electronics

Article

The Impact of Data Quality on Software Testing Effort Prediction

Łukasz Radliński

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; Tel.: +48-91-449-5656

Abstract: Background: This paper investigates the impact of data quality on the performance of models predicting effort on software testing. Data quality was reflected by training data filtering strategies (data variants) covering combinations of *Data Quality Rating*, *UFP Rating*, and a threshold of valid cases. Methods: The experiment used the ISBSG dataset and 16 machine learning models. A process of three-fold cross-validation repeated 20 times was used to train and evaluate each model with each data variant. Model performance was assessed using absolute errors of prediction. A 'win–tie–loss' procedure, based on the Wilcoxon signed-rank test, was applied to identify the best models and data variants. Results: Most models, especially the most accurate, performed the best on a complete dataset, even though it contained cases with low data ratings. The detailed results include the rankings of the following: (1) models for particular data variants, (2) data variants for particular models, and (3) the best-performing combinations of models and data variants. Conclusions: Arbitrary and restrictive data selection to only projects with *Data Quality Rating* and *UFP Rating* of 'A' or 'B', commonly used in the literature, does not seem justified. It is recommended not to exclude cases with low data ratings to achieve better accuracy of most predictive models for testing effort prediction.

Keywords: software testing; effort prediction; machine learning; ISBSG; data quality

Citation: Radliński, Ł. The Impact of Data Quality on Software Testing Effort Prediction. *Electronics* **2023**, *12*, 1656. https://doi.org/10.3390/electronics12071656

Academic Editors:Agnieszka Konys, Agnieszka Nowak-Brzezińska and Stanley C. Ahalt

Received: 6 February 2023
Revised: 10 March 2023
Accepted: 28 March 2023
Published: 31 March 2023

1. Introduction

The lifecycle of the software development process usually consists of various phases or activities related to project planning, requirements specification, analysis and designing, programming, testing, integration, deployment, and additional supportive activities. Since software projects are usually complex, time consuming, and costly, supporting them with a reliable prediction of the required effort is essential. The area of software development effort prediction has been extensively studied in software engineering literature [1–5]. Most of this research has been focused on the prediction of the total effort for the project. Limited literature focuses on predicting effort for particular phases, like testing, even though it is a significant and challenging area [6,7]. Thus, the first motivation for this study is related to the importance of the outcome variable for prediction, i.e., testing effort.

Studies on software development prediction use various private and publicly available datasets. This particular study used the widely known ISBSG dataset [8] because it contains critical data relevant to the goal of this study: *Data Quality Rating* and *UFP Rating*. These attributes reflect the data quality of a project in this dataset. In most studies on software effort prediction involving the ISBSG dataset, the authors follow the ISBSG guidelines [9] and filter the data by only including projects with *Data Quality Rating* and *UFP Rating* classified as A or B, denoting at least good quality [7,10–17]. Generally, performing analyses on the best possible data seems reasonable. However, such data filtering causes the dataset to be reduced, and predictions may be too optimistic as they are based on, and performed only for, projects with high data ratings. Still, projects with lower ratings also exist in the raw dataset but are rarely considered in predictive studies. On the other hand, machine learning models usually perform better when they are trained on larger datasets. Thus, using such a reduced dataset means that these models may not reveal their potential.

Existing literature on the treatment of missing values for effort prediction is limited, especially concerning the testing area. There are various approaches to solving this problem. A simple, and frequently followed solution, is to use only those attributes that have no missing values. The opposite solution is to include all available attributes, even though some may have a high proportion of missing values, i.e., very few valid values. Most popular, and generally regarded as accurate, machine learning models require the training data to have no missing values. Then, these values may be replaced according to some rules. Thus, the spectrum of possible solutions to the problem with missing values ranges from setting a threshold of valid values from 1, indicating that only attributes with no missing values are included, down to 0, indicating that all attributes are included regardless of their fractions of missing values. Intermediate values are also possible. For example, if a threshold of valid values equals 0.3, only attributes for which the proportion of valid values is at least 0.3 would be included. The impact of these strategies for solving the problem of missing values on software testing effort prediction has yet to be explored in the literature.

Overall, existing literature does not provide clear information and recommendations on using particular predictive models and strategies for dataset filtering. Therefore, based on the explained motivations, the goal of this paper was to investigate the impact of training data quality on the performance of models for software testing effort prediction. The general research question for this study is: ***Which of the following is it better to use to train predictive models for software testing effort prediction: (1) the entire available dataset containing some low-quality data or (2) on a subset of the entire dataset containing selected higher-quality data?*** This study covered three aspects of data quality related to data filtering: two attributes in the ISBSG dataset, namely *Data Quality Rating* and *UFP Rating*, and a threshold of valid values. These three elements were used in the definitions of data handling strategies, named data variants herein. These data variants reflect the extreme options of these strategies and the intermediate alternatives. Thus, this study explored how using particular data variants influences the accuracy of predictions from various predictive models.

The main contribution of the paper is the investigation of the performance of frequently used predictive models for software testing effort prediction and several data variants. Specifically, this paper investigates model performance for particular data variants and the performance of data variants for particular models, and identifies the best-performing combinations of models with data variants. Existing literature, neither generally on software project effort prediction, nor, specifically, on software testing effort prediction, does not report results of such experiments, especially at the scale employed herein, which investigates several predictive models and 40 data variants. Furthermore, this study justifies using particular data variants for each model to achieve better predictive accuracy.

This study investigates the following three main, and two supportive, specific research problems:

RP1: The performance of particular models for testing effort prediction—This involved creating a ranking of models based on a comparison of prediction accuracy between the models, both for each data variant and across all of them. In addition, the performance of models for various conditions reflected in training data variants was explored.

RP1A: The stability of performance of models across passes—As a follow-up to the RP1, this was motivated by previous analyses that investigated a different outcome attribute [18], and which revealed the fluctuating performance of most predictive models depends on a particular iteration of data split into CV and test subset.

RP2: Identification of good and poor data variants for particular models for testing effort prediction—This involved ranking data variants, based on comparing prediction accuracy achieved with various training data variants for each predictive model.

RP2A: The stability of performance of data variants for particular models between CV and test data subsets—As a follow-up to the RP2, this involved investigating whether the data variants that delivered predictions at the specific rank (accurate or inaccurate) for particular models in the cross-validation subphase also delivered a similar rank of predictive accuracy on the test data subset. This was investigated for each model in each pass.

RP3: Identification of the best combinations of models and data variants— Investigation of the previous RPs delivered partial answers, as they focused on identifying the best-performing models for given data variants and best-performing data variants for given models. For a complete picture, it was also necessary to identify the best combinations of models and data variants.

This paper is organized as follows. Section 2 describes the data used in this study and the research process followed. Section 3 presents the obtained results. Section 4 discusses the obtained results and presents ideas for future work. Section 5 draws up the conclusions.

2. Materials and Methods

2.1. Overview of the Research Process

An overview of the research process followed in this study is illustrated in Figure 1. The main phases of this process were the following:

1. Data preprocessing—Section 2.2; the outcome of this phase, i.e., a prepared dataset—Section 2.3;
2. Preparation of the environment—Section 2.4; the outcomes of this phase, i.e., the definitions of data variants–Section 2.5, predictive models—Section 2.6, hyperparameter tuning grids—Section 2.7;
3. Model training and evaluation—Section 2.8;
4. Analysis of results—Section 2.9.

2.2. Data Preprocessing

The dataset used in this study was the ISBSG 2020 R1 [8]. The raw format contains data on 9592 software projects described by 253 attributes related to project size, effort, schedule, development and application environment, used documents and techniques, etc. Preparing this dataset for use in this study required performing data preprocessing. This phase was divided into two subphases: general and specific preprocessing. The former involved the whole raw dataset. The latter was specific to the goal of this study and, thus, was performed on the subset of the original dataset.

The general data preprocessing was performed to correct the dataset so that it could be used in this and other studies. Specifically, missing values originally encoded by states like 'don't know', 'not available', 'NA', etc., were encoded as a special indicator of a missing value (NA in R language). Obvious mistakes were corrected, e.g., spelling/typographical, inconsistent capitalisation for nominal attributes, and incorrect decimal places for numeric attributes. For major nominal attributes, similar states were encoded under a common value, e.g., *Application Type* states, like 'CRM', 'Contact Management', 'Customer Management', and 'Customer relationship management', were all encoded as the last of these states. An important step covered removing discrepancies between two or more attributes. For example, for a given project, there was data on effort spent on a particular development phase reported, but the attribute *Project Activity Scope* originally did not contain information that the particular activity (phase) had even been performed. The attributes with a very low fraction of valid cases (usually <10%) and many states but very few counts per state were removed. Attributes with unclear interpretation, no valid value, having only one state, and evident inconsistencies that could not be resolved were also removed. After this step, there were 191 attributes left in the dataset.

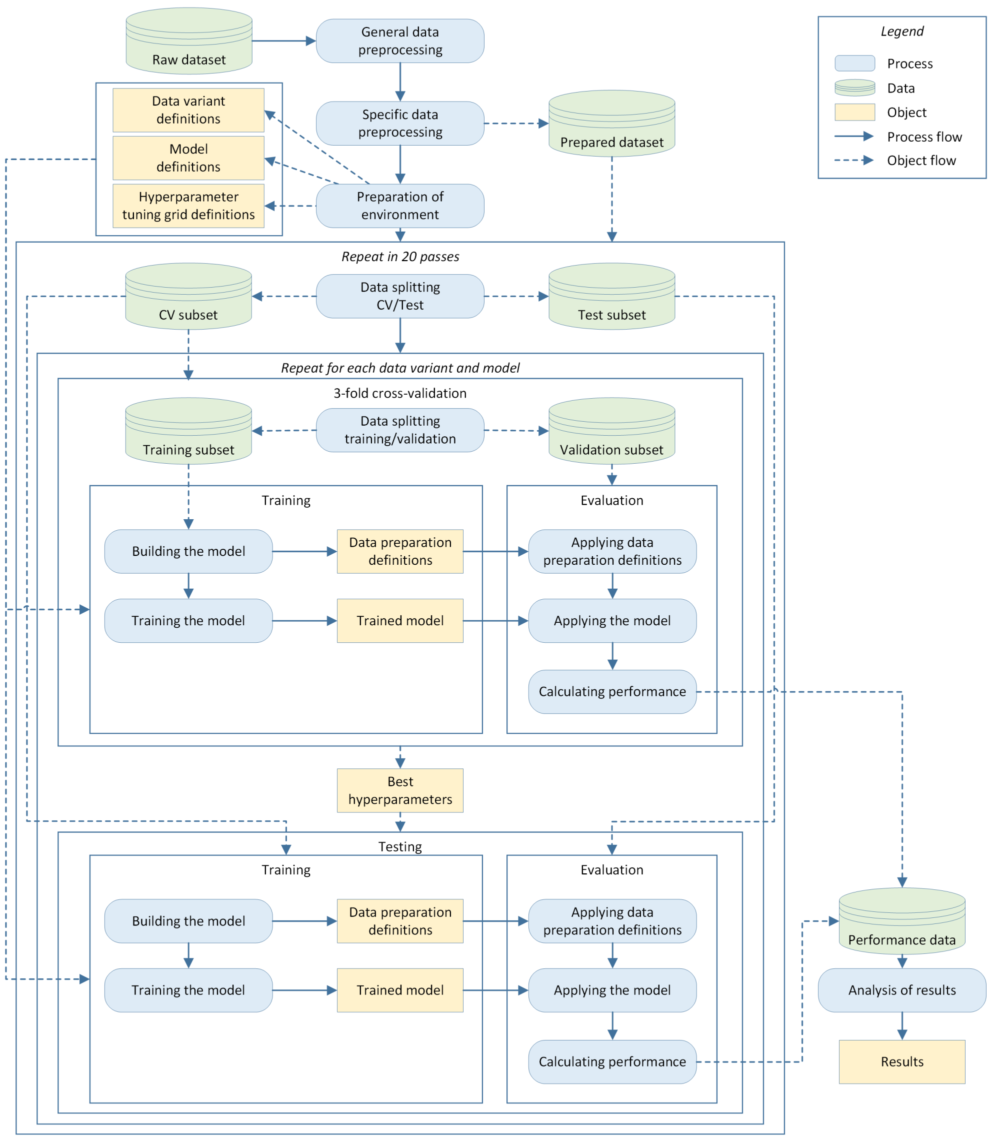


Figure 1. Overview of the research process.

