FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 43 (2018) No. 4
ISSN 0867-6356
DOI: 10.1515/fcds-2018-0017 e-ISSN 2300-3405

Predicting Aggregated User Satisfaction in Software Projects

Łukasz Radliński*

Abstract. User satisfaction is an important feature of software quality. However, it was rarely studied in software engineering literature. By enhancing earlier research this paper focuses on predicting user satisfaction with machine learning techniques using software development data from an extended ISBSG dataset. This study involved building, evaluating and comparing a total of 15,600 prediction schemes. Each scheme consists of a different combination of its components: manual feature preselection, handling missing values, outlier elimination, value normalization, automated feature selection, and a classifier. The research procedure involved a 10-fold cross-validation and separate testing, both repeated 10 times, to train and to evaluate each prediction scheme. Achieved level of accuracy for best performing schemes expressed by Matthews correlation coefficient was about 0.5 in the cross-validation and about 0.5–0.6 in the testing stage. The study identified the most accurate settings for components of prediction schemes.

Keywords: user satisfaction, prediction scheme, software projects, machine learning, ISBSG.

1. Introduction

User satisfaction is one of the features of software quality. The ISO/IEC 25010:2011 standard, that is widely used in software quality studies, e.g. [9], [14], [25], and [39], defines two quality models: (1) a quality in use model composed of five characteristics related to the outcome of interaction when a product is used in a particular context of use, and (2) a product quality model composed of eight characteristics related to static properties of software and dynamic properties of the computer system. In the first of these models satisfaction is one of top-level characteristics and is defined as a ”degree to which user needs are satisfied when a product or system is used in a specified context of use” [15]. Usually related studies focus on investigating which factors influence user satisfaction. Jones states that user satisfaction is measurable but not predictable [17 p. 456]. The lack of extensive literature on predicting user satisfaction partially confirms this claim.

* Faculty of Computer Science and Information Technology, West Pomeranian University of Technology in Szczecin, ul. Żołnierska 49, 71-210 Szczecin, Poland, lukasz.radlinski@zut.edu.pl.

However, some existing work, discussed later in Section 3, shows that there may be some potential in dealing with the challenge of predicting software quality. Thus, this study focuses on the following two goals:

1. To investigate what accuracy can be achieved in predicting aggregated user satisfaction;
2. To determine which settings for schemes' components deliver the most accurate predictions.

To achieve these goals this study used the extended edition of ISBSG R11 dataset of software projects [13] that contains data on user satisfaction, expressed by eight distinct variables. In this study, a total of 15,600 prediction schemes were built, trained, and evaluated for predictive accuracy. Each scheme consists of different components for manual feature preselection, handling missing values, outlier elimination, value normalization, automated feature selection, and a classifier.

The paper contributes to the applied science and practice by investigating a large amount of modelling/prediction algorithms and techniques in the context of their performance to predict user satisfaction. Presented results demonstrate how these algorithms and techniques perform in comparison with others of the same type and focus.

The rest of this paper is organized as follows: Section 2 presents research method followed and data for the experimental study. Section 3 discusses related work. Section 4 considers limitations and threats to validity. Section 5 presents obtained results of comparing the schemes and their components. Section 6 formulates conclusions and plans for future work.

2. Data and method

This study used the extended version of the ISBSG R11 dataset [13] on past software development projects. The whole dataset contains data on 5024 software projects that were developed worldwide. These projects vary in terms of the type, size, duration, development activities involved, environmental factors, objectives, and documents and techniques used. In the extended version of the dataset, used in this study, the projects are described by 205 attributes. Table 1 describes the basic statistics for the main numeric attributes. Table 2 describes the most frequent states (values) for key environmental attributes. The dataset does not contain attributes describing the country or geographical area where the project was developed.

Table 1. Selected basic statistics for numeric attributes

	Original extended dataset		After preprocessing	
Attribute	Range	Median	Range	Median
Project implementation year	1989–2009	2002	1995–2008	2000
Functional Size (function points)	3–19050	194	22–1670	143
Summary Work Effort (hours)	0–645694	1746	82–18054	2381
Total defects delivered (count)	0–2554	1	0–2554	6

Table 2. Most frequent states for non-numeric attributes

	Original extended dataset		After preprocessing	
Attribute	Value	%	Value	%
Development type	New development Enhancement Re-development	39 59 2	New development Enhancement Re-development	51 46 3
Application type	Accounting Financial transaction process Transaction/production Management inform. system	19 19 8 7	Network management Office information system Management inform. system Process control	12 11 10 9
Business area type	Communications Telecommunications Insurance Banking	19 19 10 10	Manufacturing Engineering Marketing Logistics	24 15 13 7
Organization type	Communications Insurance Services Banking	14 14 11 10	Manufacturing Communications Computers & Software Banking	14 13 11 7

According to the author's knowledge this is the only publicly available dataset of software projects of sensible volume for prediction purposes that contains data on user satisfaction – thus no comparison of predictions with another dataset was performed.

The research process, illustrated in Figure 1, involved the following stages and their steps:

Basic preprocessing and data selection: selecting only cases with 'A' or 'B' for *data quality rating*, partitioning into the CV (65 cases) and test subsets (24 cases); cleaning and adjusting/correcting the values; removing variables with few counts for their states or with many missing values; creating dummy variables from multiple response nominal variables; removing cases with missing user satisfaction values. After performing these tasks 89 projects were kept in the dataset. The significant data reduction was caused mainly by the fact that only a small part of projects were supported with attributes describing user satisfaction that are essential in this study.

The target variable for prediction was calculated by aggregating the following eight individual variables:

- User satisfaction with the ability of system to meet stated objectives;
- User satisfaction with the ability of system to meet business requirements;
- User satisfaction with the quality of the functionality provided;
- User satisfaction with the quality of the documentation provided;
- User satisfaction with the ease of use;
- User satisfaction with the training given;
- User satisfaction with the speed of defining solution;
- User satisfaction with the speed of providing solution.

