https://doi.org/10.31449/inf.v44i2.2759 Informatica **44** (2020) 231–239 **231**

Data Mining Approach to Effort Modeling on Agile Software Projects

Hrvoje Karna, Sven Gotovac and Linda Vicković
University of Split, Poljička cesta 35, 21000, Split, Croatia
E-mail: hrvoje.karna@gmail.com, sven.gotovac@fesb.hr, linda.vickovic@fesb.hr

Keywords: agile scrum, data mining, effort estimation, *k*-nearest neighbor, software engineering, project management

Received: April 26, 2019

Software production is a complex process. Accurate estimation of the effort required to build the product, regardless of its type and applied methodology, is one of the key problems in the field of software engineering. This study presents the approach to effort estimation on agile software project using local data and data mining techniques, in particular k-nearest neighbor clustering algorithm. The applied process is iterative, meaning that in order to build predictive models, sets of data from previously executed project cycles are used. These models are then utilized to generate estimate for the next development cycle. Used data enrichment process, proved to be useful as results of effort prediction indicate decrease in estimation error compared to the estimates produced solely by the estimators. The proposed approach suggests that similar models can be built by other organizations as well, using the local data at hand and this way optimizing the management of the software product development.

Povzetek: V prispevku je predstavljen pristop strojnega rudarjenja za modeliranje agilnih programskih projektov.

1 Introduction

Accurate estimation of work effort required to build the product is a critical activity in software development industry [1] and it is carried out on most projects [2]. Previously a number of approaches have been proposed to reliably estimate the effort, such as theoretical [3], formal [4], analogy-based estimation [5], just to name a few. Despite all, expert estimation [6] remains the most widely used method of effort estimation.

Regardless of its comparative advantages, such as ease of implementation and the validity of the results it produces [7], expert effort estimation can still be improved [8]. Estimation is particularly challenging in large agile projects [9]. One way to achieve this is to use own, locally built, collections of past project data [10], [11]. The emergence of machine learning algorithms and data mining in general, paired with the availability of tools, has led to progress in application of these methods in practice [12].

This paper presents an approach to effort estimation using data mining techniques, particularly *k*-Nearest Neighbor (KNN) clustering algorithm [13], on local collection of telco project data. The approach uses local data [14], extracted from the tracking system implemented on the project. The process itself is iterative, implemented in a way that at first it uses a collection of data from initial project phase in order to build primary predictive model.

Then in the next project phase – an upgrade, this model is being enriched with the data from the recently completed iteration in order to gradually improve its properties, and thus reduce the estimation error.

This research builds upon our previous work [15] now being applied to a large agile project and using different approach to predict effort. Instead of project clustering applied in [15], in this paper KNN is used to cluster work items and for each new instance it finds the nearest neighbors and calculates the model predicted effort.

The proposed approach itself follows on one hand the iterative nature of agile scrum methodology [16] implemented on the project while at the same time fitting it to the cyclicality of the CRISP-DM process [17]. This proved to be efficient way to improve estimation accuracy and therefore can be suggested as a method by which organizations can improve the process of project management.

The remaining part of this paper is organized as follows: Section 2 presents the current state of the research of the areas being discussed in the paper. Section 3 elaborates the design of the study, applied approach and techniques used to model effort estimation. In Section 4 results are presented together with their implication and potential limitations. The concluding section summarizes the findings and gives directions for future work.

2 Related research

Data mining techniques provide a means to analyze and extract patterns from data and through that process produce previously unknown and potentially useful information [18]. They emerged as an interdisciplinary domain with evolution and merging of databases, statistics and machine learning [19].

It can be viewed as a method for discovering knowledge from large sets of data [20]. Data mining consists of a set of techniques applicable for different purposes [21]. Clustering being one among them is particularly useful in prediction [22] and KNN is one of the most widely used algorithms [23].

Research in the field of software development effort estimation is active since the emergence of this industry [24]. During that period this has resulted in the number of approaches intended to estimate the effort required to build the product [25], each with their own advantages and limitations. Up to now, due to its comparative advantages, expert effort estimation remains the most frequently used technique in practice [26]. Paired with modern data analysis techniques it has potential to significantly improve reliability of the estimates [27].

Mining software engineering data raises the interest of researcher for quite some time [28], it also poses specific challenges [29]. It has been applied to different types of data [30], [31] and uses a number of techniques [32]. The application of these techniques is particularly appropriate in software engineering as it is rich in data [33] while, on the other hand, they can be used to optimize the software development process, software itself and support decision making process [34].

Agile development methods emerged from the need to efficiently handle close interaction with the customer, flexibility in requirements definition and the urge to deliver software on time and within the budget [35]. In contrast to sequential, agile development methods propose incremental approach to building of the software product [36]. These practices can also be used to handle the system and team scale issues [37] what is especially important in today's dynamic business environment.

Agile scrum executes the project in a sequence of iterations called sprints, where each sprint represents a cycle within which development activities occur [38]. During sprint planning, team members determine sprint goal, prioritize and estimate the effort of work items [39].