The first important step of the data preprocessing specific to this study involved subsetting the dataset. There are no universally established attribute selection criteria involving the ISBSG dataset because they depend on the goals and methods used in the particular study. After investigating criteria used in other studies [7,14,17,19] the following inclusion criteria were applied:

- *Effort Test* > 0—the outcome value for prediction must be known;
- *Resource Level* = 1—only development team effort was included for effort-related attributes;
- *Count Approach* = 'IFPUG 4+'—projects in which size was estimated using IFPUG 4+ technique;
- *Adjusted Function Points* != NA—projects with no missing values for *Adjusted Function Points*, because it was not reasonable to predict effort without data on project size.

While including other categories for *Resource Level* and *Count Approach* would be an exciting path for the analysis, there were significantly fewer counts for these categories, and, thus, they were excluded. After this filtering, 1242 projects remained in the dataset.

The ISBSG dataset contains two essential attributes: *Data Quality Rating*, defined as "Project Rating. This field contains an ISBSG rating code of A, B, C or D applied to the project data by the ISBSG quality reviewers", and *UFP Rating*, defined as "Unadjusted Function

Point Rating. This field contains an ISBSG rating code applied to the Functional Size (Unadjusted Function Point count) data by the ISBSG quality reviewers", also expressed on a 4-point ordinal scale from A do D [8]. Category A denotes that nothing suspicious was found in the data, category D denotes little data credibility for a given project, and categories B and C denote intermediate states. Precise data evaluation and rating criteria with these categorisations are not publicly available. In addition, preliminary analyses of the dataset and earlier studies [20] led to the observation that, for *Data Quality Rating*, this classification mainly indicates data completeness for a given project, i.e., higher rating when there are fewer critical attributes with missing values. For *UFP Rating*, it mainly reflects data integrity between variables describing project size. In many studies, the dataset was filtered to include only projects with categories A or B for one or both rating attributes [7,10–15] or with even stricter filtering including only projects with *Data Quality Rating* in the A category [16,17]. Furthermore, the ISBSG guidelines [9] recommend using projects for statistical analyses with *Data Quality Rating* A or B, because of the uncertainty about some of the size or effort values for projects with ratings C or D. However, arbitrary data filtering, based on these ratings, may be too conservative or too optimistic [21]. The current study investigated the impact of data filtering, according to these categories, on predictions. Hence, no such upfront data filtering was performed.

At this stage, several attributes were further removed, e.g., those having a low fraction of valid values (<0.3), attributes with project sizing expressed in units other than *Adjusted Function Points*, attributes reflecting or based on the total effort for the project, effort breakdown attributes for phases following software testing, and attributes with unclear interpretation. Note that some actions were performed at the general and the specific preprocessing stages because attribute removal criteria were less strict at the former than at the latter subphase. Attributes unsuitable as predictors and unrelated to the testing effort were also removed because, usually, it is better to have fewer but better attributes [22].

There were several attributes of the multinominal type, which had multiple values encoded as a single value for some projects. For example, a project might have a *1st Language* defined as "C++; Java;" which means that both these languages were used in the project as the main ones. As the predictive models cannot operate directly on such multinominal values, all attributes of this type were converted to a set of logical attributes (e.g., *1st Language: C++*, *1st Language: Java*, etc.) with values 0/1.

2.3. Prepared ISBSG Dataset

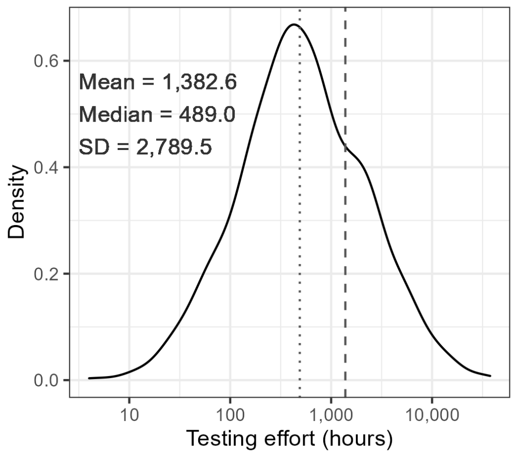
After performing two subphases of data preprocessing, the prepared ISBSG dataset contained 1242 projects described by 52 attributes, including 7 numeric, 9 nominal, 3 'natural' logical, and 28 logical predictors converted from 6 multinominal attributes and 5 additional non-predictor attributes (identifier, two rating attributes, *Year of Project* and the target for prediction).

Table 1 presents key summary statistics for the numeric attributes. The *Year of Project* reflects a year of project completion. The target predicted attribute is called *Effort test*. As mentioned earlier, the raw dataset also contains other attributes describing effort in a project, but this study used only those listed in this table. They reflect the effort on development phases earlier than testing because only these may serve as explanatory attributes for the *Effort test*. While the lowest value for *Effort test* is four hours, there were projects with zero effort on earlier phases. It should also be noted that the *Adjusted Function Points* is the only numeric predictor across the dataset with no missing values.

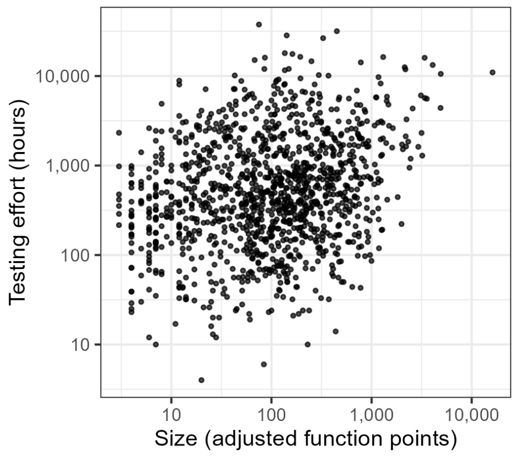
The main two attributes in this study are *Effort test*, as the outcome, and project size (*Adjusted Function Points*), as the primary predictor. Figure 2 illustrates a density distribution of testing effort and the relationship between these two attributes. We can observe high skewness of these attributes. Hence, later, when building predictive models, a $log_{10}()$ transformation was applied. While the relationship between project size and testing effort exists, it is not very strong, as other attributes influence the testing effort.

Table 1. Summary statistics for numeric variables.

Attribute	N	Mean	SD	Median	Min	Max	Skew
Year of Project	1242	2005.0	6.4	2005.0	1992	2019	0.0
Adjusted Function Points	1242	267.6	645.1	106.0	3	16148	13.7
Value Adjustment Factor	714	1.0	0.1	1.0	0.7	1.4	−0.3
Effort Plan [hours]	832	320.7	1042.8	94.5	0.0	17,668.0	11.1
Effort Specify [hours]	1072	602.7	1726.7	223.0	0.0	32,657.0	10.5
Effort Design [hours]	646	595.3	1139.3	225.5	0.0	10,759.0	4.6
Effort Build [hours]	1196	1820.9	3300.6	824.0	0.0	35,520.0	5.2
Effort Test [hours]	1242	1382.6	2789.5	489.0	4.0	37,615.0	6.0
Max Team Size [# people]	1073	26.9	37.0	12.0	0.5	468	3.7



(**a**)



(**b**)

Figure 2. (**a**) Density of testing effort. (**b**) Scatterplot of size vs. testing effort. (Note the logged scale for size and testing effort).

Table 2 presents summary statistics for nominal and logical attributes. The first two, *Data Quality Rating* and *UFP Rating*, were used to define the variants of input data (Section 2.5), and they were not predictors. All other attributes were used as predictors. Note that the usage of programming languages was reflected by the *Primary Programming Language* and the *1st Language*—the former describes a single language while the latter is a set of the main languages used.

2.4. Preparation of the Environment

Preparation of the environment for the use of predictive models involved definitions of data variants, predictive models, and hyperparameter tuning grids (Sections 2.5–2.7). It is important to highlight that, at this stage, only the definitions of these three elements were prepared (as functions in the script). However, they depended on the actual dataset used as the input to the tasks. This actual dataset was created later in the research process, i.e., in the repeated loop of cross-validation and testing. Therefore, these definitions were created during this preparation phase, but their executions (function calls in a script) occurred inside this loop.

2.5. Data Variants

Data variants define the scope of the dataset used to train predictive models and, thus, provide a data filtering strategy. Three dimensions were used to define data variants: *Data Quality Rating*, *UFP rating*, and the threshold for a fraction of valid values. The first two were provided as attributes in the raw dataset. Table 3 presents the number of projects per category of *Data Quality Rating* and *UFP rating*. These counts were used to determine data variants used to train predictive models. Specifically, there were very few projects with

UFP Rating C or D and slightly more for these categories of *Data Quality Rating*. Hence, there was no substantial justification for defining data variants for all combinations of these attributes. Instead, based on these counts, the following combinations of ratings and selection criteria for projects in the training subset were used:

- A-A: $Data\ Quality\ Rating \in \{A\} \wedge UFP\ Rating \in \{A\}$,
- A-B: $Data\ Quality\ Rating \in \{A\} \wedge UFP\ Rating \in \{A, B\}$,
- B-A: $Data\ Quality\ Rating \in \{A, B\} \wedge UFP\ Rating \in \{A\}$,
- B-B: $Data\ Quality\ Rating \in \{A, B\} \wedge UFP\ Rating \in \{A, B\}$,
- D-D: $Data\ Quality\ Rating \in \{A, B, C, D\} \wedge UFP\ Rating \in \{A, B, C, D\}$.

Table 2. Summary statistics for nominal attributes.

Attribute	N	Type [1]	# States	Mode
Data Quality Rating	1242	nominal	4	B
UFP Rating	1242	nominal	4	A
Industry Sector	972	nominal	6	Medical & Health Care
Application Group	452	nominal	4	Business Application
Development Type	1242	nominal	3	Enhancement
Development Platform	480	nominal	4	Mainframe
Language Type	628	nominal	4	3GL
Primary Programming Language	627	nominal	11	COBOL
Team Size Group	1073	nominal	14	5–8
How Methodology Acquired	381	nominal	2	Developed In-house
Architecture	422	nominal	3	Stand alone
Used Methodology	518	logical	2	yes
Upper CASE Used	385	logical	2	no
Metrics Program	533	logical	2	yes
Organisation Type	967	multinominal	7	Medical and Health Care
Application Type	473	multinominal	3	Transaction/Production System
Project Activity Scope	1242	multinominal	3	Specification
1st Language	636	multinominal	10	COBOL
1st Data Base System	389	multinominal	3	DB2
Functional Sizing Technique	572	multinominal	2	Manual supported by a tool

[1] After data preprocessing, each attribute denoted as 'multinominal' was already transformed to the set of logical attributes (as explained in Section 2.2). For brevity of the table, these attributes are presented here as single attributes as in the raw dataset.

Table 3. Counts for *Data Quality Rating* and *UFP Rating*.