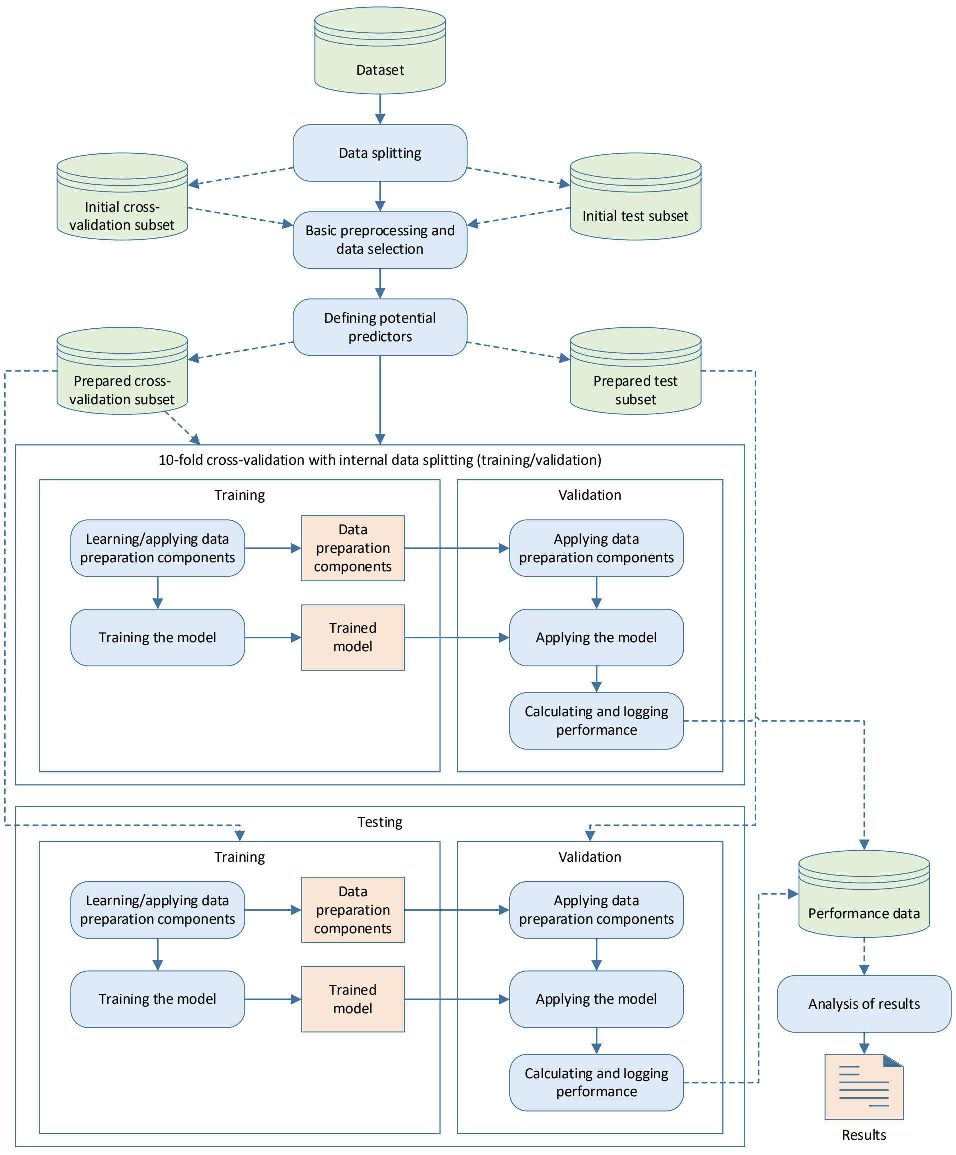
Dataset
Data splitting
Initial cross-validation subset
Initial test subset
Basic preprocessing and data selection
Defining potential predictors
Prepared cross-validation subset
Prepared test subset
10-fold cross-validation with internal data splitting (training/validation)
Training
Learning/applying data preparation components
Data preparation components
Training the model
Trained model
Validation
Applying data preparation components
Applying the model
Calculating and logging performance
Testing
Training
Learning/applying data preparation components
Data preparation components
Training the model
Trained model
Validation
Applying data preparation components
Applying the model
Calculating and logging performance
Performance data
Analysis of results
Results

Figure 1. Schematic of the research process

These variables were originally defined on the following 4-point ranked scale:

‘1’ – user needs met to a limited extent or not at all;
‘2’ – user needs largely met;
‘3’ – user needs fully met;
‘4’ – user expectations exceeded.

Because the focus of this study was the prediction of the overall, i.e. aggregated, user satisfaction it was necessary to combine them into a single target variable. This was achieved by calculating the mean of their numerical values for each case (project). Then, this calculated value was dichotomized – set as 'true' if $Mean(satisfaction) \geq 2.5$ and to 'false' if $Mean(satisfaction) < 2.5$ This dichotomization was motivated by the fact that there were very few projects with mean satisfaction of '1' or '4' – thus, learning patterns and predicting user satisfaction from very few projects would very likely fail.

Defining potential predictors. This step involved creating new potential predictors calculated from regular variables such as rates, proportions or, for variables non-normally distributed, transformed variables with log and Box-Cox transformations. After completing this and the previous step, the dataset contained projects described by 65 potential predictors (columns): 32 numeric, 17 logical or nominal with two states, 12 multiple-response nominal converted to logical dummy variables, and four other nominal with more than two states.

Preparation of prediction environment. The goal of this step was to create an environment capable of repeatable execution of the following main actions for different schemes: loading and preparing data folds, setting up 10-fold cross-validation subprocess using the CV subset, setting up a test subprocess that used a classifier trained on the CV subset to be evaluated on the test subset, setting up logging prediction performance for both CV and test subprocesses, aggregating all main subprocesses into a process capable of multiple execution – i.e. multiple passes each time with different splits of folds in the CV subset. A total of 15,600 prediction schemes were prepared as combinations of different settings for its components: two settings for manual feature preselection performed by a human expert, three missing value imputation algorithms, five outlier detection and removal algorithms, two for value normalization, 13 for automated feature selection (including two algorithms and six settings for number of features), and 20 classifiers. The complete list of these settings are presented in Tables 3–8 Further information on these algorithms, their implementation and settings is available in [6], [32] and partially in [30].

Table 3. Settings for manual feature preselection

Value	Description
no	No manual feature preselection was performed – All variables, that satisfied data quality thresholds were used.
yes	A manual feature preselection was performed – An author investigated each variable's description, value distribution, clearness of value definition and potential usefulness as a predictor variable. A variable was removed from further use if it was classified as non-useful according to author's expertise after judging the above criteria. This was performed by investigating only the CV subset to ensure that the test data are not 'seen' before evaluating predictions.

Table 4. Settings for missing value imputation algorithms

Value	Description
mean/mode	The missing value was substituted by a mean value (for a numeric variable) or a mode (for a logical or nominal variable) across all non-missing values in the CV subset.
W-REPTree	The missing values were estimated by applying a model trained to predict them. In this setting the model was a WREPTree – a fast decision tree learner that builds a decision or regression tree using information gain or variance, respectively, and prunes it using reduced-error pruning (with backfitting) [6].
k-NN	The missing values were estimated by applying a model trained to predict them. In this setting the model was k-NN algorithm that finds *k* neighbors (k=7) and classifies the unknown case by a majority vote (for nominal variable) or calculates the mean (for the numeric variable) of the found neighbor.

Table 5. Settings for outlier elimination algorithms

Value	Description
No	No outlier elimination algorithm performed, i.e., no cases removed with this setting.
INFLO lazy and greedy	Outlier elimination performed with influenced outlierness (INFLO) algorithm - a local density algorithm that considers the neighbors and the reverse neighbors when estimating the local density of a given point. It is also based on the nearest neighbors set. The normal cases have a calculated outlier score of approximately 1, while outliers have a value greater than 1 [32]. The 'lazy' setting means that cases with outlier score ≥ 1.8 were treated as outliers (i.e., very few cases) and removed from the dataset. The 'greedy' setting means that values with outlier score ≥ 1.2 were treated as outliers (i.e., more cases than with the previous setting) and removed from the dataset.
CBOF lazy and greedy	Outlier elimination performed with the outlier score based on Connectivity Based Outlier Factor – an algorithm that improves the effectiveness of a local outlier factor (LOF) algorithm when a pattern itself has similar neighborhood density as an outlier. The normal cases have a calculated outlier score of approximately 1, while outliers have a value greater than 1 [32], [38]. The 'lazy' setting means that cases with outlier score ≥ 1.5 were treated as outliers (i.e., very few cases) and removed from the dataset. The 'greedy' setting means that values with outlier score ≥ 1.2 were treated as outliers (i.e., more cases than with the previous setting) and removed from the dataset.