3 Study design

This empirical study was performed using local data from a complex telco solution development project executed in large international company. Development of the application was based on Java technology and Oracle DB. Data used for the study refers to the tracking system items and descriptive features of the estimators, as these are the entities used to construct the predictive models. The authors implemented these models before [15], [40], so the selection of predictors was based on their relative importance determined in this, our previous [41] and similar studies [2].

The study exclusively used data required to build predictive models for effort estimation and for this it was sufficient that for example, components are identified as Component_1, Component_2, etc. or that estimators are referred to as Estimator_1, Estimator_2, ..., and so on, with matching attributes taking appropriate values.

The average number of estimators per sprint fluctuated around 22, reaching at one point the total of 31. The number of estimation items per Sprint was between 80 and 110, with the total of 1,732 in Phase 1 and 532 in Phase 2. Total actual effort recorded in Phase 1 was 20,814.25 [h] and in Phase 2 sprints 5,344.50 [h].

These numbers indicate that the analyzed project belongs to the class of "large" projects [42]. In the sequence of analyzed sprint data, none of them ended up exactly on the estimated value of effort. Both under and over estimations occurred with relatively same frequency, see Table 1 (sprints 1-19) and Table 3 (sprints 20-24), yet overestimation was more common in early project phase while underestimation was more common in later phase.

Sprint	Effort [h]		Estimation Error		
	Estimated	Actual	Absolute (Relative)	MMRE	Pred(0.25)
1	1,335.00	1,297.00	+38.00 (+2.93%)	0.652	0.660
2	1,224.00	1,302.00	-78.00 (-5.99%)	0.215	0.738
3	1,294.00	1,223.00	+71.00 (+5.81%)	0.310	0.673
4	1,173.00	1,171.00	+2.00 (+0.17%)	0.359	0.774
5	375.00	358.00	+17.00 (+4.75%)	0.522	0.733
6	1,328.00	1,278.00	+50.00 (+3.91%)	0.378	0.767
7	1,314.00	1,289.00	+25.00 (+1.94%)	0.301	0.663
8	1,262.00	1,239.00	+23.00 (+1.86%)	0.323	0.670
9	1,056.00	1,078.00	-22.00 (-2.04%)	0.254	0.803
10	1,432.50	1,424.25	+8.25 (+0.58%)	0.210	0.779
11	1,120.00	1,146.00	-26.00 (-2.27%)	0.071	0.879
12	1,518.00	1,479.00	+39.00 (+2.64%)	0.383	0.780
13	1,255.00	1,304.00	-49.00 (-3.76%)	0.092	0.893
14	1,089.00	1,081.00	+8.00 (+0.74%)	0.063	0.925
15	991.00	975.00	+16.00 (+1.64%)	0.226	0.861
16	1,182.00	1,149.00	+33.00 (+2.87%)	0.180	0.843
17	970.00	979.00	-9.00 (-0.92%)	0.194	0.813
18	884.00	922.00	-38.00 (-4.12%)	0.128	0.848
19	118.00	120.00	-2.00 (-1.67%)	0.026	0.923

Table 1: Efforts and estimation error values per sprint for the training set.

The analyzed data covers Phase 1 (initial version) and Phase 2 (upgrade) of development project. Each phase was implemented in so called sprints i.e. development cycles as defined by the agile scrum methodology. Phase 1 consists of 19, while Phase 2 covers 5 sprints. Each sprint produces a given set of estimation items i.e. data records. The problem that is being solved was weather it is possible to predict the effort of the upcoming Phase 2 sprints by using the knowledge from those completed. Sprints 1 to 19 (S1-S19) were used as initial data base of items for training and test of predictive model, while sprints S20 to S24 served for validation, see Figure 1.

Upon building of the initial model M1 this model was used to predict effort of sprint S20. In each following iteration the model was enriched by the data from the last sprint, thus the data set (S1-S19+S20) was used to build model M2 and predict effort of S21, data set (S1-S19+S20+S21) was used to build model M3 and predict effort of S22, etc. This process passed through five iterations that could also be presented as follows:

1st iteration: (S1-S19) } M1 → S20
2nd iteration: (S1-S19+S20) } M2 → S21
3rd iteration: (S1-S19+S20+S21) } M3 → S22
4th iteration: (S1-S19+S20+S21+S22) } M4 → S23
5th iteration: (S1-S19+S20+S21+S22+S23) } M5 → S24

Expert estimation heavily relies on the intuition where based on the received input information estimator uses his judgment to come up with the solution [43]. This process can be improved by designing models that support the estimation of effort.

The proposed predictive model targets the agile software development environment. It uses data mining approach that is explained next in more details. This is followed by the description of the entities that represent the sources of data and the fields used as predictors of the effort. Finally, the modeling method, determined by the selected tool itself is described.

3.1 Data mining process

Building of the data mining model considered in this study required the definition of research objective. In this case it was optimization of the software development process through the application of machine learning algorithm in order to provide the way to decrease effort estimation error, thus allowing more efficient management of the project.