Data Quality	UFP Rating				Sum
Rating	A	B	C	D	
A	288	52	0	0	340
B	317	489	7	0	813
C	23	8	1	0	32
D	42	12	0	3	57
Sum	670	561	8	3	1242

The third and last dimension of data variants was the arbitrarily defined vector of values in the range $[0.3, 0.5]$ and a step of 0.1. This value reflects a fraction of valid cases, i.e., non-missing, for each attribute in the training dataset. If the actual fraction for a given attribute was lower than a threshold value, then such an attribute was excluded from the training dataset. Thus, a high value of such a threshold caused fewer predictor attributes to be left in the training dataset. At the extreme, a value of 1 indicates that only those attributes with no missing values were left in the dataset. Figure 3 illustrates the number of predictor variables that met a given value of a threshold for a fraction of valid values for various data ratings. Note that the presented counts of predictor attributes refer to the whole dataset used after data preprocessing and not to the training subset. The predictor counts for the training dataset actually used to train the models varied, depending on the pass number and cross-validation repeat, because of random data subsetting. Thus, the values in the figure are only presented for reference. For some data ratings, there were no differences (A-A and A-B) in predictor counts for threshold values $[0.3, 0.5]$. There were very few differences between ratings A-A and A-B for higher threshold values. Across the range of the investigated thresholds, usually, the lowest predictor count was for data rating B-B. The highest predictor count was for data rating D-D, B-A, A-A, or A-B, depending on the threshold value.

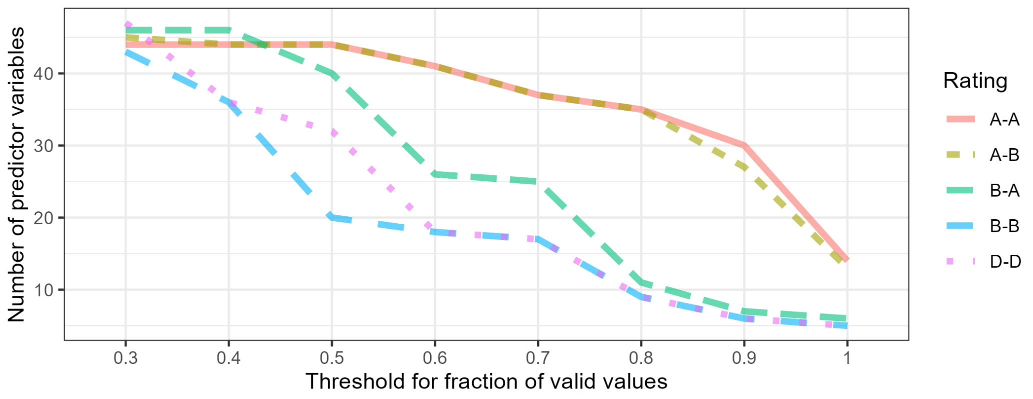


Figure 3. Counts of predictor variables for various threshold values of valid data fractions.

In total, 40 data variants were defined (5 data ratings × 8 thresholds). The following display format for complete data variants was used throughout the paper: "Q-U-T", where Q denotes the lowest acceptable *Data Quality Rating*, U denotes the lowest acceptable *UFP Rating*, and T denotes the threshold value of valid data fractions.

2.6. Predictive Models

A total of 16 predictive models were used. Table 4 lists them, together with the information on the key data handling procedures for each model. The first four models are not "real" predictive models but various baselines. These baseline models do not use any values of the predictors but provide predictions purely based on the values of the outcome variable in the training data subset. Obviously, we should not expect accurate predictions from such models. On the contrary, we should expect that "real" predictive models should not perform worse than these baselines. Thus, these baseline models were included in the study to enable better evaluation of the real predictive models. Without any baseline model, it would only be possible to compare particular models with each other. Including baseline models, enables assessing if particular models provide sensible predictions, i.e., if they are good or poor, not only amongst each other but also in an absolute sense. Some authors argued that these baseline models are too simplistic and proposed other deterministic models [23,24]. However, these proposed baselines include some logic to estimate the outcome attribute based on the values of predictors, so they are not as naive as the four baselines mentioned above. They are, in fact, "real" models, although still relatively simple, and they were not used in this study as baselines.

Selected predictive models were widely used in earlier studies on software effort prediction, and they are popular in such applications (e.g., [7,25–27]). The focus of this paper is different from how these models operate internally. Therefore, this paper does not provide their detailed definitions. The model acronyms listed in the first column of Table 4 are the names of the model implementations in the caret package [28] for the R language [29], which were used for model training and delivering predictions.

Most models, excluding some tree-based ones, required numeric data with all nominal and logical attributes encoded as numeric attributes with values 0 or 1. When an outcome attribute is heavily skewed, several models usually perform better with outcome and/or predictor attributes transformed by a function that brings values closer to a normal distribution, and the relationship between the outcome and predictor is closer to linear. Thus, for some models, log-based transformations were applied, similarly, for example, to [26,30]. It is impossible to train most models when predictor attributes contain missing values. Therefore, for most models, data transformation included imputing these missing values by a mode or median, depending on the attribute type. In this study, we wanted to use good and fast methods for data preparation. While there are potentially better imputation methods, they are usually time-consuming. Besides, earlier experiments with more complex imputation methods did not confirm their clear advantages [31,32]. For most models, the normalisation of numeric attributes was performed. For some models, an attribute synthesis, using principal component analysis (PCA), was also applied to reduce the number of predictors and associations between predictors. The choice of the data handling procedures for particular models was motivated by common knowledge, earlier experiments with these models [18,33], and preliminary unpublished experiments on the subsets of data. While this study attempted to deliver sensible models, it did not attempt to identify the optimal variant for each model, because that would require significant additional calculation time and was beyond the scope of this study.

Table 4. Used predictive models.

Model	**Description**	**Data Type**	**Log Outcome**	**Log Predictors** [1]	**Impute by Mode** [2]	**Impute by Median** [3]	**Normalize**	**PCA** [4]
bMean	baseline model predicting $mean(Y_{train})$	regular	+	−	−	−	−	−
bMed	baseline model predicting $median(Y_{train})$	regular	+	−	−	−	−	−
bR	baseline model predicting a random value from Y_{train}	regular	−	−	−	−	−	−
bRU	baseline model predicting a random value from $Uniform(min(Y_{train}), max(Y_{train}))$	regular	+	−	−	−	−	−
enet	elastic net [34]	numeric	+	+	+	+	+	−
gbm	generalized boosted regression [35]	numeric	−	+	+	+	+	+
glmnet	generalized linear regression with convex penalties [36]	numeric	−	+	+	+	+	−
knn	*k*-nearest neighbour regression [37]	numeric	−	+	+	+	+	+
lm	linear regression [38]	numeric	+	+	+	+	+	+
lmStepAIC	linear regression model with stepwise feature selection [39]	numeric	+	+	+	+	+	−
M5	model trees and rule learner [40,41]	numeric	−	+	+	+	+	+
nnet	neural network with one hidden layer [42]	numeric	+	+	+	+	+	−
ranger	random forest [43]	regular	−	+	+	+	−	−
rpart2	recursive partitioning and regression tree [44]	regular	−	+	+	+	−	−
svm	support vector machines [45]	numeric	+	+	+	+	+	+
xgbTree	extreme gradient boosting [46]	numeric	−	+	−	−	−	−

[1] Denotes if numeric predictors with skewed distributions were transformed using a $log_{10}(x)$ or $log_{10}(x+1)$ function, the latter was used if, for a particular predictor, its lowest value was 0. [2] Denotes if missing values in nominal or logical predictors were filled with the mode value of each attribute. [3] Denotes if missing values in numeric predictors were filled with the median value of each attribute. [4] Denotes if numeric predictors were converted into one or more principal components to capture at least 85% variability in the attributes.

2.7. Hyperparameter Tuning Grids

Some models require the provision of parameters that drive the model training process. Since they cannot be learnt from the data, they are usually denoted as hyperparameters and provided explicitly. There are various strategies for choosing the best possible values of these hyperparameters, like grid search, random selection, simulated annealing, and other methods [47–49].

This study involved the random grid selection strategy using the CV loop for each model that required hyperparameters. All possible and sensible combinations of potential hyperparameter values were initially defined in a grid, where a row represented a unique set of values for all considered hyperparameters. Evaluation of all such combinations, like in a grid search, could be time-consuming for some models. In this study, the global optimisation of hyperparameters was not the main problem investigated. The hyperparameter selection was performed on the subset of created grids to keep the calculation times acceptably short. Specifically, if the tuning grid for a given model was large, a subset of 100 randomly selected rows was considered. For models using a large number of hyperparameters, like ranger, xgbT, and svm, such a filtered tuning grid represented only a small fraction of the entire grid. Thus, most likely, it did not contain a globally optimum set of hyperparameters. However, preliminary experiments on small subsets of data and earlier analyses [18,33] revealed that using more rows from the original grid improved the accuracy of predictions only to a small degree.

Some models require values for some hyperparameters that depend on the size of the dataset, which is actually used to train the model. For example, for the ranger model, the 'number of variables to possibly split at in each node' (*mtry*) depends on the number of predictors in the dataset. Thus, a vector of 10 values in the range $[floor(\#predictors/2),\#predictors]$ was generated. Due to such dependencies at the preparation of the environment stage, only the functions generating the tuning grids were defined, but the actual creation of tuning grids was performed inside the CV loop, where the dataset size for a particular model was already known.

2.8. Model Training and Evaluation

Model training and evaluation was performed by random data splitting into cross-validation (CV) and test subsets with project counts of 992 and 250, respectively. Performing such a data split just once would have led to bias in the results. This random data split was repeated 20 times to avoid bias, as was common in similar studies, e.g., [50]. It meant that subsequent steps were repeated but with different CV and test subsets. The term *pass* used throughout the paper denotes an iteration of steps performed on a single such data split.

Each data variant and each model was built, trained, and evaluated by performing CV and independent testing. The CV subset was used in the CV subprocess to determine the best set of values for model hyperparameters. A 3-fold CV was performed. It meant that the CV subset was split into three folds. In each iteration of the CV step, two of these folds were used to train the model, and the remaining one to evaluate it. Thus, in each iteration of the CV, a different fold was used to evaluate the model.

Building the model involved applying definitions of data variants, predictive models, and hyperparameter tuning grids to the training subset. Thus, the training subset was filtered according to the definition of the data variant. Next, the model was built based on the provided definitions and filtered dataset. Finally, the hyperparameter tuning grid was built. The detailed steps of model building in a single iteration were the following:

- filter training data by *Data Quality Rating* and *UFP Rating*, according to the current data variant;
- remove attributes not meeting the threshold according to the current data variant;
- apply missing value imputation by median (according to the current model definition);
- apply missing value imputation by mode (according to the current model definition);
- group together least frequent states (i.e., states with frequency < 0.03 of all cases) of nominal predictors as state 'other';
- convert nominal attributes to dummy attributes (if the current model requires numeric data);
- remove attributes with zero variance;
- log-transform outcome attribute (according to the current model definition);
- log-transform numeric predictors (according to the current model definition);
- normalize numeric attributes (according to the current model definition);
- apply PCA transformation (according to the current model definition);
- prepare the model;

- save data preparation definitions;
- generate hyperparameter grid.

After performing the above steps, the model was trained on such prepared data. The validation dataset was transformed using previously saved data preparation definitions (It is important to note that when transforming the validation subset, these data preparation definitions were only applied as they were learnt from the training subset. For example, data normalisation involves calculating the mean and standard deviation for the numeric attributes in the training subset. Then, these statistical measures were applied to the attributes in the validation subset, and they were not learnt (calculated) again on the validation subset). Such a transformed validation dataset was used as input to the trained model to get the predictions for testing effort. These predictions were compared with the actual values of the testing effort. To evaluate the accuracy of predictions, this study used a measure of the absolute error (AE) of prediction for each predicted case (Equation (1)), also called absolute residual (AR), and the mean absolute error to summarise the accuracy of prediction for a group of n projects (Equation (2)):

$$AE_i = |actual_i - predicted_i|, \tag{1}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} AE_i. \tag{2}$$

The main advantages of these measures are the simplicity of their calculation and interpretation, as well as the lack of bias, which are related to the use of various measures based on a relative error [50,51]. The optimum set of hyperparameters was selected for the lowest mean MAE across all three iterations of CV.