Table 6. Settings for value normalization

Value	Description
no	No value normalization was performed.
z-score	Numeric values were transformed by a statistical normalization with a formula $(value - mean)/standard\ deviation$.

Table 7. Settings for automated feature selection

Value	Description
all	All variables were used – no variable was removed by this component.
weights by information gain ratio	For each variable the information gain ratio was calculated – the higher value indicates the variable as the more relevant. Then top *k* variables were kept in the dataset while others were treated as not relevant and removed. The following values of *k* were considered: 3, 5, 10, 15, 20, and 30.
MRMR-EFS	A mixture of Minimum Redundancy Maximum Relevance (MRMR) with a correlation-based method using ensembles for feature selection (EFS). An algorithm iteratively adds a variable with the most information regarding the target variable and the least redundancy to the already selected variables [4], [33]. Top *k* variables were kept in the dataset while others were treated as not relevant and removed. The following values of *k* were considered: 3, 5, 10, 15, 20, and 30.

Table 8. Settings for classifiers[†] [6], [32]

Value	Description
W-ZeroR	A simple and naïve classifier that ‘predicts’ the mode (for a nominal class as in the case of this study). Used as a baseline for comparisons with ‘real’ predictive classifiers.
Naive Bayes	A high-bias, low-variance classifier, and it can build a good model even with a small data set. It is simple to use and computationally inexpensive. Assumes that given the value of the target the value of any attribute is independent of the value of any other attribute – but is considered to perform well even with violation of this assumption.
k-NN	The algorithm that finds *k* neighbors the closest to the predicted case and classifies this case by a majority vote (for nominal variable) of the found neighbor.
Decision Tree	A classifier with a tree-like collection of nodes where each node represents a splitting rule for a specific variable.
Random Forest	An ensemble of a number of random trees created and trained on bootstrapped subsets. A resulting set of trees delivers prediction by voting of all random trees [32].
W-OneR	An algorithm that builds a single rule for each predictive variable (discretizing numeric attributes) and chooses the one with the minimum-error for prediction [12].
W-PART	An algorithm for generating PART decision list that uses separate-and-conquer strategy – builds a partial C4.5 decision tree in each iteration and makes the ‘best’ leaf into a rule [7]
W-DTNB	A simple semi-naive Bayesian ranking method that combine naive Bayes with induction of decision tables [10].
W-J48	An open source implementation of an algorithm for generating a pruned or unpruned C4.5 decision tree [27].

[†] Classifiers prefixed with ‘W-‘ denote implementations in Weka [6], all remaining are native RapidMiner implementations [32]. The values of the main parameters for these classifiers used in this study are provided in Table 17 in Appendix 1.

Value	Description
W-NBTree	A hybrid of decision-tree and Naive-Bayes – the decision-tree nodes contain univariate splits as regular decision-trees, the leaves contain Naive-Bayesian classifiers. [21].
W-LMT	An algorithm for building logistic model trees – classification trees with logistic regression functions at the leaves [22], [37].
W-LADTree	Builds a multi-class alternating decision tree using the LogitBoost strategy [11].
W-BFTree	A boosting algorithm that builds a Best-First decision Tree classifier. Uses binary split for both nominal and numeric attributes. The method of 'fractional' instances is used to handle missing values [8], [35].
W-Logistic	An algorithm for building multinomial logistic regression model with a ridge estimator. An algorithm proposed in [23] with several modifications for calculating probability, log-likelihood and handling instance weights [6].
W-SimpleLogistic	A classifier for building linear logistic regression models. Uses the AIC criterion instead of cross-validation to prevent overfitting the model. A weight trimming heuristic is used for a significant speedup [37].
W-BayesNet	A Bayesian network – a probabilistic graphical model (directed acyclic graph) that contains a set of variables and their conditional dependencies [26]. Can be constructed from data using various search algorithms and quality measures.
W-LWL	Locally weighted learning – instance-based algorithm to assign instance weights which are then used by a specified weighted instances handler [1].
W-KStar	An instance-based classifier – predicted class is based on the class of those training instances similar to the predicted case, as determined by a similarity function. It uses an entropy-based distance function [3].
W-DecisionTable	A simple decision table majority classifier using the Inducer of Decision Table Majority (IDTM) – a decision table with default rule mapping to the majority class [20].
W-HyperPipes	For each category a HyperPipe is constructed that contains all points of that category (records the attribute bounds observed for each category). Test instances are classified according to the category that ‘most contains the instance’. Extremely simple algorithm, but has the advantage of being extremely fast.

Training classifiers and logging performance data: This step involved executing defined processes using prepared data and prediction schemes. To ensure a higher level of randomness of splits in the CV subset the whole process of training and testing these schemes was repeated ten times.

Analysis of results. The predictive accuracy in all analyses was based on the Matthews correlation coefficient (MCC) recommended to evaluate prediction accuracy in classification tasks [34]. The main benefits for using MCC are:

- Interpretation similar to the Pearson correlation coefficient – the same range of [-1, 1], a value +1 indicates perfect prediction, -1 indicates all predictions opposite to the reality, and 0 indicates prediction at the random level.
- It is regarded as a balanced measure which can be used when the variable is class (state) imbalanced.
- It is calculated from all four cells of the confusion matrix in contrast to often used F1-score, recall or precision.

This analysis was divided into three parts: The first part involved pairwise comparisons of schemes' components by visualizing the accuracy distributions for all settings of pairs of scheme's components. This paper discusses three the most interesting and useful (in the author's opinion) pairs of components investigated but more such analyses were performed.

The second part involved identification of ten best performing schemes in both CV and test subsets. Its first goal was to investigate the highest level of accuracy that can be expected to obtain using created prediction schemes. The second goal was to investigate which settings and their combinations for particular components of prediction schemes deliver the most accurate results.

The third part was focused on comparing schemes' components by ranks using four predictive accuracy measures: MCC, area under ROC curve, Cohen's kappa coefficient, and balance [36]. As mentioned earlier, the MCC is a recommended evaluation measure. However, other measures highlight other aspects of accuracy, thus they also were included in this third part of analysis to produce the distribution of accuracy measures. The overall performance of the settings in each component was evaluated using the *win-tie-loss* procedure (Algorithm 1) adapted from [19] and [24].