The data mining process applied in this study uses de-facto industry standard known as CRISP-DM (CRoss-Industry Standard Process for Data Mining). This is iterative process structured around six phases:

- *Business understanding* – identification of the business problem that has to be solved,
- *Data understanding* – obtaining, exploring and verification of the data that will be used,
- *Data preparation* – retirement of the data before it can be used for modeling,
- *Modeling* – selection of appropriate technique, building and assessment of the model,
- *Evaluation* – evaluation of results and review of the process,
- *Deployment* – use of the model in order to improve the business.

Understanding of the business and data was established prior and during initial prediction iteration: (S1-S19) } M1 → S20. For each next iteration data preparation followed by modeling and evaluation phase was executed. The presented model has academic purpose i.e. evaluation of proposed approach, so currently there is no deployment in real environment. Once the model proves effectiveness, it is possible to recommend its application in practice.

3.2 Entities and data

The study uses following entities and related fields as data sources:

● *Item*: these are the records by which the work is represented and stored in the tracking system implemented on the analyzed project i.e. tickets. Variables used to represent work item entity are: Assignment (representing type of item association to the estimator, taking the form of either "own" or "assigned"), Component (identifying the component

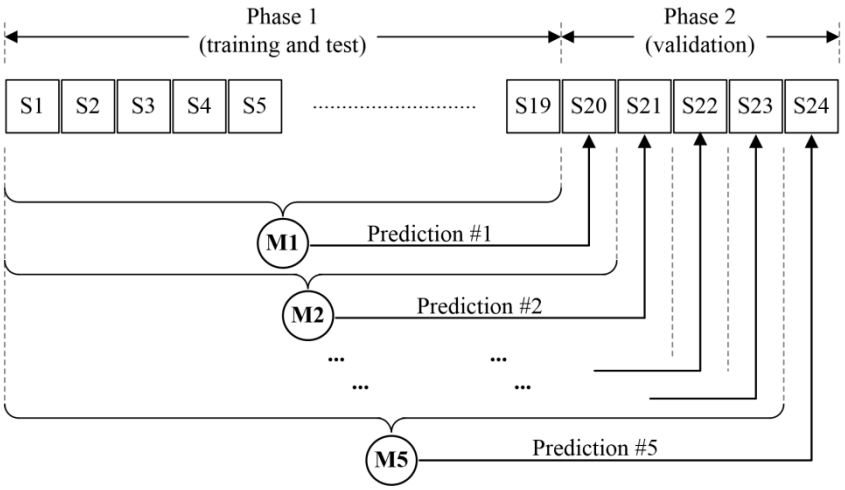

Figure 1: Model building and prediction process used in the study.

within the system that is related to, identified as Component_1, Component_2, …), Area (refers to the area of work with possible values: PM, QM, CM, System, …, Other), Activity (refers to the type of activity with possible values: Management, Quality, Design, Implementation, Test, …, Installation, Documentation, etc.), Type (identifies type of the item according to the applied scrum methodology, being either user story, task, defect, or other) and Priority (or urgency, it indicates the order in which item should be taken into execution in relation to the other items, describe as Prio_1, Prio_2, …, where Prio_1 refers to the highest priority). As it is evident, these are descriptive attributes related to the item at the moment of its creation. Additional fields associated with the item entity used to record the efforts are: Estimated Effort, Remaining Effort and Actual Effort. These were populated at the moment of item creation and later updated as the work progresses until its completion.

- *Estimator*: the estimator is basically the employee engaged on the project, sometimes referred to as a project team member. In the model the estimator is represented with set of variables describing his: Role (representing his primary occupation on the project, with potential values: Project Manager, Solution Architect, Software Engineer, Configuration Manager, etc.), Seniority Level (representing the level of seniority, being either Junior, Mid-Level or Senior), Total Experience (representing the total number of years of work experience), Company Experience (representing number of years of experience within the current company), Number of Projects (representing number of projects employee participated in while working for the current company) and Estimation Competence (representing the level of estimation competence, being either Beginner, Intermediate or Advanced).

The list of fields used as predictors and target, together with associated measurement type is presented in Table 2.

Entity	Field		
	Name	Measurement	Role
ITEM	Assignment	Flag	Predictor
	Component	Nominal	
	Area	Nominal	
	Activity	Nominal	
	Type	Nominal	
	Priority	Ordinal	
ESTIMATOR	Role	Nominal	
	Seniority Level	Ordinal	
	Total Experience	Continuous	
	Company Experience	Continuous	
	Number of Projects	Continuous	
	Estimation Competence	Ordinal	
	Actual Effort	Continuous	Target

Table 2: Predictors and target of proposed model.

3.3 *k*-Nearest Neighbor algorithm

The model uses *k*-Nearest Neighbor (KNN) algorithm. The nearest neighbor (NN) rule assigns to unclassified incoming observation the class of the nearest sample in the set, the simplest form of KNN when k = 1 [44]. KNN is based on measuring the distance between data to decide the final classification output based on their similarity [45].

KNN is an extension of NN and due to its advantages has been used for solving classification problems in numerous domains, the algorithm procedure can be presented as follows [46]:

$$T = \{(x_i, y_i)\};\ i = 1, \dots, N$$

Let T denote the training set, where $x_i \in \Re^m$ is a training vector in the m-multidimensional feature space, and $y_i$is the corresponding class label. Given a query x', its unknown class y' is assigned in two steps.