The next stage was to evaluate the model on the unseen test data. Hence, the process of building the model and the final training data preparation steps were performed, again similarly as to that inside the CV loop. However, this time it was performed on the entire training subset, not on folds, as during CV. Training the model was performed using the set of hyperparameters that was determined during CV as the optimum. Then, the trained model was evaluated on the test data. Calculated MAE on the test data served as the final indicator of model accuracy in each of the 20 passes and for each data variant.

2.9. Analysis of Results

The results were analysed using the dataset with summaries of predictions where each summary consisted of performance measures from a single iteration of model evaluation on the test subset. RP1, RP1A, RP2, and RP3 were investigated using a 'win–tie–loss' procedure [23,26,50,52,53]. For each inner iteration of the experiment, i.e., for each pass, model and data variant, the distribution of absolute prediction errors, based on the test subset, was produced. For RP1, this distribution, provided by each model, was compared to the performance of all other models. This comparison was performed using the Wilcoxon signed-rank test. As the same test was used several times, a Benjamini—Hochberg adjustment procedure of the p-values [54] for the Wilcoxon test was used to control false discoveries. It was performed separately for each data variant in each pass. If the Wilcoxon test did not reveal statistically significant differences in the distribution of AE, then the number of ties for both compared models was incremented. Otherwise, if this test revealed such a statistically significant difference, then, based on the median AEs, the number of wins or losses was incremented, respectively, for each compared model. This procedure was repeated for each model, each data variant, and each pass. Thus, given $M = 16$ models, a sum of wins, ties, and losses for each model was $(M-1) \times 40$ *data variants* $\times$ 20 *passes* = 12,000.

The results of these calculations were also applied to the RP1A. However, the focus here was on comparing performance between passes. Thus, the results from all passes were not summed up together, but, instead, grouped by each pass.

This procedure was also applied to the RP2, where particular data variants were compared with each other. These comparisons were performed individually for each model. The motivation for this was the hypothesis that particular data variants' usefulness might differ for particular predictive models. Hence, it was more beneficial not just to identify the overall best data variant but rather the best for each model. Thus, given $D = 40$ data variants, a sum of wins, ties, and losses for each data variant and each model was $(D - 1) \times 20\ passes = 780$.

Investigation of the RP3 also involved using this procedure. Thus, combinations of models and data variants were created and compared with each other in each pass. Hence, given $C = 640$ combinations ($16\ models \times 40\ data\ variants = 640$), a sum of wins, ties, and losses for each combination was $(C - 1) \times 20\ passes$ = 12,780.

Upon calculating counts of wins, ties, and losses, an aggregating measure of the *difference* was calculated as follows: $difference = wins - losses$. It served as the primary measure for investigating these RPs, i.e., for the RP1 to create a ranking of predictive models, for the RP1A to compare model stability across passes, for the RP2 to create a ranking of data variants, and for the RP3 to create a ranking of the best combinations of models and data variants.

Investigation of the RP2A was the only one where the 'win–tie–loss' procedure was not applied. Instead, it involved comparing the values of MAE calculated on CV and test subsets. Specifically, for each model in each pass, there were 40 values of MAE on the CV subset and 40 values of MAE on the test subset, where a single value reflected model performance with a particular data variant. To analyse the relationship between these groups of MAE across data variants, Spearman's rank correlation coefficient ρ was calculated. A value close to 1 indicated a positive correlation, i.e., that the rank of data variants on the CV data was similar to the test data. Such values were desirable as they revealed the performance stability between CV and test subsets. A value of zero indicated no such relationship. A value close to -1 indicated that the ranks of the data variants on the CV subset were the opposite, as in the test subset. Observing such values would indicate that using separate CV and test data splits did not allow for determining the best-performing data variants, because such independent splits provided inconsistent results. Spearman's ρ calculation was repeated for each model and pass.

3. Results

3.1. RP1: The Performance of Particular Models for Testing Effort Prediction

Table 5 contains the ranking of predictive models with the models sorted in a decreasing order of the *difference*, i.e., with the best-performing models at the top. The best-performing models were svm and enet, followed by nnet and lm. As expected, the worst-performing models were two baselines involving random prediction: bR and bRU. However, two other analysed baselines performed better than two real models: rpart2 and glmnet.

It should be noted that this overall ranking of models was based on predictions involving all data variants for each model. It may only serve as a rough overview of the models. Specific models may be sensitive to particular data variants and perform well on some data variants while doing so poorly on others. Thus, it is necessary to investigate model performance for particular data variants.

Figure 4 illustrates each model's performance for one major grouping of data variants, i.e., based on *Data Quality Rating* and *UFP Rating* combinations. We can observe that, for the most restrictive ratings A-A and A-B, the top-3 performing models were the following: svm, enet, and lm. For rating B-A, the top-3 performing models were the following: enet, lm, and nnet. For ratings B-B and D-D, the top-3 performing models were the following: nnet, svm, and enet. The top-3 models from the overall ranking (Table 5) performed no worse than the top-5 for particular rating combinations.

For ratings A-A and A-B, the order of models was almost the same. Rating B-A brought some changes in the order of the models. For ratings B-B and D-D, the order of models was almost identical, but different from rating B-A.

Table 5. Summary of model performance.

Model	Wins	Ties	Losses	Difference
svm	7274	4252	474	6800
enet	6842	4765	393	6449
nnet	6479	4594	927	5552
lm	6131	4775	1094	5037
xgbTree	5452	5183	1365	4087
knn	5164	5511	1325	3839
lmStepAIC	5343	4867	1790	3553
ranger	4469	5045	2486	1983
gbm	4186	4353	3461	725
M5	3682	3960	4358	−676
bMean	3133	2736	6131	−2998
bMed	2972	2735	6293	−3321
rpart2	2510	3275	6215	−3705
glmnet	1512	3327	7161	−5649
bR	979	336	10,685	−9706
bRU	7	16	11,977	−11,970

We can also observe that for ratings A-A, A-B, and B-A, the values of the mean differences for almost all models, except the two worst baselines, spanned almost equally. However, for ratings B-B, and even clearly for D-D, we can observe groups of models with a similar value of the *difference*. Specifically, this included the group of the best-performing models, including nnet, svm, enet, lm, xgbTree, and lmStepAIC.

No significant shifts of model ranks were observed, where a model performing well for one combination of ratings would perform poorly for a different combination relative to the other models. However, a noticeable increase of the mean *difference* was observed for lmStepAIC and xgbTree comparing ratings A-A and A-B with ratings B-B and D-D. A substantial decrease between these ratings was observed for glmnet.

Figure 5 illustrates each model's performance for another major grouping of data variants, i.e., based on the threshold for fractions of valid values. For thresholds in the range $[0.3, 0.6]$, the top-4 models with very similar mean *differences* were the following: enet, svm, nnet, and lm. These four models were also the best for threshold 0.7 but all with lower mean *difference*. For thresholds 0.8–1.0, consistently, the best models were svm and enet. The third place was reached by the models lm, xgbTree, and nnet, respectively, for thresholds 0.8, 0.9, and 1.0.

For thresholds in the range $[0.3, 0.6]$, the mean *difference* for all models was very stable at almost the same level. There were also very few tiny differences in the order of particular models. However, starting with the threshold 0.7, as this value increased the range of the mean *difference* decreased for almost all models except the worst two baselines. It meant that the best models performed relatively more poorly, and the worst models performed relatively better. The most noticeable shift of the mean *difference* was observed for the rpart2 model for thresholds in the range $[0.6, 0.8]$.

For threshold 0.9, and even more evidently for 1.0, two baselines, i.e., bMedian and bMean, reached a mean *difference* higher than several real models. Specifically, for threshold 1.0, these two baselines reached the fifth and sixth places, respectively, and, thus, beat eight real models.

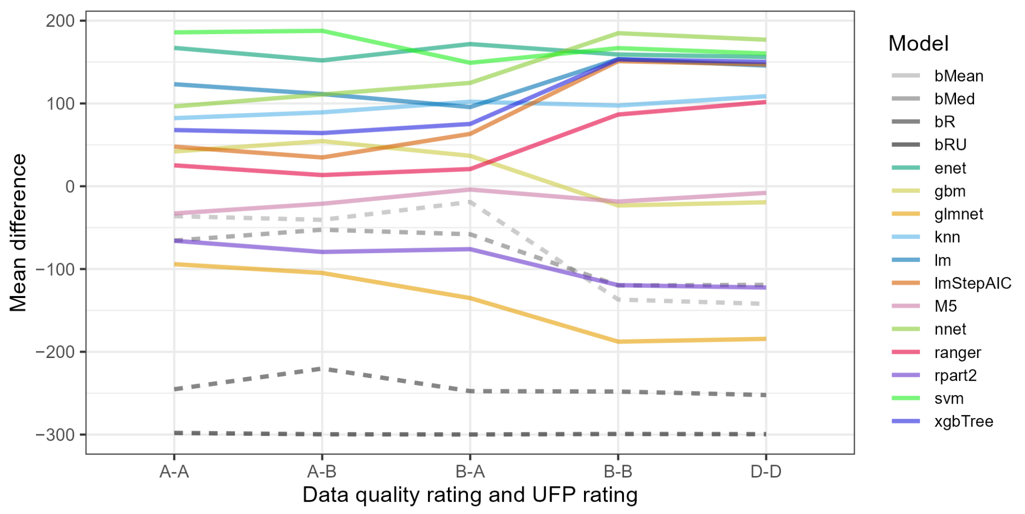


Figure 4. Mean *difference* of models for various data quality and UFP ratings. Each data point denotes a mean *difference* of a particular model for a given combination of *Data Quality Rating* and *UFP Rating* across all values of thresholds for fractions of valid values in all passes.

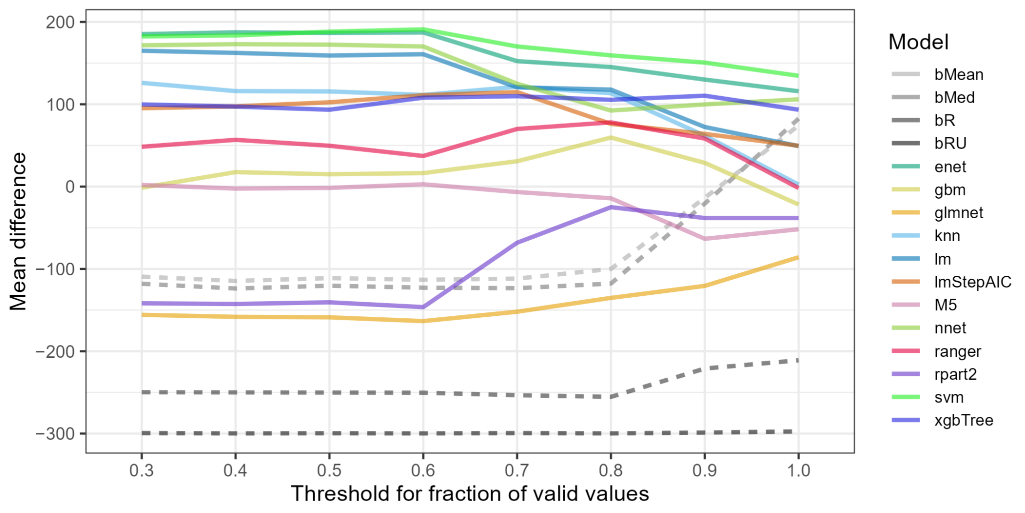


Figure 5. Mean *difference* of models for various thresholds for fractions of valid values. Each data point denotes a mean *difference* of a particular model for a given threshold for fractions of valid values across all investigated combinations of *Data Quality Rating* and *UFP Rating* in all passes.