Algorithm 1. Calculation of overall performance

```
for-each pass p do
   for-each accuracy measure $a_i$ do
      for-each component setting $c_j$ do
         run Wilcoxon signed-rank test against all other component settings $c_k$
         adjust p-values with Bonferroni procedure
         determine win, tie or loss
            $win_{pij} \leftarrow$ if adjusted p-value $\leq 0.05$ AND $Median(a_{pij}) > Median(a_{pik})$
            $loss_{pij} \leftarrow$ if adjusted p-value $\leq 0.05$ AND $Median(a_{pij}) < Median(a_{pik})$
            $tie_{pij} \leftarrow$ if (adjusted p-value $\leq 0.05$ AND $Median(a_{pij}) = Median(a_{pik})$ OR
                         adjusted p-value $\geq 0.05$
calculate $Sum(win_j)$, $Sum(loss_j)$, $Sum(tie_j)$
$difference_j \leftarrow Sum(win_j) - Sum(loss_j)$
return $Sum(win_j)$, $Sum(loss_j)$, $Sum(tie_j)$, $Sum(difference_j)$
```

This procedure was applied to the prediction results from the CV and test subsets joined together. It involves repeatable running of the Wilcoxon signed-rank test, applying Bonferroni correction procedure. Its main step uses prediction measures to calculate the number of 'wins', 'ties' and 'losses' – pairwise for each setting of a component of a prediction scheme against all other settings in this component. Thus, the number of 'wins' and 'losses' tells how many times a particular setting of a component delivered predictions statistically significantly better or worse, respectively, than other settings. The number of 'ties' tell how many times a particular setting delivered predictions not statistically different from other settings or when the median value of particular evaluation measure was equal compared to the other setting. The value of the 'difference' is the difference between the number of 'wins' and the number of 'losses' for a particular setting of a component.

Preparation of prediction environment and running predictions were performed using RapidMiner [32]. Other steps were performed in R [28].

3. Related work

The earlier study [30] that is the closest to the current one involved a similar research process to the one described in Section 2 and on the same dataset. That work focused on a single aspect of user satisfaction, i.e, with the ability of system to meet stated objectives, not on predicting aggregated user satisfaction as in the current study. That study also evaluated fewer schemes (288) compared to 15,600 in the current study. The schemes in the earlier study were also simpler – they did not consider outlier elimination and value normalization. Also, in the previous work each scheme was used once on the CV and test subsets; in the current study this process was repeated ten times for each scheme. Finally, the analysis of results in earlier study was simpler and did not contain three steps performed here. Thus, the current study is a significant extension to [30]. That study reports that the median accuracy of predictions was MCC=0.30 for the validation subset and MCC=0.24 for the test subset. Two schemes with the highest accuracy (MCC=0.71) involve no manual feature preselection, imputation of missing values by mean/mode, number of features selected was 30, and classifiers were W-LMT and W-SimpleLogistic, respectively.

A different study [29], that also involved the use of ISBSG dataset, used a classifier of CN2 rules. Achieved prediction accuracy in a validation subset for 10-fold cross-validation was MCC=0.42.

Two studies involved developing a Bayesian network. The first [5] was focused on predicting resources. The second [31] was focused on predicting various software quality attributes defined in the ISO 25010 standard [15]. In both models user satisfaction was one of the target predictive variables. Unfortunately, none of these studies report achieved accuracy predictions.

Another study [2] compared various schemes to predict project outcome, defined as either 'success' or 'failure'. The authors made the following findings: feature selection using information gain score improves accuracy, statistical and ensemble classifiers are robust for predicting project outcome, and Random Forest produced on average the most accurate predictions.

4. Limitations and threats to validity

Results in this study are subject to some limitations and threats to validity. ISBSG dataset is not a random sample from population of software projects, thus no generalization of results or conclusions can be performed to other projects. Specifically, the ISBSG contains data on software projects from different contributors who wish to share the data on their projects with others. The investigated version of the dataset does not contain certain types of projects, e.g. in military application area or modern mobile applications.

Secondly, the author made arbitrary decisions on treatment of several variables, e.g. those with lots of missing data, transformations or adjustments of values. Also, the author made arbitrary decisions on the definition of schemes and their components. The author used own experience and recent literature when making such decisions to limit the bias caused by subjectivity. As a result of such human-based data preparation, to treat the test data as 'unseen' until the evaluation of predictions, the process was performed once – i.e. the repetition of data partitioning was performed ten times but only within the cross-validation subset, the test subset was defined once for all these passes of the process.

The last step of the analysis involved repeated execution of the statistical test. To mitigate the inflated possibility of false alarms this analysis involved adjusting p-values using a Bonferroni procedure.

The aggregated satisfaction used in predictions averages over eight individual user satisfaction variables from original dataset. This calculated mean treats each of them with the same weight while in specific projects some individual variables may be more important overall than others. However, the dataset did not contain information on the importance of particular satisfaction variables.

Finally, to combine certain values this study often used the mean which is a non-robust measure of central location. Recent literature recommend using robust measures such as the trimmed mean [18]. However, the software packages used in this study very often had no option/setting to use a more robust measure than the mean. Using robust measure would require significant effort to re-implement or extend certain operators of these packages. While this path is worth examining it is planned for the future work.

Also, the author made an arbitrary decision to use the mean when calculating the aggregated user satisfaction from eight distinct variables. This choice was motivated by the fact that in the author's opinion this aggregated value should be prone to and reflect negative deviations, e.g. when for a given project one of these satisfaction variables is significantly worse than all others it is likely that overall satisfaction would be poor. Plans for future work include examining this aspect in details.

5. Results

5.1. Pairwise comparisons of schemes' components

Creating prediction scheme involved two steps of feature selection: initially performed manually by a domain expert and then automated performed using a tool-supported algorithm. Results in Figure 2 show that on average for the CV subset for each setting of automated feature selection algorithm the prediction accuracy was higher when the features were earlier preselected by a human expert than when no such preselection was performed. Thus, automated feature selection may not be sufficient on its own. In the test subset, on average, schemes with automated feature selection usually performed no worse than with no such preselection (by investigating the quartiles) but here the differences were smaller than in the CV subset, sometimes hardly noticeable.