First, a set of k similarly labelled target neighbors for the query x' is identified. Denote the set

$$T' = \{(x_i^{NN}, y_i^{NN})\};\ i = 1, \dots, k$$

arranged in an increasing order in terms of Euclidian distance $d(x', x_i^{NN})$ between x' and x_i^{NN}

$$d(x', x_i^{NN}) = \sqrt{(x' - x_i^{NN})^T (x' - x_i^{NN})}$$

Secondly, the class label of the query is predicted by the majority voting of its nearest neighbors:

$$y' = \arg \max_y \sum_{(x_i^{NN},\ y_i^{NN}) \in T'} \delta(y = y_i^{NN})$$

where y is a class label, y_i^{NN} is the class label for the *i*-th nearest neighbor among its *k* nearest neighbors. $\delta(y = y_i^{NN})$, the Dirac delta function, takes a value of one if $y = y_i^{NN}$ and zero otherwise.

The quality of *k*-Nearest Neighbor algorithm depends on the choice of *k* and the distance measure parameter [47]:

- *k*: the selection of k is dependent on the selected data set. There are different recommendations but, instead of having the same number of nearest neighbors, it is good to find the best k automatically [48], the approach used in our study in order to choose the best number of neighbors within the range.
- *Distance*: the distance or dissimilarity measure, between two existing cases x_i and y_i can generally be expressed by Euclidean distance, as presented above. Computed distance is basically the magnitude of the vector obtained by subtracting the training data point from the point to be classified.

Thus, in the space defined by the input fields i.e. predictors, cases positioned near each other are referred to as neighbors. Those dissimilar are more distant from each other. For a new case i.e. target that enters the model, the procedure calculates the predicted value of a

continuous measurement type as a mean of k nearest neighbor values [49].

KNN was previously used for estimating effort and provided better results in comparison to other techniques [50], [51]. However, what distinguishes this study is that it is performed on a local set of data on a project driven by agile methodology while applying iterative effort modeling per sprint. This makes it unique according to our knowledge.

3.4 Modeling and evaluation

From the input set 12 variables were used as predictors and single variable (Actual Effort) as a target. The experiment was conducted using IBM SPSS Modeler tool 14.2 [52]. In each iteration for analyzed data sets a stream representing data flow was formed to perform experiment. The modelling element implements the *k*-Nearest Neighbor algorithm, with k set in range of minimum of 3 and maximum of 5, allowing procedure to choose the best number of neighbors, in order to compute the value of the target variable.

Effort modeling for each validation Phase 2 sprint is performed in the following steps:

1. Predictive model is built using data from previously executed i.e. finished sprints,
2. Existing effort values were removed from input data of the sprint that is estimated in iteration,
3. Sprint data is feed into the prediction stream,
4. Predictive modeling is performed and model estimates are generated,
5. Effort estimates are exported from the stream for subsequent evaluation.

After generation of the model predictions for each Phase 2 iteration, as a part of the evaluation procedure, the comparison of the results of estimations produced by models vs. estimators in relation to the actual values of the total reported effort per sprint was performed. In addition to that criterion, the standard measures of estimation error, MMRE and Pred at level x, are used [53]. They are explained next.

Estimation error is the difference between the estimated effort (EST) and the actual value (ACT):

$$EE = EST - ACT$$

Magnitude of relative error (MRE) is the absolute value of estimation error relative to the actual:

$$MRE = \frac{|EST - ACT|}{ACT}$$

it is the basic metric used to calculate Mean Magnitude of Relative Error (MMRE):

$$MMRE = mean\ (MRE) = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(EST_i - ACT_i)}{ACT_i}\right|$$

The Pred(x) is a criterion that defines the predictions having a relative error of less than or equal to level x, the set threshold, defined as:

$$Pred(x) = \frac{100}{N}\sum_{i}^{N}\begin{cases}1 & if \quad MRE_{\,i} \le x/100\\ 0 & otherwise\end{cases}$$

The x is typically set to 25 so that it reveals the portion of the estimates that are within the tolerance of 25% from the actuals.

Using these metrics it was possible to conduct a reliable evaluation of the predictive model efficiency.

4 Results and discussion

In this section results of the modeling process are presented and commented. Additionally, the implications and limitations related to the data, model and study are discussed.

4.1 Study results

The results of the effort predictions generated by the models, together with the values of effort estimated by the expert estimators and actuals, for each of the validation sprints are listed in Table 3 and illustrated in Figure 3. To meet the standards used by both industry practitioners and scientific community during evaluation we use a comparison of the estimated and the actual efforts [26] as well as MMRE and Pred(0.25). From the data it can be seen that validation sprints vary in volume, ranging from some 500 [h] up to more than 1,500 [h] of actual effort, making validation set representative.

By reviewing the results of the total values produced by the estimators and predictive models, compared to the actuals of each sprint we can conclude that models provided better estimates in four (S21, S22, S23 and S24) out of five iterations. Given the volume of the iteration S20 the difference in gains that the experts made in relation to the model predictions was practically negligible. Within the last four sprints, in three cases the model's prediction was significantly better than the estimates the experts gave.