Table 6 ranks models for each data variant. The order of columns reflects the order of models, as in the ranking in Table 5. For all data variants from A-A-0.3 to A-B-0.8 and four other data variants, the best-performing model was svm. For most B-A-* data variants, except B-A-1.0, the best model was enet. For data variants B-B-*, most often, the best model was nnet, followed by svm and enet. Similar results were for data variants D-D-*, except for data variant D-D-0.7, where the best model was xgbTree. Although the lm model did not have the highest *difference* for any data variant, on average, it reached the fourth place, one better than xgbTree that was the best for one data variant. Four models, i.e., svm, enet, nnet and xgbTree, performed the best for at least one data variant. The worst-performing models for all data variants were consistently bRU and bR, with exceptions for B-B-0.9 and B-B-1.0, where glmnet performed worse than bR.

The lowest value of the *difference* sufficient for a model to be the best was usually for data variants with ratings A-A, A-B, or B-A, and thresholds of valid values of 1.0 or 0.9. In particular, for data variant B-A-1.0 the $difference = 84$ was sufficient for svm to beat other models. Only three models, M5, bR and bRU, had a negative value of the *difference*.

Such a result showed that, for this data variant, the differences in performance between the models were the lowest. On the other hand, for data variant A-A-0.6 the enet model reached a $difference = 213$, which was sufficient only for the second place, because svm reached a $difference = 218$. The highest ranges between the best- and least-performing models were for data variants: B-B-0.9, D-D-0.9, A-A-0.3, and A-B-0.4.

Table 6. The values of the *difference* of each model grouped by each data variant. The models in particular columns were sorted from the best to the worst, according to their mean *difference* across all data variants provided in the last row. The data variants in particular rows were sorted in alphabetical order. Values with a single underline denote the best model for a given data variant, and values with a double underline denote the worst model for a given data variant.

Data								Model [1]							
Variant	**svm**	**enet**	**nnet**	**lm**	**xgbT**	**knn**	**lmS**	**Ranger**	**gbm**	**M5**	**bMean**	**bMed**	**rpart2**	**glmnet**	**bR**	**bRU**
A-A-0.3	231	206	184	202	62	119	58	−29	3	16	−88	−114	−166	−134	−251	−299
A-A-0.4	222	208	188	209	49	86	63	−1	44	17	−97	−126	−170	−142	−251	−299
A-A-0.5	223	209	187	212	42	85	64	−8	41	18	−90	−122	−171	−140	−251	−299
A-A-0.6	218	213	191	206	49	94	68	−13	27	2	−85	−115	−166	−141	−249	−299
A-A-0.7	182	144	50	81	75	118	72	58	83	−51	−64	−108	33	−121	−254	−298
A-A-0.8	166	118	1	80	110	126	2	91	96	−49	−58	−98	69	−97	−258	−299
A-A-0.9	120	118	−39	−1	102	22	35	52	40	−147	99	84	29	6	−224	−296
A-A-1.0	125	121	9	−3	54	8	21	52	2	−69	94	74	15	16	−224	−295
A-B-0.3	215	164	180	190	48	136	53	−12	36	0	−77	−90	−174	−137	−234	−298
A-B-0.4	229	168	178	189	25	134	45	−7	82	3	−89	−104	−177	−141	−235	−300
A-B-0.5	227	169	178	189	18	138	47	−9	83	2	−88	−103	−177	−139	−235	−300
A-B-0.6	220	186	163	172	76	124	68	−21	58	11	−87	−100	−178	−155	−237	−300
A-B-0.7	170	113	80	98	69	110	99	72	72	−28	−97	−109	21	−124	−247	−299
A-B-0.8	177	140	56	90	62	103	12	54	93	−22	−76	−95	56	−99	−251	−300
A-B-0.9	125	139	6	−16	101	−7	−23	29	25	−61	92	90	−11	−22	−167	−300
A-B-1.0	139	137	46	−21	115	−24	−23	2	−13	−74	97	91	5	−21	−156	−300
B-A-0.3	168	209	141	113	76	130	54	35	11	4	−33	−68	−137	−153	−250	−300
B-A-0.4	167	211	142	113	77	130	52	34	18	3	−35	−68	−139	−155	−250	−300
B-A-0.5	164	207	185	78	73	127	65	−1	13	19	−29	−60	−129	−166	−247	−299
B-A-0.6	187	197	146	103	89	121	101	−4	35	14	−42	−77	−147	−177	−246	−300
B-A-0.7	184	191	170	111	80	120	106	0	5	24	−47	−80	−146	−170	−248	−300
B-A-0.8	132	169	102	121	53	72	68	19	78	−40	−13	−76	0	−135	−250	−300
B-A-0.9	107	121	64	94	86	90	41	69	87	−40	−14	−82	59	−127	−255	-300
B-A-1.0	84	69	49	32	68	27	19	14	47	−16	61	47	31	2	−234	−300
B-B-0.3	144	180	183	158	162	125	158	113	−37	−8	−173	−159	−112	−180	−254	−300
B-B-0.4	156	172	173	156	160	123	160	124	−30	−11	−176	−161	−112	−180	−254	−300
B-B-0.5	160	177	176	159	174	86	171	127	−26	−24	−175	−162	−110	−175	−258	−300
B-B-0.6	171	176	178	167	173	94	167	100	−27	−14	−176	−163	−116	−170	−260	−300
B-B-0.7	164	160	166	163	162	126	150	99	5	−2	−176	−163	−123	−173	−258	−300
B-B-0.8	166	152	156	148	151	135	154	107	22	1	−176	−161	−123	−174	−258	−300
B-B-0.9	201	141	234	157	131	92	143	58	−19	−46	−107	−92	−134	−233	−228	−298
B-B-1.0	173	115	213	121	113	−1	106	v35	−74	−44	64	102	−126	−217	−214	−296
D-D-0.3	154	167	170	162	151	120	153	135	−20	−2	−176	−159	−120	−175	−260	−300
D-D-0.4	144	178	184	145	175	107	167	134	−26	−24	−176	−160	−115	−173	−260	−300
D-D-0.5	167	171	136	158	160	142	165	139	−36	−23	−174	−155	−116	−174	−260	−300
D-D-0.6	159	164	173	156	153	124	151	124	−11	1	−176	−159	−125	−174	−260	−300
D-D-0.7	151	154	159	150	163	132	147	121	−11	24	−175	−157	−126	−172	−260	−300
D-D-0.8	156	147	147	150	151	131	145	119	9	39	−177	−159	−127	−171	−260	−300
D-D-0.9	200	131	234	128	132	112	124	84	11	−23	−138	−103	−134	−227	−231	−300
D-D-1.0	152	137	213	117	117	2	125	−42	−71	−56	55	99	−116	−209	−227	−296
Mean	170	161	139	126	102	96	89	50	18	−17	−75	−83	−93	−141	−243	−299

[1] lmS = lmStepAIC, xgbT = xgbTree.

3.2. RP1A: The Stability of Performance of Models across Passes

Figure 6 presents the values of the *difference* for each model in each pass. The lowest variability of the *difference* across passes were for bRU ($\sigma = 2.6$) and bR ($\sigma = 18.7$), and, among non-baseline models, for enet ($\sigma = 27.0$), knn ($\sigma = 28.2$) and glmnet ($\sigma = 28.3$). The highest variability of the *differences* across passes were for bMean ($\sigma = 49.0$), lmStepAIC ($\sigma = 45.3$), and ranger ($\sigma = 43.1$). The lowest range of the *differences* across passes were for bRU (10) and bRU (68), and, among non-baseline models, for M5 (104), glmnet (112), and enet (112). The lowest range of the *difference* across passes was for bMean (168), ranger (151), and lmStepAIC (147).

A noticeable variability of the *difference* for most models across passes caused changes between ranks of particular models. However, there were no significantly deviating values of the *difference* in particular passes that would reflect exceptionally good or poor model performance in those passes. Overall, these results demonstrate high stability of model performance across passes.

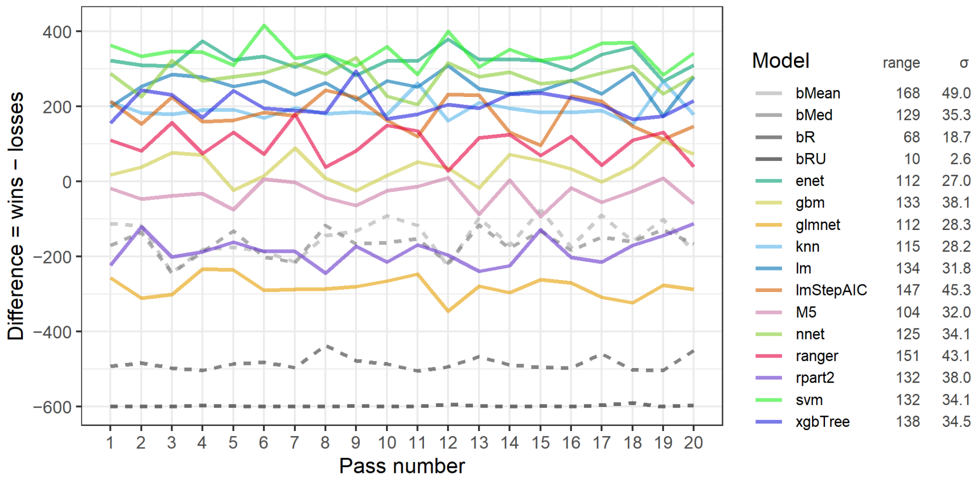


Figure 6. Stability of the *difference* for models across passes. Each data point denotes a *difference* for a particular model in a given pass across all investigated data variants.

3.3. RP2: Identification of Good and Poor Data Variants for Particular Models

Table 7 provides a ranking of data variants for each model (Please note that, although Tables 6 and 7 have a similar appearance of rows and columns, the presented values of *difference* have diverse meanings. Table 6 provides values of the *difference* for particular models grouped by each data variant. In contrast, Table 7 provides values of the *difference* for particular data variants grouped by each model). On average, i.e., across all models, the best-performing data variants were based on ratings D-D-* followed by B-B-* for various thresholds of valid values in the range $[0.3, 0.8]$. Specifically, for models enet, knn, lm, and ranger, the best data variant was D-D-0.3, for xgbTree it was D-D-0.4, for M5 and svm it was D-D-0.8, for nnet it was B-B-0.3, for gbm it was B-B-0.8, and for lmStepAIC it was B-B-0.5. For models performing poorly, like glmnet and rpart2, other data variants appeared to be the best: B-A-0.3 tied with B-A-0.4, and B-A-0.9, respectively. Data variants D-D-0.7, D-D-0.6, and D-D-0.5 were not the best for any model, yet they were overall ranked at places second, third, and fifth, respectively. The data variant with rating A-A-* was not the best for any model.

On average, across models, the worst-performing data variants were B-B-1.0, D-D-1.0, A-B-1.0, A-B-0.9, A-A-1.0, and A-A-0.9. These data variants were restrictive regarding reducing the number of rows or attributes in the training dataset, or both.

Among the baseline models, the best data variants were A-A-* for bR and bRU, and B-A-* for bMean and bMedian. Specifically, with more restrictive data variants, the baselines involving random predictions (bR and bRU) performed better but, as investigated earlier, were still the worst compared to other models.

Table 7. The values of the *difference* of each data variant grouped by each model. The data variants in particular rows were sorted from the best to the worst, according to their mean *difference* across all models provided in the last column. The models in particular columns were sorted in alphabetical order. Values with a single underline denote the best data variant for a given model, and values with a double underline denote the worst data variant for a given model.