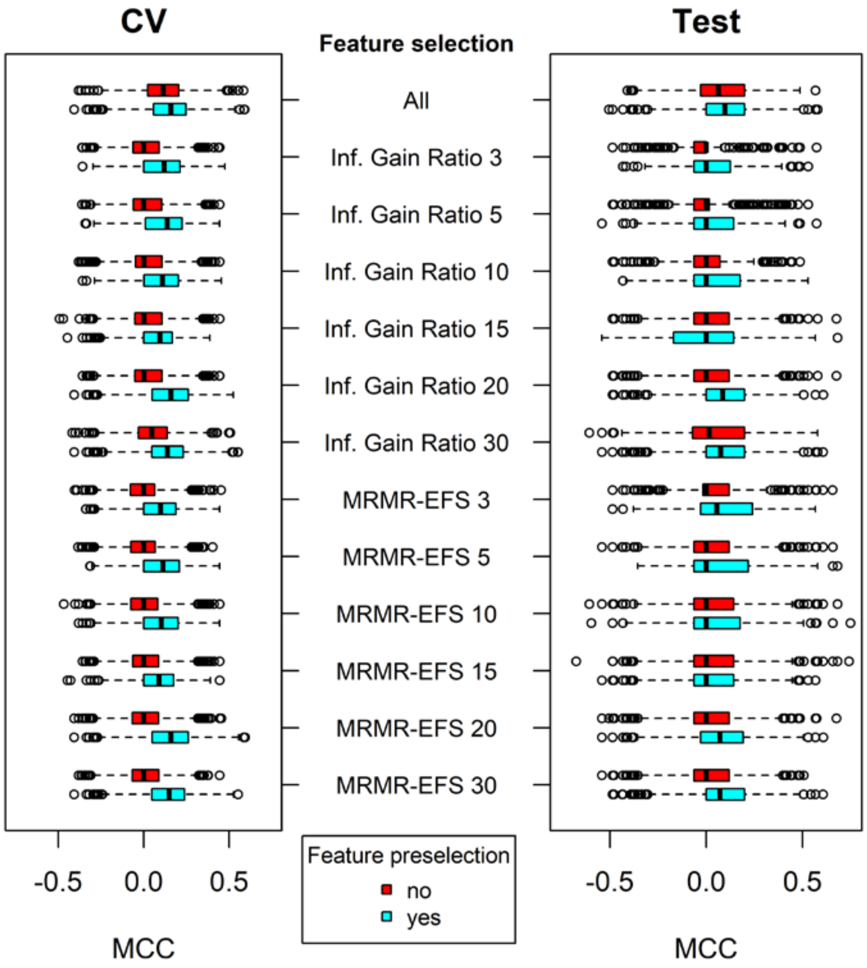

Figure 2. Accuracy by manual and automated feature selection

The second comparison involved a number of features in automated selection and a classifier (Figure 3). In the CV subset, classifiers produce different patterns of accuracy for varying number of features selected. Several classifiers are more accurate with the increasing number of features selected, i.e., Naïve-Bayes, k-NN, Random Forest, W-KStar, W-Decision Table, and W-Hyper Pipes. Other group of classifiers, e.g. W-OneR, LMT, W-LADTree, W-BFTree, on average deliver surprising predictive performance – the most accurate for very few or a lot of features selected and the least accurate for medium number of features selected. Decision Tree was the only classifier that on average performed less accurate with increasing number of features selected. In the test subset the results were noisy and observations for most classifiers in the CV subset were not confirmed in the test subset.

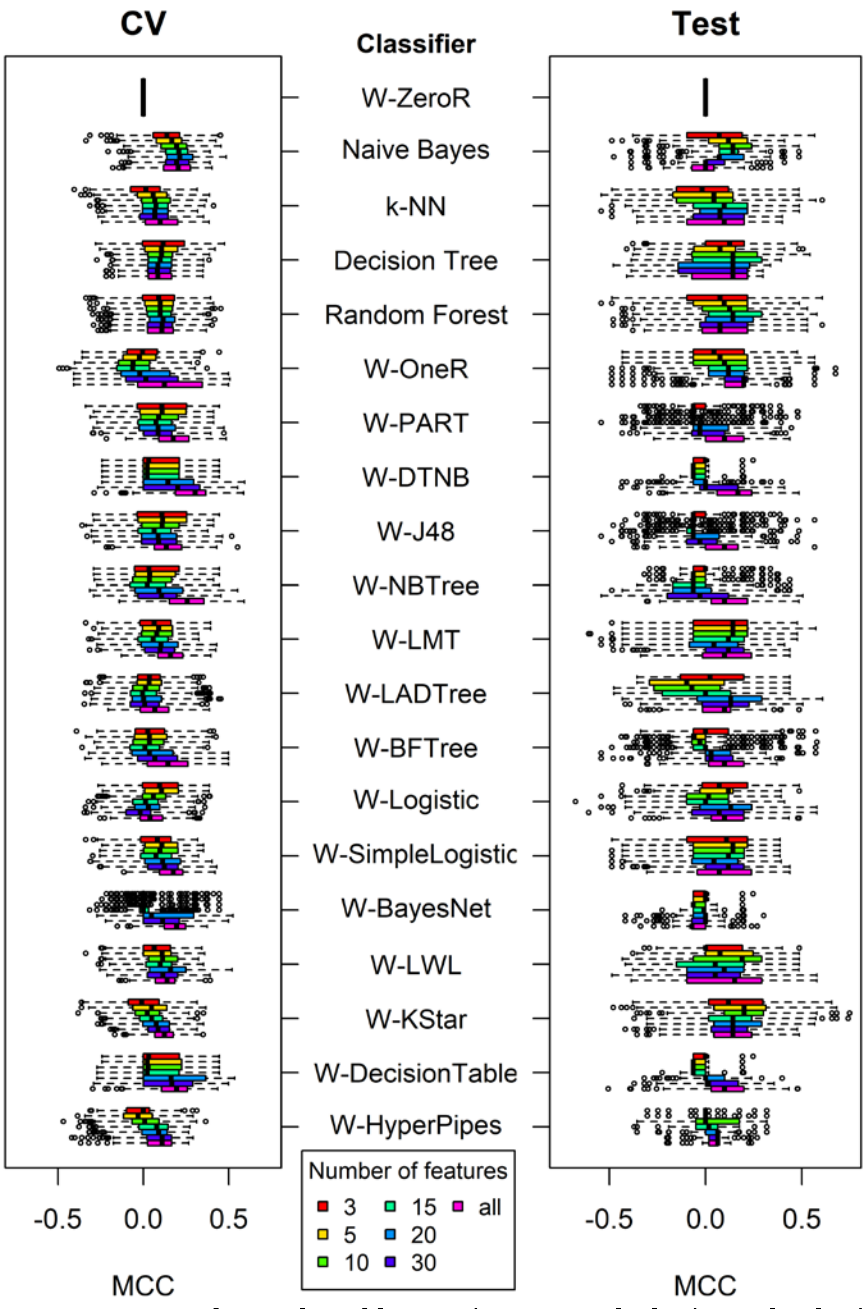

Figure 3. Accuracy by number of features in automated selection and a classifier

The last pairwise comparison involved missing value imputation and automated feature selection (Figure 4). In both the CV and test subsets for information gain ratio as automated feature selection on average a procedure of filling missing values by mean/mode performed the better with the increasing value of number of features selected – both in absolute scale and comparing to other procedures of handling missing data. For MRMR-EFS with different values of features no such clear tendency for missing value procedures was observed – yet, in the CV subset all missing value handling procedures seem to benefit from a higher number of features.