Regarding the direction of the estimation error, from the validation set, estimators underestimated effort in three out of five sprints and the same was the outcome of the predictions made by the models. It is interesting that errors from both experts and the predictive models had the same tendency i.e. the models do not show the tendency to either under or overestimate but that results depend exclusively of the properties of the provide data set. It seems that both classify in the similar way, that is, the model in certain way mimics the reasoning process but performs better.

Using this evaluation approach, typical for industry practitioners, and comparing the average estimation error produced by the models it is evident that it was smaller in magnitude as presented in Table 3 and that it had a positive tendency i.e. as the modeling progressed the trend of error correction was better, see Figure 4. This can be attributed to the data mining learning process in which as the quantity of data used to build predictive models was increased from iteration to iteration. As it is evident, this had a positive impact on the accuracy of the

Sprint	Effort [h]			Error[%]		
	Estimated	Actual	Model	Estimators	Model	Correction
20	814.00	866.00	809.00	-6.00%	-6.58%	-0.58%
21	1,688.00	1,549.00	1,647.80	8.97%	6.38%	2.60%
22	1,305.00	1,109.50	1,258.40	17.62%	13.42%	4.20%
23	1,133.00	1,281.00	1,215.60	-11.55%	-5.11%	6.45%
24	497.00	539.00	500.60	-7.79%	-7.12%	0.67%

Table 3: Efforts, Estimation error and correction per sprint for the validation set.

predictions as the tendency of their reliability increased. Therefore, we can assume that a similar trend would have continued if this phase of development consisted of more sprints.

The use of both MMRE and Pred was an option in order to provide more accurate study results as these metrics show different tendencies [54]. MMRE and Pred(0.25) are both measures of relative estimation error in a collection of instances, but quite different. Greater values of MMRE indicate greater magnitudes of error, while higher Pred(0.25) score indicates better estimation efficacy i.e. more predictions within a set tolerance (in this case of 25%) from the actuals.

Comparison of the MMRE and Pred(0.25) values, which are *de facto* standard measures used by the scientific community in the field, generated by the estimators and models for the validation set is provided in Table 5. These results clearly indicate that the model generated estimates produced an overall smaller estimation error. Here again we notice a practically equal score in S20 and improvement in S21, S22, S23 and S24, see Figure 5.

Another observation that can be made is that predictive model, based on *k*-Nearest Neighbor (KNN) algorithm, generally outperformed the expert estimators in their ability to estimate. This indicates that selection of learners used in the model was properly carried out and that the overall modeling process itself was effective.

The results of the performed evaluation confirm the applicability of the proposed approach and suggest that similar models could be built using data mining techniques and local data at hand, that way optimizing the estimation process and management of the agile software projects.

4.2 Implications

Used for the purpose of software development effort estimation data mining methods do not only solve that problem but provide a way for better understanding of the context in which estimation occurs and factors that affect it. This way an additional insight to the problem was achieved.

The study once again confirmed the possibility of application of machine learning algorithms to solve the identified problem, this time on a somewhat different type of project. It encourages the enhanced effort estimation process through the synergy of expert estimation and estimation supported by the use of modern methods of prediction.

4.3 Limitations

Potential limitation of this study is the fact that it was performed using the data from a single large agile project. In regards to that in future it would be desirable to conduct similar experiments using the data sets from other projects and environments.

In order to produce more general models future research could as well include projects driven by other

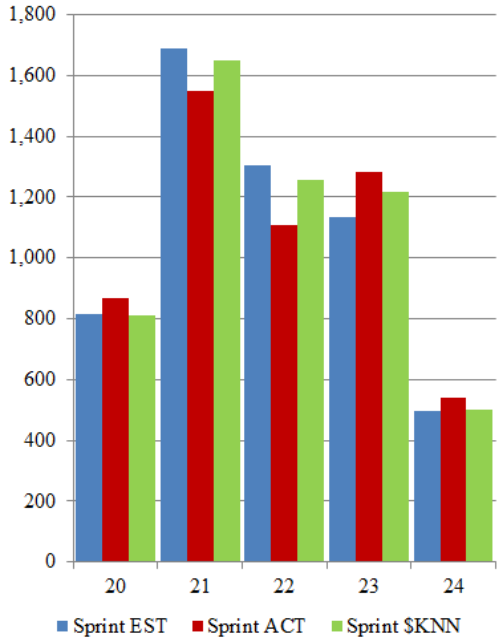

Figure 3: Values of estimated, actual and predicted effort for the validation set.

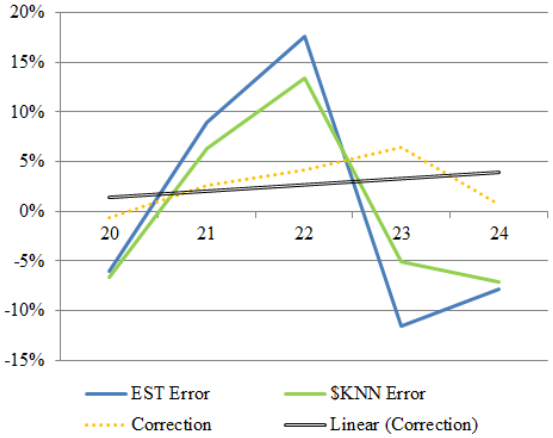

Figure 4: Estimators error, model error and correction in % for validation set with correction trend.