Data	Model [1]																Mean
Variant	bMean	bMed	bR	bRU	enet	gbm	glmnet	knn	lm	lmS	M5	nnet	Ranger	rpart2	svm	xgbT	
D-D-0.3	−72	−48	−136	−464	501	277	109	371	468	530	310	410	584	269	317	521	247
D-D-0.7	−72	-48	−136	−464	382	299	158	354	412	500	353	376	525	256	346	538	236
D-D-0.6	−72	−48	−136	−464	401	311	152	342	456	476	326	339	544	257	340	468	231
D-D-0.4	−72	−48	−136	−464	456	198	125	286	367	528	284	438	564	269	294	567	228
D-D-0.5	−72	−48	−136	−464	443	194	134	312	425	532	271	330	565	269	313	512	224
D-D-0.8	−72	−48	−136	−464	378	332	156	259	390	489	386	280	512	256	378	490	224
B-B-0.7	−176	−120	−96	−464	326	323	105	308	342	514	319	323	461	271	376	491	206
B-B-0.3	−176	−120	−96	−464	450	175	−38	290	357	511	255	447	504	274	303	511	199
B-B-0.4	−176	−120	−96	−464	436	188	−51	283	341	524	269	421	505	274	271	493	194
B-B-0.8	−176	−120	−96	−464	283	341	77	235	319	475	320	303	459	271	341	453	189
B-B-0.6	−176	−120	−96	−464	321	218	75	273	344	489	268	329	447	270	293	484	185
B-B-0.5	−176	−120	−96	−464	384	193	5	207	331	534	226	338	491	274	314	498	184
B-A-0.7	256	192	64	176	163	65	245	188	76	208	241	177	−48	−1	176	67	140
B-A-0.6	256	192	64	176	154	133	251	200	65	184	205	150	−57	−1	172	39	136
A-A-0.6	8	−16	88	496	265	98	251	217	331	15	133	175	−119	−207	261	−95	119
A-A-0.4	8	−16	88	496	245	153	246	177	319	−39	170	156	−76	−207	266	−119	117
A-A-0.5	8	−16	88	496	245	141	246	177	319	−39	150	153	−97	−207	266	−115	113
A-A-0.3	8	−16	88	496	233	96	250	256	323	−41	137	159	−143	−207	251	−106	112
B-A-0.5	256	192	64	176	137	90	249	177	−95	42	154	178	−87	−1	37	−8	98
B-A-0.4	256	192	64	176	137	45	290	210	−27	−28	146	92	−29	−4	15	−18	95
B-A-0.3	256	192	64	176	140	34	290	202	−34	−41	147	86	−7	−4	−3	−17	93
A-B-0.6	−16	−8	80	256	69	114	128	215	204	−34	152	160	−159	−215	244	−117	67
A-B-0.4	−16	−8	80	256	51	163	128	194	219	−71	76	196	−116	−215	200	−197	59
A-B-0.5	−16	−8	80	256	51	163	128	192	219	−71	76	196	−129	−215	187	−199	57
A-B-0.3	−16	−8	80	256	25	32	128	221	219	−69	49	149	−181	−215	174	−146	44
B-A-0.8	256	192	64	176	−92	138	183	−93	−47	−52	−38	−169	−11	267	−155	−152	29
B-A-0.9	256	192	64	176	−315	96	207	−211	−335	−320	−98	−424	25	418	−415	−165	−53
A-A-0.7	8	−16	88	496	−307	49	136	−201	−334	−312	−200	−381	−84	140	−207	−234	−85
A-B-0.7	−16	−8	80	256	−346	4	−75	−181	−358	−232	−165	−321	−84	216	−283	−262	−111
A-A-0.8	8	−16	88	496	−420	20	102	−309	−464	−460	−276	−521	−101	240	−315	−217	−134
D-D-0.9	−72	−48	−136	−464	−289	−103	−571	−99	−140	−50	−87	68	−101	−98	−63	15	−140
A-B-0.8	−16	−8	80	256	−408	24	−82	−313	−433	−417	−171	−443	−85	234	−349	−286	−151
B-B-0.9	−176	−120	−96	−464	−311	−230	−621	−141	−106	−115	−127	45	−138	−188	−95	−34	−182
B-A-1.0	256	192	64	176	−423	−473	−154	−638	−585	−555	−568	−568	−513	−158	−475	−465	−305
A-A-0.9	8	−16	88	496	−619	−551	−294	−641	−680	−620	−653	−712	−583	−225	−610	−543	−385
A-A-1.0	8	−16	88	496	−618	−659	−287	−654	−681	−600	−603	−645	−560	−352	−635	−548	−392
A-B-0.9	−16	−8	80	256	−638	−609	−464	−671	−691	−625	−593	−646	−625	−415	−623	−510	−425
A-B-1.0	−16	−8	80	256	−631	−677	−465	−679	−684	−660	−635	−567	−678	−413	−658	−464	−431
D-D-1.0	−72	−48	−136	−464	−632	−708	−711	−661	−567	−542	−617	−537	−698	−599	−631	−553	−511
B-B-1.0	−176	−120	−96	−464	−627	−697	−741	−654	−585	−558	−592	−540	−677	−578	−618	−577	−519

[1] lmS = lmStepAIC, xgbT = xgbTree.

3.4. *RP2A: The Stability of Performance of Data Variants between CV and Test Data Subsets*

Figure 7 illustrates the values of Spearman's ρ for MAE between CV and test subsets across data variants for each model and pass. Most data points overlap each other for high values of Spearman's ρ, especially in the range $[0.8, 1]$. It is desirable to have such results, as they confirm the stability of model performance across data variants between CV and test subsets. The focus here was not on these high values, but rather on values outside this range, indicating potential problems.

On average, among non-baseline models, the stability of model performance was very high. The mean and median values of ρ across all passes were very high, both exceeding 0.9 for models such as enet, gbm, glmnet, knn, lm, nnet, ranger, and xgbTree. For lmStepAIC and svm, the median ρ was around 0.9, but the mean was slightly lower. Only for M5 and rpart2, were both the mean and median ρ lower than 0.9 but still high (>0.75). In only single passes, the ρ for some models was noticeably lower, such as for the following: for svm in pass 5 ($\rho = 0.20$), for M5 in pass 19 ($\rho = 0.53$), for glmnet in pass 9 (ρ = 0.57), for lmStepAIC in pass 9 ($\rho = 0.59$). In six other cases, the value of ρ was in the range $[0.70, 0.75]$. In 15 cases it was in the range $[0.75, 0.80]$ and in all other cases, it was higher than 0.75.

Among non-baseline models, there were four cases with the negative ρ: bMedian in pass 17 ($\rho = -0.97$), bMean in pass 6 ($\rho = -0.60$), bMean in pass 17 ($\rho = -0.60$), and bMean in pass 12 ($\rho = -0.50$). These results suggested substantial inconsistency between prediction accuracy on CV and test subsets in these passes. However, for bMean and bMedian the mean ρ was still relatively high, and the median ρ was very high. Thus, although there were these exceptional cases of high inconsistencies, and a few more cases of low positive ρ, for the baseline models, on average and over all passes, the stability of performance across all data variants between CV and test subsets was quite high.

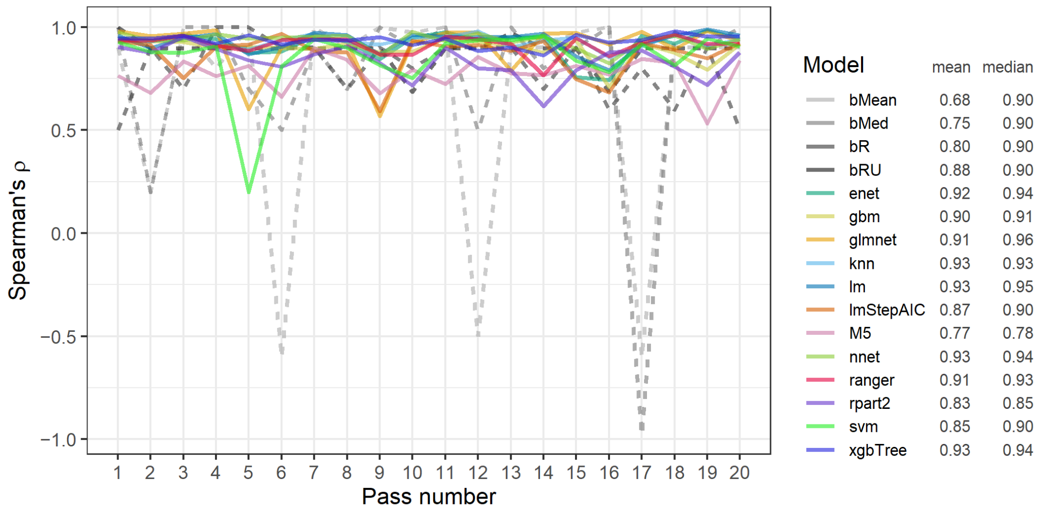


Figure 7. Spearman's ρ for MAE between CV and test subsets for each model and pass across all data variants. The values in braces denote the mean and median values of ρ for each model across all passes.

3.5. RP3: Identification of the Best Combinations of Models and Data Variants

Table 8 ranks models and data variant combinations across all passes. It illustrates the best 20 combinations overall and the single best combination for models outside of them. Among these top-20 combinations, eight used the enet model, four used the nnet or lmStepAIC, and two used the lm or xgbTree. All of them were based on data ratings D-D-* or B-B-* and thresholds of valid values in the range $[0.3, 0.7]$. Among these top-performing combinations, the values of the *difference* were close to each other. The most number of wins was earned by a combination of nnet with D-D-0.4, which reached the second-best place overall. The combination of enet with D-D-0.4, the best combination overall, also had the fewest losses.

Among combinations outside the top-20, the best data variants for particular models used data rating D-D-* for svm, knn, ranger, M5, and gbm models. The best combinations for models rpart2 and glmnet involved B-A-0.9 and A-A-0.3, respectively. Combinations using ratings A-A-*, A-B-*, or B-A-* were the best-performing only for the worst models, rpart2, glmnet, and the baselines. The best combinations for bMean and bMedian performed better than the best combination for the glmnet model. Combinations involving bRU and bR gained the fewest wins and ties because they consistently delivered the least accurate predictions, usually far from the actual values and from the predictions from other models.

The threshold of valid cases influenced the number of predictors that would be kept in the training dataset. However, for baselines, this threshold did not influence predictions because of the nature of these models. Specifically, they do not use the values of predictors to deliver predictions for the outcome values. For example, a set of combinations of bMean with rating B-A-* were all tied at rank 322. These combinations delivered the exact predictions for all thresholds for valid values.

Table 8. Summary of the performance of the best combinations of models and data variants.

Rank	Model	Data Variant [1]	Wins	Ties	Losses	Difference
1	enet	D-D-0.3	10,869	1812	99	10,770
2	nnet	D-D-0.4	10,888	1681	211	10,677
3	enet	B-B-0.3	10,789	1840	151	10,638
4	enet	D-D-0.4	10,742	1896	142	10,600
5	enet	D-D-0.5	10,758	1859	163	10,595
6	nnet	B-B-0.3	10,795	1769	216	10,579
7	enet	B-B-0.4	10,708	1897	175	10,533
8	nnet	B-B-0.4	10,689	1922	169	10,520
9	nnet	D-D-0.3	10,657	1879	244	10,413
10	lmStepAIC	D-D-0.4	10,441	2216	123	10,318
11	enet	D-D-0.6	10,407	2220	153	10,254
12	lm	D-D-0.3	10,542	1944	294	10,248
13	xgbTree	D-D-0.7	10,388	2205	187	10,201
14	lmStepAIC	B-B-0.5	10,290	2387	103	10,187
15	lmStepAIC	D-D-0.5	10,392	2177	211	10,181
16	enet	B-B-0.5	10,335	2283	162	10,173
17	enet	D-D-0.7	10,333	2273	174	10,159
18	lm	D-D-0.6	10,286	2360	134	10,152
19	lmStepAIC	B-B-0.4	10,285	2348	147	10,138
20	xgbTree	D-D-0.4	10,364	2187	229	10,135
23	svm	D-D-0.8	10,233	2354	193	10,040
78	knn	D-D-0.3	9136	3302	342	8794
85	ranger	D-D-0.4	8967	3412	401	8566
155	M5	D-D-0.8	7388	3967	1425	5963
168	gbm	D-D-0.8	7150	3797	1833	5317
272	rpart2	B-A-0.9	5791	2663	4326	1465
322	bMean	B-A-*	4656	3186	4938	−282
340	bMed	B-A-*	4160	3290	5330	−1170
423	glmnet	A-A-0.3	2389	4173	6218	−3829
559	bR	A-B-*	1196	1137	10,447	−9251
601	bRU	A-A-*	607	298	11,875	−11,268

[1] The '*' symbol denotes a wildcard and reflects a set of data variants for particular data ratings but varying thresholds for a fraction of valid values.