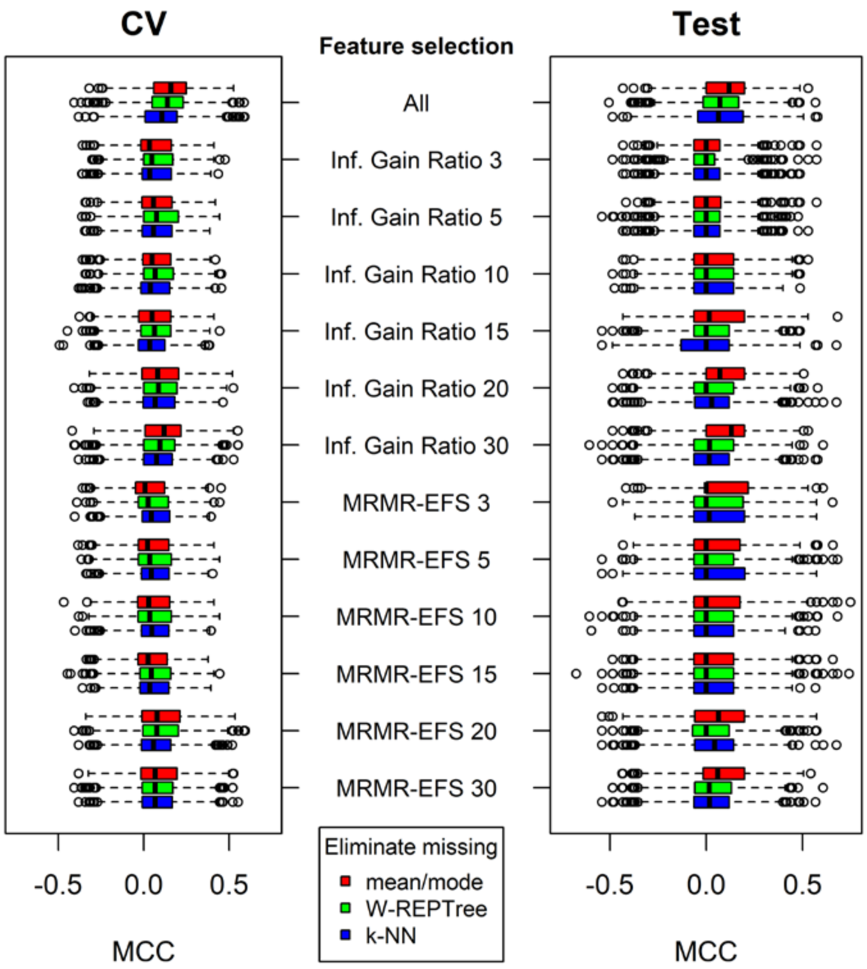

Figure 4. **Accuracy by missing value imputation and automated feature selection**

5.2. Best and worst performing schemes

The next part of this analysis involve the identification of the best performing schemes in the CV subset (Table 9) and the test subset (Table 10). These schemes were identified based on the highest mean MCC across all ten passes of in evaluation process. In the CV subset all top five schemes achieved the same mean MCC of 0.500 and shared the same setting for all components except automated feature selection. Surprisingly, all five of them used a very simple classifier – W-OneR.

Table 9. Best performing schemes for the CV subset

Feature pre-selection	Impute missing	Outlier elimination	Normalization	Feature selection	Classifier	MCC
yes	mean/mode	INFLO lazy	no	Inf. Gain Ratio 20	W-OneR	0.500
yes	mean/mode	INFLO lazy	no	MRMR-EFS 20	W-OneR	0.500
yes	mean/mode	INFLO lazy	no	Inf. Gain Ratio 30	W-OneR	0.500
yes	mean/mode	INFLO lazy	no	MRMR-EFS 30	W-OneR	0.500
yes	mean/mode	INFLO lazy	no	All	W-OneR	0.500
no	mean/mode	CBOF lazy	no	All	W-OneR	0.490
Yes	mean/mode	CBOF greedy	no	MRMR-EFS 20	W-OneR	0.483
Yes	mean/mode	CBOF greedy	no	Inf. Gain Ratio 30	W-OneR	0.483
yes	mean/mode	CBOF greedy	no	MRMR-EFS 30	W-OneR	0.483
yes	mean/mode	CBOF greedy	no	All	W-OneR	0.483

Table 10. Best performing schemes for the test subset

Feature pre-selection	Impute missing	Outlier elimination	Normalization	Feature selection	Classifier	MCC
no	W-REPTree	No	z-score	MRMR-EFS 15	W-KStar	0.598
yes	mean/mode	CBOF greedy	no	MRMR-EFS 5	W-KStar	0.527
yes	mean/mode	CBOF greedy	z-score	MRMR-EFS 5	W-KStar	0.527
yes	mean/mode	CBOF lazy	no	MRMR-EFS 5	W-KStar	0.520
yes	mean/mode	CBOF lazy	z-score	MRMR-EFS 5	W-KStar	0.520
no	mean/mode	No	z-score	MRMR-EFS 15	W-KStar	0.489
no	mean/mode	No	z-score	Inf. Gain Ratio 3	W-KStar	0.477
yes	mean/mode	INFLO lazy	no	Inf. Gain Ratio 5	W-KStar	0.471
yes	mean/mode	CBOF lazy	no	Inf. Gain Ratio 5	W-KStar	0.471
yes	mean/mode	INFLO lazy	z-score	Inf. Gain Ratio 5	W-KStar	0.471
yes	mean/mode	CBOF lazy	z-score	Inf. Gain Ratio 5	W-KStar	0.471

In the test subset different schemes performed the best, i.e. with different settings for most schemes' components. Surprisingly, in the test subset the best schemes achieved higher accuracy than the best in the CV subset. The W-KStar turned to be the classifier with the best performance in all top schemes. The best performing scheme involved imputation of missing values with W-REPTree. However, this method of handling missing values appeared only once in top performing schemes – all other schemes involved substitution with mean/mode. Quite surprisingly, the best performing schemes usually were based on very few features automatically selected: 5 or 3, with two exceptions of 15.

These results show that often quite simple schemes (or their components) perform the most accurate, e.g., W-OneR classifier for the CV subset and using very few features in automated feature selection in the test subset. In addition, simple substitution of missing values with mean/mode appeared to be the best choice in almost all top performing schemes in both the CV and test subsets.

5.3. Comparing schemes' components by ranks

The last and the most important step of this analysis involved each component by comparing all settings used in a given component using the procedure explained in Section 2. The results illustrated in this Section were sorted by the last column ('difference') from the best to the worst. Table 11 shows that under this procedure of comparison using manual feature preselection have always performed better than not performing such selection. Thus, a recommendation for performing such feature preselection is strongly supported by obtained results.

Table 11. Aggregated accuracy for feature preselection

Preselection	wins	ties	losses	difference
Yes	40	0	0	40
No	0	0	40	-40

Table 12 shows that filling missing values by mean/mode on average performed the best compared with other missing value handling procedures. However, the difference between all three settings was not high and even the least performing k-NN achieved 23 'wins' of 80 possible (the best average/mode achieved 34 'wins').

Table 12. Aggregated accuracy for missing value handling procedure

Impute missing	wins	ties	losses	difference
mean/mode	34	22	24	10
W-REPTree	33	17	30	3
k-NN	23	21	36	-13

As illustrated in Table 13, the 'lazy' version of CBOF outlier detection algorithm, i.e. detecting very few outliers, performed clearly the best compared to other algorithms. For both CBOF and INFLO the 'lazy' versions performed better than 'greedy' suggesting that the latter might have removed too many cases used to train the model. Three of four algorithms delivered worse results than not applying any outlier elimination algorithm at all.

Table 13. Aggregated accuracy for outlier elimination

Outlier removal	wins	ties	losses	difference
CBOF lazy	124	22	14	110
No	99	7	54	45
CBOF greedy	60	31	69	-9
INFLO lazy	58	17	85	-27
INFLO greedy	19	3	138	-119

Table 14 shows that performing standardization with z-score resulted in better prediction in very few cases. In most cases, i.e., 34 of 40, there was no difference if such standardization was applied or not.

Table 14. Aggregated accuracy for normalization

Normalize	wins	ties	losses	difference
z-score	6	34	0	6
no	0	34	6	-6

Table 15 shows that automated feature selection using MRMR-EFS with 20 features performed clearly the best compared to other feature selection algorithms. However, it is difficult to draw conclusions for other settings used. For example, usually the higher number of features used resulted in better accuracy – but MRMR-EFS with 30 features selected and using all available features preformed quite poorly, i.e., as one of the worst performing algorithms. On the other hand, MRMR-EFS with only five features selected performed better than nine of other settings investigated. Obtained results confirm a widely accepted practice that training the model on all available features, i.e. with no automated feature selection, delivers usually quite poor results – in this study the usage of all features was outperformed by nine other analyzed settings and it outperformed only three other settings.