Sprint	Error and Correction					
	Estimators		Model		Correction	
	MMRE	Pred(0.25)	MMRE	Pred(0.25)	MMRE	Pred(0.25)
20	0.086	0.890	0.093	0.878	↓	↓
21	0.501	0.718	0.397	0.835	↑	↑
22	0.700	0.537	0.546	0.734	↑	↑
23	0.183	0.823	0.146	0.872	↑	↑
24	0.106	0.857	0.087	0.915	↑	↑

Table 5: MMRE and Pred(0.25) error and correction indicator per sprint for the validation set.

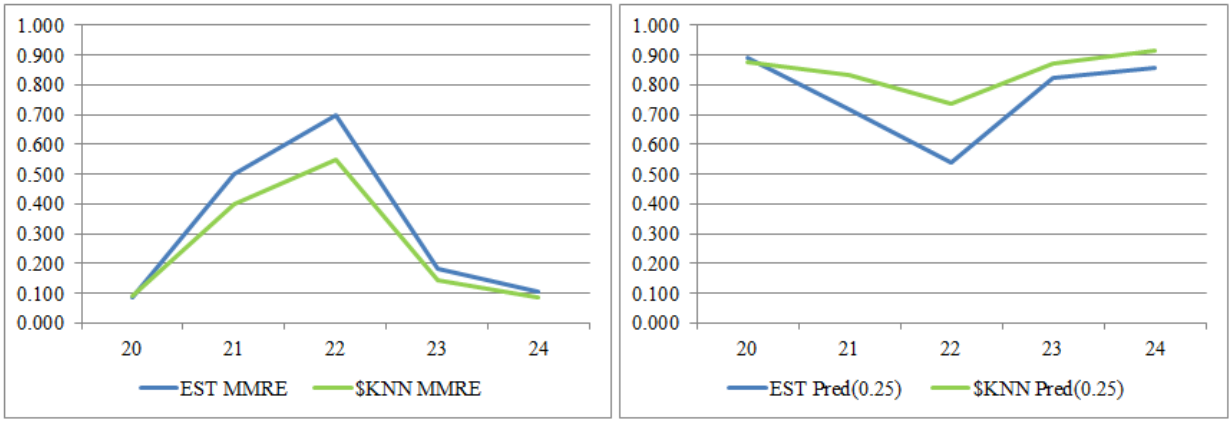

Figure 5: MMRE and Pred(0.25) values generated by estimators and model per sprint for the validation set.

methodologies and those implemented using other technologies. Certainly, relatively straightforward approach used to construct the models, described here, encourages its replication.

5 Conclusions and future work

The paper presented the approach to the effort estimation on agile software project using data mining techniques on the local set of telco project data. It positions the research within the field of software engineering in addition to affirming the actuality of the topic being presented.

Recent research suggests the intensive application of proposed methods to model the effort estimation though, up to our best knowledge, there has been no similar experiment conducted using data set constructed from the mentioned sources and within such environment. This is, among other, the contribution of this work.

The approach proved its validity, providing corrections of the estimated effort generated by the models in most cases in comparison to the experts, and thus can be suggested for use in this or similar forms.

Future work is aimed towards extending the presented model by including data from other entities within the studied environment. Additionally, if possible it would be valuable to include data from other projects of different size and technological basis.

All stated can contribute to the reliability and performance of the predictive models being built and in case of their application in practice support the development process through more optimized project management.

References

[1] M. M. Kirmani. Software Effort Estimation in Early Stages of Software Development A Review, International Journal of Advanced Research in Computer Science (IJARCS), 8(5):1155-1159, 2017. https://doi.org/10.26483/ijarcs.v8i5.3662

[2] T. Haapio and T. Menzies. Exploring the effort of general software project activities with data mining, International Journal of Software Engineering and Knowledge Engineering, 21(5):725-753, 2011. https://doi.org/10.1142/s0218194011005438

[3] L. H. Putnam. A general empirical solution to the macro software sizing and estimating problem, IEEE Transactions on Software Engineering, 4(4):345-361, 1978. https://doi.org/10.1109/tse.1978.231521

[4] A. J. Albrecht and J. E. Gaffney. Software function, source lines of code, and development effort prediction: a software science validation, IEEE Transactions on Software Engineering, 9(6):639-648, 1983. https://doi.org/10.1109/tse.1983.235271

[5] M. Shepperd et al. Effort estimation using analogy, in Proc. of the 18th International Conference on Software Engineering (ICSE), Berlin, Germany, 1996. https://doi.org/10.1109/icse.1996.493413

[6] M. Jørgensen. A review of studies on expert estimation of software development effort, Journal of Systems and Software, 70(1-2):37–60, 2004. https://doi.org/10.1016/s0164-1212(02)00156-5

