice Hall Upper Saddle River, NJ.

J. D. Rodriguez, A. Perez, and J. A. Lozano. 2010. Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation. *IEEE Trans. Pattern Anal. Mach. Intell* 32, 3 (2010), 569–575.

P. Royston. 1982. An extension of Shapiro and Wilk's W test for normality to large samples. *Applied Statistics* 31, 2 (1982), 115–124.

F. Sarro, S. Di Martino, F. Ferrucci, and C. Gravino. 2012a. A Further Analysis on the Use of Genetic Algorithm to Configure Support Vector Machines for Inter-release Fault Prediction. In *Proc. of SAC'12*. ACM, 1215–1220. DOI:http://dx.doi.org/10.1145/2245276.2231967

F. Sarro, F. Ferrucci, and C. Gravino. 2012b. Single and Multi Objective Genetic Programming for Software Development Effort Estimation. In *Proc. of SAC'12*. ACM, 1221–1226. DOI:http://dx.doi.org/10.1145/2245276.2231968

F. Sarro, F. Ferrucci, M. Harman, A. Manna, and J. Ren. 2017. Adaptive Multi-objective Evolutionary Algorithms for Overtime Planning in Software Projects. *IEEE TSE* (2017). DOI:http://dx.doi.org/10.1109/TSE.2017.2650914

F. Sarro, M. Harman, Y. Jia, and Y. Zhang. 2018. Customer Rating Reactions Can Be Predicted Purely Using App Features. In *Proc. of the 26th IEEE International Requirements Engineering Conference (RE'18)*.

F. Sarro, A. Petrozziello, and M. Harman. 2016. Multi-objective software effort estimation. In *Proc. of ICSE'16*. 619–630. DOI:http://dx.doi.org/10.1145/2884781.2884830

M. Shepperd, M. Cartwright, and G. Kadoda. 2000. On Building Prediction Systems for Software Engineers. *ESE* 5, 3 (2000), 175–182.

M. Shepperd and C. Schofield. 2000. Estimating Software Project Effort using Analogies. *IEEE TSE* 23, 11 (2000), 736–743.

M. Shepperd, C. Schofield, and B. Kitchenham. 1996. Effort Estimation using Analogy. In *Proc. of Int. Conf. on Software Engineering*. IEEE press, 170–178.

M. J. Shepperd and S. G. MacDonell. 2012. Evaluating prediction systems in software project estimation. *IST* 54, 8 (2012), 820–827.

B. Sigweni, M. Shepperd, and T. Turchi. 2016. Realistic Assessment of Software Effort Estimation Models. In *Proc. of EASE'16*. ACM, 41:1–41:6. DOI:http://dx.doi.org/10.1145/2915970.2916005

E. Stensrud, T. Foss, B. Kitchenham, and I. Myrtveit. 2003. A Further Empirical Investigation of the Relationship Between MRE and Project Size. *ESE* 8, 2 (2003), 139–161.

E. Stensrud and I. Myrtveit. 1996. Human Performance Estimatingwith analogy and Regression Models: an Empirical Validation. In *Proc. of METRICS'98*. IEEE press, 205–213.

C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto. 2018. The Impact of Automated Parameter Optimization on Defect Prediction Models. *IEEE TSE* (2018).

P. A. Whigham, Caitlin A. Owen, and S. G. Macdonell. 2015. A Baseline Model for Software Effort Estimation. *ACM TOSEM*. 24, 3 (2015), 20:1–20:11. DOI:http://dx.doi.org/10.1145/2738037

F. H. Yun. 2010. China: Effort Estimation Dataset. (2010). https://doi.org/10.5281/zenodo.268446

H. Zhang, L. Gong, and S. Versteeg. 2013. Predicting Bug-fixing Time: An Empirical Study of Commercial Software Projects. In *Proc. of ICSE'13*. 1042–1051.

Online Appendix to: Linear Programming as a Baseline for Software Effort Estimation

Federica Sarro, University College London
Alessio Petrozziello, University of Portsmouth

A. DESCRIPTION OF THE DATASETS USED IN THE EMPIRICAL STUDY

The Albrecht+Kemerer (AK) dataset consists of 24+15 (=39) industrial software projects coming from the Albrecht [Albrecht and Gaffney 1983] and Kemerer [Kemerer 1987] datasets, respectively. The Albrecht software projects were developed by the IBM DP service organisation and are characterised in terms of thousand Source Lines of Code (i.e., KSLOC) and Function Points count, which is essentially a weighted sum of the numbers of inputs, outputs, files, and inquiries provided to, or generated by, a software. Also, the effort required to design, develop, and test the application is provided for each of the projects in terms of number of work-hours. The Kemerer software projects are characterised by two categorical variables (Language and Hardware), two software size measures based on Source Lines of Code (i.e., KSLOC), Adjusted Function Points (i.e., AdjFP) and Raw Function Points (i.e., RAWFP), and two dependent variables, namely the project's duration and the total effort needed to realise each of the projects and computed based on man/month. Note that the use of the KSLOC and the project's duration is usually discouraged in the construction of effort estimation models because these variables are usually correlated to the effort. Since Albrecht and Kemerer have one independent variables in common which can be used at prediction time (i.e., AdjFP) we select this to build the effort estimation model and used the variable effort as the dependent one.