On average the best performing classifier was W-SimpleLogistic (Table 16). The W-LWL achieved similar number of 'wins' but significantly higher number of 'losses' (213 vs 157). The Naïve Bayes achieved the third highest number of 'wins' but due to high number of 'losses' it was ranked as the average classifier. In fact, its very low number of 'ties' show that it performed usually as either significantly better or significantly worse compared to other classifiers. Surprisingly, the W-BayesNet performed as the worst classifier – this may be due to not optimal settings used in training the classifier and will be further investigated later.

Table 15. Aggregated accuracy for automated feature selection

Feature selection	wins	ties	losses	Difference
MRMR-EFS 20	334	146	0	334
Inf. Gain Ratio 30	284	91	105	179
Inf. Gain Ratio 20	249	138	93	156
MRMR-EFS 5	213	197	70	143
MRMR-EFS 10	191	189	100	91
Inf. Gain Ratio 5	131	176	173	-42
Inf. Gain Ratio 10	119	190	171	-52
MRMR-EFS 15	108	203	169	-61
MRMR-EFS 30	115	178	187	-72
All	180	28	272	-92
Inf. Gain Ratio 15	108	149	223	-115
Inf. Gain Ratio 3	78	144	258	-180
MRMR-EFS 3	32	127	321	-289

Table 16. Aggregated accuracy for classifiers

Classifier	wins	ties	losses	Difference
W-SimpleLogistic	422	181	157	265
W-LWL	413	134	213	200
W-PART	369	212	179	190
W-J48	338	222	200	138
W-KStar	366	163	231	135
W-HyperPipes	393	106	261	132
W-BFTree	352	169	239	113
W-LMT	330	198	232	98
Decision Tree	330	185	245	85
Naive Bayes	401	25	334	67
Random Forest	291	223	246	45
W-Logistic	310	133	317	-7
W-DTNB	278	156	326	-48
W-ZeroR	335	41	384	-49
k-NN	216	217	327	-111
W-DecisionTable	232	166	362	-130
W-NBTree	200	183	377	-177
W-OneR	222	93	445	-223
W-LADTree	192	121	447	-255
W-BayesNet	84	124	552	-468

6. Conclusions and future work

Performed analyses led to achieving the goals stated in Section 1, i.e., to investigate what accuracy can be achieved in predicting aggregated user satisfaction, and to determine which settings for schemes' components deliver the most accurate predictions. The identification of the most accurate schemes may be defined in different ways. Thus, this study involved three analyses. The first one covered pairwise comparisons of scheme's components to observe how settings for one component influence the accuracy of predictions with varying settings of yet another component. The second one identified which component settings formed the most accurate schemes and if the same schemes were the most accurate in the CV and test subsets. The third aimed at ranking settings for each component – to deliver guidance on which settings usually deliver the most accurate predictions.

Obtained results enable to draw the following conclusions:

1. The best performing schemes predict aggregated user satisfaction with accuracy of MCC≈0.5 in the CV stage and MCC≈0.6 in testing. Compared to typical studies in software engineering, e.g. effort and defect prediction, these values are low but open the possibility of improvement with planned extensions explained below.
2. Substantial set of prediction schemes deliver very low accuracy, with MCC<0, i.e. with prediction worse than random selection. This is due to the construction of certain components, i.e. specific combinations of component settings are not sensible but were still included in the analysis of results.
3. Performance of most components of prediction schemes varies between the CV and test subsets. It is likely that the projects in the test subset are different in their characteristics than those in the CV subset.
4. Performing normalization on data values rarely results in more accurate predictions.
5. Although prediction schemes contain automated feature selection component, on average manual feature selection improves the accuracy of predictions.
6. Usually selecting 20 or 30 attributes for automated feature selection provide the most accurate predictions.
7. Identifying component settings that provide the most accurate results based on a single evaluation measure (MCC) and for a single data subset produce different outcome than when using more evaluation measures and both data subsets jointly.

Obtained results may serve as a base for further research focused on building accurate schemes for predicting user satisfaction as they demonstrate how particular techniques and algorithms perform in combination with other components and against competing techniques and algorithms for the same component type. Successful construction of such schemes would enable future practical applications, i.e. creating tools for informing software project managers and partially enhancing decision support functions.

The main extension of this study planned for future is to replicate it with enhancement of prediction schemes such that the parameters for components, especially for classifiers, will not be fixed but be optimized. Also, this replication may also involve performing a range of passes of the process but each time with different split to the CV and test subset – this requires automating data preparation step performed here by a human expert. Another planned extension involve the analysis of impact of particular features used in training on the accuracy of results as well as identification of the most influential features. Finally, plans for future work involve dealing with some issues raised in Section 4 – most notably with different ways of aggregating satisfaction variables than their mean and with replacing the usage of a mean to the more robust measure in various algorithms/components of prediction schemes.

References

[1] Atkeson C.G., Moore A.W., Schaal S., Locally Weighted Learning, *Artificial Intelligence Review*, **11**, 1-5, 1997, 11-73.

[2] Cerpa, N., Bardeen, M., Astudillo, C. A., Verner, J., Evaluating different families of prediction methods for estimating software project outcomes, *Journal of Systems and Software*, **112**, 2016, 48–64.

[3] Cleary J.G., Trigg L.E., K*: an instance-based learner using and entropic distance measure, in: *Proceedings of the Twelfth International Conference on International Conference on Machine Learning (ICML'95),* Armand Prieditis and Stuart J. Russell (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, 108-114.

[4] Ding, C.H.Q., Peng, H., Minimum redundancy feature selection from microarray gene expression data, in: *Proc. the 2nd IEEE Comp. Society Bioinformatics Conf.*, Stanford, CA, IEEE Comp. Society, Los Alamitos, 2003, 523–529.

[5] Fenton, N., Marsh, W., Neil, M., Cates, P., Forey, S., Tailor, M., Making Resource Decisions for Software Projects, in: *Proceedings of the 26th International Conference on Software Engineering*, IEEE Computer Society, Washington, DC, 2004, 397–406.

[6] Frank E., Hall M.A., Witten I.H., *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*, Morgan Kaufmann, Fourth Edition, 2016, http://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf, last accessed 2018/05/22.

[7] Frank E., Witten I.H., Generating Accurate Rule Sets Without Global Optimization. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98)*, Jude W. Shavlik (Ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998, 144-151.

[8] Friedman J., Hastie T., Tibshirani R., *Special Invited Paper. Additive Logistic Regression: A Statistical View of Boosting*, The Annals of Statistics, **28**, 2, 2000, 337-374.

[9] Garcés, L., Ampatzoglou, A., Avgeriou, P., Nakagawa, E.Y., *Quality attributes and quality models for ambient assisted living software systems: A systematic mapping*, Information and Software Technology, **82**, 2017, 121-138.

[10] Hall M., Frank E., Combining Naive Bayes and Decision Tables, in: D.L. Wilson & H. Chad (Eds), *Proceedings of Twenty-First International Florida Artificial Intelligence Research Society Conference*, AAAI Press, Coconut Grove, Florida, USA, 2008, 318-319.

[11] Holmes G., Pfahringer B., Kirkby R., Frank E., Hall M., Multiclass Alternating Decision Trees, in: *Proceedings of the 13th European Conference on Machine Learning (ECML '02)*, Tapio Elomaa, Heikki Mannila, and Hannu Toivonen (Eds.). Springer-Verlag, London, UK, 2002, 161-172.