[7] M. Jørgensen. Practical Guidelines for Expert-Judgment-Based Software Effort Estimation, IEEE Software, 22(3):57–63, 2005. https://doi.org/10.1109/ms.2005.73

[8] B. Tanveer et al. Effort estimation in agile software development: Case study and improvement framework, Journal of Software: Evolution and Practice, Special Issue, 29(11):e1862, 2017. https://doi.org/10.1002/smr.1862

[9] M. Usman et al. Effort Estimation in Large-Scale Software Development: An Industrial Case Study, Information and Software Technology, 99:21-40, 2018. https://doi.org/10.1016/j.infsof.2018.02.009

[10] E. Mendes. Improving Software Effort Estimation Using an Expert-Centered Approach, in Proc. of the 4th international conference on Human-Centered Software Engineering (HCSE '12), Springer-Verlag, Berlin, Heidelberg, 18-33, 2012. https://doi.org/10.1007/978-3-642-34347-6_2

[11] M. Jørgensen. What We Do and Don't Know about Software Development Effort Estimation, IEEE Software, 31(2):37-40, 2014. https://doi.org/10.1109/ms.2014.49

[12] M. Kim et al. Data scientists in software teams: state of the art and challenges, IEEE Transactions on Software Engineering, 44(11):1024-1038, 2017. https://doi.org/10.1109/tse.2017.2754374

[13] M. Bicego and Marco Loog. Weighted K-Nearest Neighbor Revisited, in Proc. of the 23rd International Conference on Pattern Recognition (ICPR '16), Cancún, México, 2016. https://doi.org/10.1109/icpr.2016.7899872

[14] L. Radlinski and W. Hoffmann. On Predicting Software Development Effort Using Machine Learning Techniques and Local Data, International Journal of Software Engineering and Computing (IJSEA), 2(2):123-136, 2010.

[15] H. Karna et al. Application of Data Mining Methods for Effort Estimation of Software Projects, Software: Practice and Experience, 49(2):171-191, 2018. https://doi.org/10.1002/spe.2651

[16] J. Sutherland and K. Schwaber. The scrum guide, Available: https://www.scrumguides.org/

[17] J. Ponce. Data Mining and Knowledge Discovery in Real Life Applications (2nd ed.), Springer, New York, 2009. https://doi.org/10.5772/62143.

[18] Maninderjit Kaur, Sushil Kumar Garg. Survey on Clustering Techniques in Data Mining for Software Engineering, International Journal of Advanced and Innovative Research, 3(4), 2014.

[19] U. Fayyad et al. From Data Mining to Knowledge Discovery in Databases, AI Magazine, 17(3):37-54, 1996. https://doi.org/10.1609/aimag.v17i3.1230

[20] T. Menzies. Practical machine learning for software engineering and knowledge engineering, Handbook of Software Engineering and Knowledge Engineering, 1:837-862, 2001. https://doi.org/10.1142/9789812389718_0035

[21] M. Halkidi et al. Data Mining in Software Engineering, Intelligent Data Analysis Journal (IDA '11), 15(3):413-441, 2011. https://doi.org/10.3233/ida-2010-0475

[22] C. Morbitzer et al. Application of Data Mining Techniques for Building Simulation Performance Prediction Analysis, in Proc. of the 8th International IBPSA Conference, Eindhoven, Netherlands, 2003.

[23] P. Li et al. The Distance-Weighted K-nearest Centroid Neighbor Classification, Journal of Information Hiding and Multimedia Signal Processing, 8(3):611-622, 2017.

[24] B. W. Boehm. Software Engineering Economics, Englewood Cliffs, Prentice Hall, NJ, USA, 1981.

[25] A. K. Bardsiri and S. M. Hashemi. Software Effort Estimation: A Survey of Well-known Approaches, International Journal of Computer Science Engineering (IJCSE), 3(1), 2014.

[26] M. Usman et al. An Effort Estimation Taxonomy for Agile Software Development, International Journal of Software Engineering and Knowledge Engineering (IJSEKE), 27(4):641–674, 2017. https://doi.org/10.1142/s0218194017500243

[27] K. Dejaeger et al. Data Mining Techniques for Software Effort Estimation: A Comparative Study, IEEE Transactions on Software Engineering, 38(2):375-397, 2012. https://doi.org/10.1109/tse.2011.55

[28] A. E. Hassan and Tao Xie. Software intelligence: the future of mining software engineering data, in Proc. of the Workshop on Future of Software Engineering Research (FoSER '10), Santa Fe, NM, USA, 2010. https://doi.org/10.1145/1882362.1882397

[29] T. Xie et al. Data Mining for Software Engineering, IEEE Computer, 42(8):55-62, 2009. https://doi.org/10.1109/mc.2009.256

[30] G. Robles et al. Estimating Development Effort in Free/Open Source Software Projects by Mining Software Repositories, in Proc. of the 11th Working Conference on Mining Software Repositories (MSR 2014), ACM Press New York, NY, USA, 222-231, 2014. https://doi.org/10.1145/2597073.2597107