The China dataset [Yun 2010] includes data of 499 projects developed by different Chinese companies. We used the basic elements used to calculate Function Points (i.e., Input, Output, Inquiry, File, Interface) as independent variables and the variable Effort as the dependent one.

The Desharnais dataset [Desharnais 1989] comprises 81 software projects derived from a Canadian software company. We considered the total effort as a dependent variable, but not the length of the code. We also excluded from our analysis the categorical variables (i.e., Language and YearEnd) and four projects that have missing values, as done in previous work (e.g., [Sarro et al. 2016; Kadoda and Shepperd 2001; Shepperd and Schofield 2000]). Therefore, we used the following independent variables: TeamExp (i.e., the team experience measured in years), ManagerExp (i.e., the manager experience measured in years), Entities (i.e., the number of the entities in the system data model), Transactions (i.e., the number of basic logical transactions in the system), AdjustedFPs (i.e., the Adjusted Function Points).

The Finnish dataset [Shepperd et al. 1996] contains data from 38 industrial software projects developed by nine different Finnish companies. Each project is described by the dependent variable Effort, expressed in person-hours, and five other variables, among which we excluded the PROD variable since it represents the productivity expressed in terms of Effort and size, and only used HW (i.e., the type of hardware), FP (i.e., Function Points), AR and CO as the independent variables to build effort estimation models.

The Kitchenham dataset [Kitchenham et al. 2002] contains data from 145 maintenance and development industrial projects managed by a single outsourcing company, including effort estimates and actuals (dependent variable), and function points count (independent variable). The estimates were made as part of the company's standard project estimating process that involved producing two or more estimates for each project and selecting one estimate to be the basis of client-agreed budgets.

The Maxwell dataset [Maxwell 2002] contains 62 industrial software projects developed for one of the biggest commercial banks in Finland. We employed 17 features: Function Points (SizeFP) and 16 ordinal variables, i.e., number of different development languages used (Nlan), customer participation (T01), development environment adequacy (T02), staff availability (T03), standards used (T04), methods used (T05), tools used (T06), softwares logical complexity (T07), requirements volatility (T08), quality requirements (T09), efficiency requirements (T10), installation requirements (T11), staff analysis skills (T12), staff application knowledge (T13), staff tool skills (T14), and staff team skills (T15). As for the Desharnais dataset, we did not use categorical variables.

The Miyazaki dataset [Miyazaki et al. 1994] is composed by 48 industrial software projects developed by 20 different software companies of the Fujitsu Large Systems Users Group. For this dataset, we considered the following independent variables: SCRN (i.e., the number of different input or output screens), FORM (i.e., the number of different report forms), and FILE (i.e., the number of different record format). The dependent variable is Effort, defined as the number of person-hours needed from system design to system test, including indirect effort such as project management.

The Nasa dataset [Bailey and Basili 1981] consists of 18 software projects developed for the NASA/Goddard Space Flight Center. The projects used in our analysis are described in terms of the two independent variables Methodology (Me) and Experience (Exp), which represent, respectively, the methodologies used during design and development, and the experience of the customer and of the programmers. Effort is the dependent variable and measures the actual effort (expressed in man-months) needed to release each of the projects from the beginning of the design phase through the acceptance testing, therefore it includes the effort for programming, management and support hours. A detailed description of these factors and the collection procedure can be found elsewhere [Bailey and Basili 1981]. Note that we excluded from our analysis those factors that are unknown in the early phases of a project such as the actual number of source lines developed [Bailey and Basili 1981].

The Nasa93Coc dataset [Menzies et al. 2005] consists of 93 projects developed between 1971 and 1987 by different NASA centres. The effort (our dependent variable) was measured in calendar months of 152 hours and includes development and management hours. This dataset includes 15 COCOMO I discrete attributes (i.e., rely, data, cplx, time, stor, virt, turn, acap, aexp, pcap, vexp, lexp, modp, tool, sced) , which are in the range Very Low to Extra High as defined by [Boehm 1981] and software size in thousand Source Lines of Code (i.e., KSLOC), which was estimated directly or with a function point analysis.

The Telecom dataset [Shepperd and Schofield 2000] consists of 18 projects characterised by two independent variables, i.e., the number of changes made as recorded by the configuration management system (Changes) and the number of files changed by a given enhancement project (i.e., Files), and the dependent variable Effort which constitutes the actual effort. According to [Shepperd and Schofield 2000] only the variable Files can be used for predictive purposes since none of the other information were available at the time the prediction was made.