[12] Holte R.C., *Very simple classification rules perform well on most commonly used datasets*. Machine Learning. **11**, 1993, 63-91.

[13] *ISBSG Repository Data Release 11*. International Software Benchmarking Standards Group, 2009.

[14] Idri, A., Bachiri, M., Fernández-Alemán, J.L., *A Framework for Evaluating the Software Product Quality of Pregnancy Monitoring Mobile Personal Health Records*, Journal of Medical Systems, **40**, 3, 2016, art. no. 50, 1-17.

[15] *ISO/IEC: Software engineering Software product Quality Requirements and Evaluation (SQuaRE) System and software quality models*, volume ISO/IEC 25010:2011(E), 2011.

[16] Jin W., Tung A.K.H., Han J., Wang W., Ranking Outliers Using Symmetric Neighborhood Relationship, in: Ng WK., Kitsuregawa M., Li J., Chang K. (eds) *Advances in Knowledge Discovery and Data Mining. PAKDD 2006*. Lecture Notes in Computer Science, vol 3918. Springer, Berlin, Heidelberg, 2006.

[17] Jones C., *Applied Software Measurement: Global Analysis of Productivity and Quality*, McGraw-Hill Education, 3rd edition, 2008.

[18] Kitchenham B.A., Madeyski L., Budgen D., Keung J., Brereton P., Charters S., Gibbs S., Pohthong A., *Robust Statistical Methods for Empirical Software Engineering*, Empirical Software Engineering, **22**, 2, 2017, 579-630.

[19] Kocaguneli E., Menzies T., Bener A., Keung J. W., *Exploiting the Essential Assumptions of Analogy-Based Effort Estimation*, IEEE Transactions on Software Engineering, **38**, 2, 2012, 425–438.

[20] Kohavi R., The power of decision tables, in: *Proceedings of the 8th European Conference on Machine Learning (ECML'95)*, Nada Lavrač and Stefan Wrobel (Eds.). Springer-Verlag, Berlin, Heidelberg, 1995, 174-189.

[21] Kohavi R., Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid, in: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, Evangelos Simoudis, Jiawei Han, and Usama Fayyad (Eds.). AAAI Press, 1996, 202-207.

[22] Landwehr N., Hall M., Frank E., *Logistic Model Trees*. Machine Learning, **59**, 1-2, 2005, 161-205.

[23] Le Cessie S., Van Houwelingen J., *Ridge Estimators in Logistic Regression*, Journal of the Royal Statistical Society. Series C (Applied Statistics), **41**, 1, 1992, 191-201.

[24] Menzies T., Jalali O., Hihn J., Baker D., Lum K., *Stable rankings for different effort models*, Automated Software Engineering, **17**, 4, 2010, 409–437.

[25] Olsina, L., Lew, P., Dieser, A., Rivera, B., *Updating quality models for evaluating new generation web applications*, Journal of Web Engineering, **11**, 3, 2012, 209-246.

[26] Pearl J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Representation and Reasoning Series (2nd printing ed.). San Francisco, California: Morgan Kaufmann, 1988.

[27] Quinlan R., *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[28] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2017.

[29] Radlinski L., How software development factors influence user satisfaction in meeting business objectives and requirements?, in: Madeyski, L., Ochodek, M. (eds.), *Software Engineering from Research and Practice Perspectives*, chapter 6, Nakom, Poznan-Warszawa, 2014, 101–119.

[30] Radliński Ł., *Preliminary evaluation of schemes for predicting user satisfaction with the ability of system to meet stated objectives*, Journal of Theoretical and Applied Computer Science, **9**, 2, 2015, 32–50.

[31] Radlinski L., *Towards expert-based modeling of integrated software quality*, Journal of Theoretical and Applied Computer Science, **6**, 2, 2012, 13–26.

[32] *RapidMiner Studio*, https://rapidminer.com/products/studio/, last accessed 2018/05/22.

[33] Schowe B., Morik K., Fast-Ensembles of Minimum Redundancy Feature Selection, in: Okun O., Valentini G., Re M. (eds) *Ensembles in Machine Learning Applications. Studies in Computational Intelligence*, vol 373. Springer, Berlin, Heidelberg, 2011.

[34] Shepperd M., Bowes D., Hall T., *Researcher Bias: The Use of Machine Learning in Software Defect Prediction*. IEEE Transactions on Software Engineering, **40**, 2014, 603–616.

[35] Shi H., *Best-first Decision Tree Learning*, Thesis, Master of Science. The University of Waikato, Hamilton, New Zealand, 2007.

[36] Song Q., Jia Z., Shepperd M., Ying S., Liu J., *A General Software Defect-Proneness Prediction Framework*, IEEE Transactions on Software Engineering, **37**, 3, 2011, 356-370.

[37] Sumner M., Frank E., Hall M., Speeding up logistic model tree induction, in: *Proceedings of the 9th European conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'05)*, Alípio Mário Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and João Gama (Eds.). Springer-Verlag, Berlin, Heidelberg, 2005, 675-683.

[38] Tang J., Chen Z., Fu A. W. C., Cheung, D. W., Enhancing Effectiveness of Outlier Detections for Low Density Patterns, in: *Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*. Taipei, 2002, 535-548.

[39] Vargas J.A., García-Mundo L., Genero M., Piattini M., A systematic mapping study on serious game quality, in: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14)*, ACM, New York, 2014, Article no. 15.

Received 7.08.2018, Accepted 19.10.2018

Appendix 1. Parameter settings for classifiers

Table 17. Values of main parameters used for classifiers

Classifier	Parameters
W-ZeroR	—
Naive Bayes	laplace correction = true
k-NN	k = 7, measure types = MixedMeasures, mixed measure = Mixe-dEuclideanDistance
Decision Tree	criterion = gain ratio, maximal depth = 20, apply pruning = true, confidence = 0.25, apply prepruning = true, minimal gain = 0.1, minimal leaf size = 2, minimal size for split = 4, number of pre-pruning alternatives = 3
Random Forest	number of trees = 10, criterion = gain ratio, maximal depth = 20, apply pruning = true, confidence = 0.25, apply prepruning = true, minimal gain = 0.1, minimal leaf size = 2, minimal size for split = 4, number of prepruning alternatives = 3, guess subset ratio = true, voting strategy = confidence vote, other important settings as for DT
W-OneR	B = 6
W-PART	C = 0.25, M = 2, U = false
W-DTNB	X = 10, I = false
W-J48	C = 0.25, M = 2, A = true
W-NBTree	—
W-LMT	I = -1, M = 10, W = 0, A = true
W-LADTree	B = 10
W-BFTree	P = POSTPRUNED, M = 2, N = 5, C = 1.0
W-Logistic	R = 1.0E-8, M = 100
W-SimpleLogistic	I = 0, M = 500, H = 50, W = 0, A = true
W-BayesNet	D = -Q, Q = K2, E = SimpleEstimator
W-LWL	A = LinearNNSearch -A ”EuclideanDistance -R first-last”, K = 7, U = 4, W = SMO
W-KStar	B = 20, E = false, M = n (normal)
W-DecisionTable	S = BestFirst -D 1 -N 5, X = 5, I = false
W-HyperPipes	—