[31] K. Molokken and M. Jørgensen. Expert Estimation of Web-Development Projects: Are Software Professionals in Technical Roles More Optimistic Than Those in Non-Technical Roles?, Empirical Software Engineering, 10(1):7-29, 2005. https://doi.org/10.1023/b:emse.0000048321.46871.2e

[32] P. K. Suri and P Ranjan. Comparative analysis of software effort estimation techniques, International Journal of Computer Applications (IJCA), 48(21):12-19, 2012. https://doi.org/10.5120/7479-0540

[33] Q. Taylor et al. Applications of data mining in software engineering, International Journal of Data Analysis Techniques and Strategies, 2(3):243-257, 2010. https://doi.org/10.1504/ijdats.2010.034058

[34] L. L. Minku et al. Data mining for software engineering and humans in the loop, Progress in

Artificial Intelligence (PRAI), 5(4):307-314, 2016. https://doi.org/10.1007/s13748-016-0092-2

[35] R. Marques et al. Assessing Agile Software Development Processes with Process Mining: A Case Study, in Proc. of the 20th IEEE Conference on Business Informatics (CBI), 109-118, Vienna, Austria, 2018. https://doi.org/10.1109/cbi.2018.00021

[36] P. Abrahamsson et al. Agile software development methods: Review and analysis, 2002. Available: www.vtt.fi/inf/pdf/publications/2002/P478.pdf

[37] T. Dingsøyr et al. Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation, Empirical Software Engineering, 23(1):490–520, 2018. https://doi.org/10.1007/s10664-017-9524-2

[38] M. Choetkiertikul et al. A deep learning model for estimating story points, IEEE Transactions on Software Engineering, 2018. https://doi.org/10.1109/tse.2018.2792473

[39] K. S. Rubin. Essential Scrum: a practical guide to the most popular agile process (1st Edition), Addison-Wesley, Upper Saddle River, NJ, USA, 2013.

[40] H. Karna and S. Gotovac. Estimators Characteristics and Effort Estimation of Software Projects, in Proc. of the 9th International Joint Conference on Software Technologies (ICSOFT-EA 2014), Vienna, Austria, 2014. https://doi.org/10.5220/0005002600260035

[41] H. Karna and S. Gotovac. Modeling Expert Effort Estimation of Software Projects, in Proc. of 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2014), Split, Croatia, 2014. https://doi.org/10.1109/softcom.2014.7039106

[42] J. Aguilar et al. The Size of Software Projects Developed by Mexican Companies, in Proc. of the International Conference on Software Engineering Research and Practice (SERP'14), 2014. Available: https://arxiv.org/abs/1408.1068

[43] M. Jørgensen et al. Human judgement in effort estimation of software projects, Presented at Beg, Borrow, or Steal Workshop, in Proc. of the International Conference on Software Engineering, Limerick, Ireland, 2000.

[44] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification, IEEE Transactions on Information Theory, 13(1):21–27, 1967. https://doi.org/10.1109/tit.1967.1053964

[45] L.-Y. Hu et al. The distance function effect on *k*-nearest neighbor classification for medical datasets, SpringerPlus, 5(1), 2016. https://doi.org/10.1186/s40064-016-2941-7

[46] J. Gou et al. A New Distance-weighted *k*-nearest Neighbor Classifier, Journal of Information & Computational Science, 9(6):1429–1436, 2012.

[47] J. Huang et al. Cross-validation based K nearest neighbor imputation for software quality datasets: An empirical study, The Journal of Systems and Software, 132:226–252, 2017. https://doi.org/10.1016/j.jss.2017.07.012

[48] E. Kocaguneli et al. Exploiting the essential assumptions of analogy-based effort estimation, IEEE Transactions on Software Engineering, 38(2):425–439, 2012. https://doi.org/10.1109/tse.2011.27

[49] IBM SPSS Modeler: KNN Node, Available: https://www.ibm.com/support/knowledgecenter/en/SS3RA7_15.0.0/com.ibm.spss.modeler.help/knn_node_general.htm

[50] R. Olu-Ajayi. An Investigation into the Suitability of *k*-Nearest Neighbor (k-NN) for Software Effort Estimation, International Journal of Advanced Computer Science and Applications (IJACSA), 8(6), 2017. https://doi.org/10.14569/ijacsa.2017.080628

[51] P. Le and V. Nguyen. A *k*-Nearest Neighbors approach for COCOMO calibration, in Proc. of the 4th NAFOSTED Conference on Information and Computer Science, Hanoi, Vietnam, 219-224, 2017. https://doi.org/10.1109/nafosted.2017.8108067

[52] IBM SPSS Modeler 14.2, Available: http://www-01.ibm.com/support/docview.wss?uid=swg27022140

[53] B. Kitchenham et al. Assessing prediction systems, University of Otago Information Science Discussion Paper No. 99/14, 1999. Available: http://hdl.handle.net/10523/1015

[54] C. Tofallis. A better measure of relative prediction accuracy for model selection and model estimation, Journal of the Operational Research Society, 66(3):1352–1362, 2015. https://doi.org/10.1057/jors.2014.103