4. Discussion

4.1. Experimental Results

Investigation of the RP1 revealed that across the range of analysed data variants, the svm model performed the best, and the enet model was ranked second. Other models performed noticeably worse than these two leaders. However, the ranking of models varied for particular groups of data variants. Specifically, the svm model performed the best for ratings A-A-*, A-B-*, and for thresholds of valid values in the range $[0.5, 1.0]$. The enet model performed the best for rating B-A-* and for thresholds of valid values in the range $[0.3, 0.4]$. The nnet model performed the best for ratings B-B-* and D-D-*, which confirmed that such models require more data for good performance.

The variability of model performance for particular data variants was noticeable, but the change of relative ranks of models usually did not exceed 2–4 places. Thus, model performance across data variants and their groups was quite stable. The only significant exception was a relatively high performance for two baseline models, bMean and bMedian, for thresholds of valid values in the range $[0.9, 1.0]$. This was likely caused by the fact that, for these thresholds, there were very few predictors left in the training subset, and even models regarded as accurate could not learn adequate patterns from the data to predict testing effort accurately. As revealed while investigating the RP1A, the variability of models among passes was noticeable but not significant to the degree that particular models would perform exceptionally better or worse, as in the other passes.

Investigation of the RP2 revealed that, for most models, especially those that were the most accurate, the best performance was achieved with data variants based on ratings D-D-* and B-B-*. No consistent pattern regarding the optimum threshold of valid values common for all models was identified, i.e., for particular models, different threshold values were the best.

Investigation of the RP2A revealed that, for most models in most passes, the performance of particular data variants on the CV subset was consistent with their performances on the test subset. The most noticeable deviations were for bMean and bMedian models but, among non-baseline models, only for selected models in the individual passes.

Although the best model overall across all data variants and passes was the svm, the analysis of the RP3 demonstrated that the best combination of the model with data variant was enet with D-D-0.4, and the best combination involving svm was ranked 23rd (with data variant D-D-0.8). Some combinations performed exceptionally well for models in the middle of the overall ranking of models. Some were in the top-20 or even top-10 of the best combinations. They included models like lm, xgbTree, and, most importantly, lmStepAIC.

All data variants in the best-performing combinations of particular models were based on ratings D-D-*, most often with thresholds of valid values in the range $[0.3, 0.4]$. Hence, these top models performed the best on the higher volumes of data, i.e., the higher numbers of rows and attributes. This, in itself, was not surprising. However, this held, even though the larger datasets contained cases of poor ratings and/or attributes with many missing values. Let us revisit the general question for this study formulated in Section 1: "*Which of the following is it better to use to train predictive models for software testing effort prediction: (1) the entire available dataset containing some low-quality data or (2) on a subset of the entire dataset containing selected higher-quality data?*" The results obtained demonstrated that the first of these approaches should be preferred, at least for the best-performing models.

4.2. Related Work

One of the closest studies to the current one on software testing effort prediction was performed recently [7]. It also used the ISBSG dataset (but the older edition from 2018) and involved comparing several prediction models. The author did not provide an overall ranking of models but identified the best-performing models for particular subsets of the data. These subsets were selected depending on a mixture of attributes describing the project environment, e.g., the type of development, development platform, language type, and sizing method. For most of these subsets, the best-performing model was svm. The obtained results are not comparable with the current study because of the significant methodological differences, most notably including no evaluation on an independent test subset, creation of models with just a single predictor of *functional size*, and arbitrary data selection to projects with ratings B-B.

Some studies used a more detailed set of attributes. For example, Ref. [26] used individual predictors like input count, output count, enquiry, file, and interface counts. These attributes were used to calculate the *functional size* (number of function points). In that study, both *functional size* and those individual attributes were used as predictors causing dependencies among predictors. Such a decision was not justified in that paper.

Several studies on software testing effort prediction used datasets much smaller than ISBSG. Some of them used less popular predictive models not included in this study, like Bayesian classification, particle swarm optimisation, genetic algorithms, differential evolution, firefly and water wave algorithms, bat algorithm, fuzzy techniques, and/or other extensions to linear regression algorithms. An overview of these studies was presented in [6]. Due to the differences between the used datasets and/or the research methodology, results from these studies are not comparable to the current study.

A study in [55] involved an investigation of the relationship between data ratings and the total project effort in the ISBSG dataset. Specifically, the study analysed which combinations of *Data Quality Rating* and *UFP Rating* in a linear model reached the highest coefficient of determination r^2. The authors found that the best-performing rating was A-A, followed by A-B and B-B (tied), and the worst was B-A. Nevertheless, all differences were minor. It was not a predictive study, i.e., these models were not evaluated on a separate test subset, but the test data were the same as for learning the model. Furthermore, these models explicitly included the *Year of Project*, reflecting a possible trend in productivity changing over the years. The current study did not use the *Year of Project* as a predictor.

The current study used single models or combinations of models of the same type. To achieve better predictive accuracy, it may be preferable to use more complex ensemble methods [56,57] by integrating models of different types. Following such an approach may lead to increased predictive accuracy. However, such ensembles were not explored here because the aim was to use popular models in similar studies. Ensemble models are less popular and with no generally established approach on how to create them for such comparative studies in software (testing) effort prediction. Hence, they are often not regarded as reference models for comparisons.

4.3. Utilizing the Results

This study created three main rankings: models for particular data variants, data variants for particular models, and the combinations of models and data variants. The results obtained were achieved on solid methodological grounds, including appropriate statistical tests and repetitions of random data splits. Thus, these rankings may be helpful as a starting point for researchers and practitioners wishing to conduct an extended experiment on software testing prediction.

Specifically, the results obtained demonstrated that, for most models, especially those delivering the most accurate predictions, arbitrarily restricting data for experimentation to projects with ratings A or B is unjustified. Most models performed the best for ratings D-D-*. To highlight the importance of this, note a significant difference to the studies using only projects with ratings B-B or even with higher quality ratings. Such upfront data restriction causes not only model training to be performed on such top-quality data but also model evaluation on such a subset. In contrast, in this study, the data filtering by these ratings was performed only for the training data. However, model evaluation was performed on the entire dataset, regardless of data ratings. Therefore, we might have expected that, because the test data contained lower quality data, the evaluation of predictive models on such data would have revealed their poor performance. However, it was observed only for a few models, usually not those top-performing.

The main practical recommendation supported by the results obtained is as follows: when training predictive models for software testing effort prediction with the ISBSG dataset and machine learning methods, it is best to use all data rows, i.e., to include projects with low categories of C and D for *Data Quality Rating* and *UFP Rating*. The obtained results did not allow the formulation of a universal recommendation regarding the treatment of missing values, as the best strategies depended on the particular models. For example, for models like enet, nnet, lm, lmStepAIC, knn, and ranger, it is best to set low values of the threshold of valid values, like $[0.3, 0.4]$, even though this means that the remaining large fraction of missing values would be filled with the mean or mode. For other models, like xgbTree, svm, M5, and gbm, it is best to set medium-high values of this threshold, like $[0.7, 0.8]$. For no model, is it advisable to set a threshold of 1, where only attributes with no missing values would be used.

4.4. Threats to Validity

The data selected for this study may be considered a threat to validity because only a single dataset was used. This selection of the ISBSG dataset was motivated by its popularity in similar studies on software effort prediction andbecause, according to the author's knowledge, this is the only publicly available dataset providing data ratings. In addition, the study was performed on a subset of the dataset because the raw dataset did not contain essential data for excluded projects, especially on the testing effort, or where project sizing and effort were estimated with methods other than the most popular and those selected for this study. Still, even after such data selection, the dataset used in this study was quite large and contained reasonably diverse projects.

Another group of threats is related to the arbitrary decisions made throughout the experiment related to the selection of models and data preparation steps for particular models, attribute filtering, number of passes, number of folds in CV, values of model parameters, and sets of hyperparameters, evaluation measures, etc. For example, a symmetrical absolute error measure was used to evaluate model performance. It treated both underestimation and overestimation equivalently. However, in some contexts, focusing on one of them might be essential. For example, it might be desirable to pay more attention to the underestimation that may lead to problems with funding a project or to the overestimation that may lead to unnecessary allocation of resources. This problem was not explored here as it depends on the project context, but the used datasets do not contain sufficient data. In general, all these decisions were motivated by the goal of this study, decisions made in similar studies and the author's preliminary experiments, and the balance between the duration of calculations (mainly model training time) and the ability to achieve good solid results for solving the stated research problems. Due to such a trade-off, this study did not focus on global optimisation, e.g., in terms of identifying globally optimal hyperparameters for predictive models. Still, as mentioned before, sub-optimal solutions were sufficient and in line with the goals of this study.

4.5. Future Work

The ISBSG dataset contains two attributes reflecting the quality of the data. However, other approaches propose comprehensive and more detailed data quality evaluation involving more data quality characteristics [21,58–60]. Thus, future work may focus on investigating these additional characteristics. An example of such an extension may be incorporating the influence of the outliers. Although such a topic was investigated in isolation in [14,15,26], it may be worth examining it, together with other dimensions of data quality analysed in this study. Other future work paths may include those intentionally omitted in this study, e.g., exploring a range of ensemble models or more data, including projects with other sizing and effort estimate methods.

5. Conclusions

The results obtained in the experiments of this study led to drawing up the following conclusions:

- Among the best three models overall, the svm (support vector machines) performed the best for the most restrictive ratings A-A and A-B, enet (elastic net) performed the best for ratings B-A, and nnet (neural network) performed the best for the least restrictive ratings B-B and D-D.
- Using restrictive data variants, i.e., involving ratings A-A, A-B or B-A that reduced the number of rows (projects) and high thresholds of valid values of 1.0 or 0.9, led to poor performance of most models.
- Most models, especially the most accurate, performed the best with data variants using ratings D-D. The performance of most models on data variants with ratings B-B was lower.
- No global optimum threshold of valid values common for all models was detected as it was specific for a particular model.
- The relative rank of models for particular groups of data variants, including only particular data ratings or only thresholds of valid values, was quite stable.
- Performance of data variants for most models and most passes, excluding less than ten exceptions, was stable between CV and test subsets.

The standard practice in most experiments described in the existing literature on software effort prediction using the ISBSG dataset is an arbitrary and restrictive data selection only to projects with *Data Quality Rating* and *UFP Rating* with values A or B. The most important conclusion from this study is that such a standard approach does not seem justified in terms of the performance of most predictive models for testing effort prediction. Therefore, it is recommended not to exclude cases with low data ratings. Instead, training data with all cases, regardless of their ratings, should be used to achieve better accuracy of most predictive models.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: This research was made possible by ISBSG—https://www.isbsg.org, accessed on 1 January 2021. According to the ISBSG Research Policy & Procedure, it is not possible to publish any raw data provided by ISBSG. Because scripts with code used for data processing and prediction, which were implemented by the author, reveal portions of such data, for the same reason they are also not publicly available.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Wen, J.; Li, S.; Lin, Z.; Hu, Y.; Huang, C. Systematic literature review of machine learning based software development effort estimation models. *Inf. Softw. Technol.* **2012**, *54*, 41–59. [CrossRef]
2. Jorgensen, M.; Shepperd, M. A Systematic Review of Software Development Cost Estimation Studies. *IEEE Trans. Softw. Eng.* **2007**, *33*, 33–53. [CrossRef]
3. Ali, A.; Gravino, C. A systematic literature review of software effort prediction using machine learning methods. *J. Softw. Evol. Process.* **2019**, *31*, e2211 . [CrossRef]
4. Eduardo Carbonera, C.; Farias, K.; Bischoff, V. Software development effort estimation: A systematic mapping study. *IET Softw.* **2020**, *14*, 328–344. [CrossRef]
5. Mahmood, Y.; Kama, N.; Azmi, A.; Khan, A.S.; Ali, M. Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. *Softw. Pract. Exp.* **2022**, *52*, 39–65. [CrossRef]
6. Bluemke, I.; Malanowska, A. Software Testing Effort Estimation and Related Problems: A Systematic Literature Review. *ACM Comput. Surv.* **2021**, *54*, 1–38. [CrossRef]
7. López-Martín, C. Machine learning techniques for software testing effort prediction. *Softw. Qual. J.* **2022**, *30*, 65–100. [CrossRef]
8. ISBSG. *ISBSG Repository Data Release 2020 R1*; International Software Benchmarking Standards Group: Melbourne, Australia, 2020.
9. ISBSG. *Guidelines for Use of the ISBSG Data*; International Software Benchmarking Standards Group: Melbourne, Australia, 2020.
10. López-Martín, C. Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects. *Appl. Soft Comput.* **2015**, *27*, 434–449. [CrossRef]
11. Mendes, E.; Lokan, C.; Harrison, R.; Triggs, C. A Replicated Comparison of Cross-Company and Within-Company Effort Estimation Models Using the ISBSG Database. In Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS'05), Como, Italy, 19–22 September 2005; IEEE: Piscataway, NJ, USA, 2005; p. 36. [CrossRef]
12. Huijgens, H.; van Deursen, A.; Minku, L.L.; Lokan, C. Effort and Cost in Software Engineering: A Comparison of Two Industrial Data Sets. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden, 15–16 June 2017; ACM: New York, NY, USA, 2017; pp. 51–60. [CrossRef]
13. Gencel, C.; Heldal, R.; Lind, K. On the Relationship between Different Size Measures in the Software Life Cycle. In Proceedings of the 2009 16th Asia-Pacific Software Engineering Conference, Batu Ferringhi, Malaysia, 1–3 December 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 19–26. [CrossRef]
14. Seo, Y.S.; Bae, D.H. On the value of outlier elimination on software effort estimation research. *Empir. Softw. Eng.* **2013**, *18*, 659–698. [CrossRef]
15. Ono, K.; Tsunoda, M.; Monden, A.; Matsumoto, K. Influence of Outliers on Estimation Accuracy of Software Development Effort. *IEICE Trans. Inf. Syst.* **2021**, *E104.D* , 91–105. [CrossRef]
16. Mittas, N.; Angelis, L. Ranking and Clustering Software Cost Estimation Models through a Multiple Comparisons Algorithm. *IEEE Trans. Softw. Eng.* **2013**, *39*, 537–551. [CrossRef]
17. Murillo-Morera, J.; Quesada-López, C.; Castro-Herrera, C.; Jenkins, M. A genetic algorithm based framework for software effort prediction. *J. Softw. Eng. Res. Dev.* **2017**, *5*, 4. [CrossRef]
18. Radlinski, L. Stability of user satisfaction prediction in software projects. *Procedia Comput. Sci.* **2020**, *176*, 2394–2403. [CrossRef]
19. Mendes, E.; Lokan, C. Replicating studies on cross- vs. single-company effort models using the ISBSG Database. *Empir. Softw. Eng.* **2008**, *13*, 3–37. [CrossRef]
20. Liebchen, G.A.; Shepperd, M. Data sets and data quality in software engineering. In Proceedings of the 4th International Workshop on Predictor Models in Software Engineering, Leipzig, Germany, 12–13 May 2008; ACM: New York, NY, USA, 2008; pp. 39–44. [CrossRef]
21. Bosu, M.F.; Macdonell, S.G. Experience: Quality Benchmarking of Datasets Used in Software Effort Estimation. *J. Data Inf. Qual.* **2019**, *11*, 1–38. [CrossRef]

22. Fernández-Diego, M.; González-Ladrón-de Guevara, F. Application of mutual information-based sequential feature selection to ISBSG mixed data. *Softw. Qual. J.* **2018**, *26*, 1299–1325. [CrossRef]
23. Sarro, F.; Petrozziello, A. Linear Programming as a Baseline for Software Effort Estimation. *ACM Trans. Softw. Eng. Methodol.* **2018**, *27*, 1–28. [CrossRef]
24. Whigham, P.A.; Owen, C.A.; Macdonell, S.G. A Baseline Model for Software Effort Estimation. *ACM Trans. Softw. Eng. Methodol.* **2015**, *24*, 1–11. [CrossRef]
25. Fernández-Diego, M.; González-Ladrón-de Guevara, F. Potential and limitations of the ISBSG dataset in enhancing software engineering research: A mapping review. *Inf. Softw. Technol.* **2014**, *56*, 527–544. [CrossRef]
26. Gautam, S.S.; Singh, V. Adaptive Discretization Using Golden Section to Aid Outlier Detection for Software Development Effort Estimation. *IEEE Access* **2022**, *10*, 90369–90387. [CrossRef]
27. Xia, T.; Shu, R.; Shen, X.; Menzies, T. Sequential Model optimization for Software Effort Estimation. *IEEE Trans. Softw. Eng.* **2022**, *48*, 1994–2009. [CrossRef]
28. Kuhn, M. *caret: Classification and Regression Training*; R Package Version 6.0-93; 2022. Available online: https://CRAN.R-project.org/package=caret (accessed on 21 October 2022).
29. R Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2022. Available online: https://www.R-project.org/ (accessed on 8 September 2022).
30. Kitchenham, B.; Mendes, E. Why comparative effort prediction studies may be invalid. In Proceedings of the 5th International Conference on Predictor Models in Software Engineering, Vancouver, BC, Canada, 18–19 May 2009; ACM: New York, NY, USA, 2009; pp. 1–5. [CrossRef]
31. Radliński, Ł. Preliminary evaluation of schemes for predicting user satisfaction with the ability of system to meet stated objectives. *J. Theor. Appl. Comput. Sci.* **2015**, *9*, 32–50.
32. Radliński, Ł. Predicting Aggregated User Satisfaction in Software Projects. *Found. Comput. Decis. Sci.* **2018**, *43*, 335–357. [CrossRef]
33. Radliński, Ł. Predicting User Satisfaction in Software Projects using Machine Learning Techniques. In Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering, Prague, Czech Republic, 5–6 May 2020; Ali, R., Kaindl, H., Maciaszek, L., Eds.; SciTePress: Setubal, Portugal, 2020; Volume 1, pp. 374–381. [CrossRef]
34. Zou, H.; Hastie, T. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B (Stat. Methodol.)* **2005**, *67*, 301–320. [CrossRef]
35. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
36. Friedman, J.; Hastie, T.; Tibshirani, R. Regularization Paths for Generalized Linear Models via Coordinate Descent. *J. Stat. Softw.* **2010**, *33*, 1–22. [CrossRef]
37. Altman, N.S. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *Am. Stat.* **1992**, *46*, 175–185. [CrossRef]
38. Wilkinson, G.N.; Rogers, C.E. Symbolic Description of Factorial Models for Analysis of Variance. *Appl. Stat.* **1973**, *22*, 392. [CrossRef]
39. Venables, W.N.; Ripley, B.D. *Modern Applied Statistics with S*, 4th ed.; Statistics and Computing; Springer: New York, NY, USA, 2002. [CrossRef]
40. Wang, Y.; Witten, I.H. Induction of model trees for predicting continuous classes. In Proceedings of the Poster Papers of the European Conference on Machine Learning, Prague, Czech Republic, 23–25 April 1997; University of Economics, Faculty of Informatics and Statistics: Prague, Czech Republic, 1997.
41. Witten, I.; Frank, E.; Hall, M. *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed.; Elsevier: Amsterdam, The Netherlands, 2011. [CrossRef]
42. Ripley, B.D. *Pattern Recognition and Neural Networks*; Cambridge University Press: Cambridge, UK, 1996. [CrossRef]
43. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
44. Breiman, L.; Friedman, J.; Stone, C.J.; Olshen, R. *Classification and Regression Trees*; Chapman & Hall: Boca Raton, FL, USA, 1984.
45. Chang, C.C.; Lin, C.J. LIBSVM. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 1–27. [CrossRef]
46. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining-KDD '16, San Francisco, CA, USA, 13–17 August 2016; ACM Press: New York, NY, USA, 2016; pp. 785–794. [CrossRef]
47. Villalobos-Arias, L.; Quesada-López, C. Comparative study of random search hyper-parameter tuning for software effort estimation. In Proceedings of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering, Athens Greece, 19–20 August 2021; ACM: New York, NY, USA, 2021; pp. 21–29. [CrossRef]
48. Luo, G. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. *Netw. Model. Anal. Health Inform. Bioinform.* **2016**, *5*, 18. [CrossRef]
49. Minku, L.L. A novel online supervised hyperparameter tuning procedure applied to cross-company software effort estimation. *Empir. Softw. Eng.* **2019**, *24*, 3153–3204. [CrossRef]
50. Kocaguneli, E.; Menzies, T.; Bener, A.; Keung, J.W. Exploiting the Essential Assumptions of Analogy-Based Effort Estimation. *IEEE Trans. Softw. Eng.* **2012**, *38*, 425–438. [CrossRef]
51. Shepperd, M.; MacDonell, S. Evaluating prediction systems in software project estimation. *Inf. Softw. Technol.* **2012**, *54*, 820–827. [CrossRef]

52. Kocaguneli, E.; Menzies, T.; Keung, J.; Cok, D.; Madachy, R. Active learning and effort estimation: Finding the essential content of software effort estimation data. *IEEE Trans. Softw. Eng.* **2013**, *39*, 1040–1053. [CrossRef]
53. Menzies, T.; Jalali, O.; Hihn, J.; Baker, D.; Lum, K. Stable rankings for different effort models. *Autom. Softw. Eng.* **2010**, *17*, 409–437. [CrossRef]
54. Benjamini, Y.; Hochberg, Y. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *J. R. Stat. Soc. Ser. B (Methodol.)* **1995**, *57*, 289–300. [CrossRef]
55. Fernández-Diego, M.; Martínez-Gómez, M.; Torralba-Martínez, J.M. Sensitivity of results to different data quality meta-data criteria in the sample selection of projects from the ISBSG dataset. In Proceedings of the 6th International Conference on Predictive Models in Software Engineering-PROMISE '10, Timisoara, Romania, 12–13 September 2010; ACM Press: New York, NY, USA, 2010; pp. 1–9. [CrossRef]
56. Ceran, A.A.; Ar, Y.; Tanrıöver, Ö.Ö.; Seyrek Ceran, S. Prediction of software quality with Machine Learning-Based ensemble methods. *Mater. Today Proc.* **2022**. [CrossRef]
57. Minku, L.L.; Yao, X. An Analysis of Multi-objective Evolutionary Algorithms for Training Ensemble Models Based on Different Performance Measures in Software Effort Estimation. In Proceedings of the 9th International Conference on Predictive Models in Software Engineering, Baltimore, MD, USA, 9 October 2013; ACM: New York, NY, USA, 2013; pp. 8:1–8:10. [CrossRef]
58. Bosu, M.F.; MacDonell, S.G. A Taxonomy of Data Quality Challenges in Empirical Software Engineering. In Proceedings of the 2013 22nd Australian Software Engineering Conference, Hawthorne, VIC, Australia, 4–7 June 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 97–106. [CrossRef]
59. Rosli, M.M.; Tempero, E.; Luxton-Reilly, A. Evaluating the Quality of Datasets in Software Engineering. *Adv. Sci. Lett.* **2018**, *24*, 7232–7239. [CrossRef]
60. Shepperd, M. Data quality: cinderella at the software metrics ball? In Proceedings of the 2nd International Workshop on Emerging Trends in Software Metrics, Honolulu, HI, USA, 24 May 2011; ACM: New York, NY, USA, 2011; pp. 1–4. [CrossRef]