A Meta-analytical Comparison of Naive Bayes and Random Forest for Software Defect Prediction.

A meta-analytical comparison of Naive Bayes and Random Forest for software defect prediction

Ch Muhammad Awais[1], Wei Gu[1], Gcinizwe Dlamini[1], Zamira Kholmatova[1], and Giancarlo Succi[2]

Innopolis University, Russia[1], Università di Bologna, Italy[2]
{c.awais,g.wei,g.dlamini,z.kholmatova}@innopolis.university,
g.succi@unibo.it

Abstract. Is there a statistical difference between Naive Bayes and Random Forest in terms of recall, f-measure, and precision for predicting software defects? By utilizing systematic literature review and meta-analysis, we are answering this question. We conducted a systematic literature review by establishing criteria to search and choose papers, resulting in five studies. After that, using the meta-data and forest-plots of five chosen papers, we conducted a meta-analysis to compare the two models. The results have shown that there is no significant statistical evidence that Naive Bayes perform differently from Random Forest in terms of recall, f-measure, and precision.

Keywords: Random Forest · Naive Bayes · Defect Prediction · Software Defect Prediction · Meta-analysis.

1 Introduction

A defect that causes software to behave unexpectedly in a way that does not meet the actual requirements is known as a software defect [8]. Software defects can be heavily influential, which gives rise to disasters. Nowadays, software systems are increasing steadily in size and complexity [26]. Industry has developed sophisticated software testing methodologies [14]. To ensure quality, newly developed or modified engineering products are often subjected to rigorous testing.

However, the nature of software testing is time-consuming and resource-hungry, typically the software testing process takes up approximately 40%-50% of the total development resources, 30% of the total effort and 50%-60% of the total cost of software development [12]. Over the past two decades, a variety of machine learning approaches have been employed to detect software defects in an endeavor to reduce, to minimize cost and automate the task of software testing [3].

In a typical workflow for building an ML defect prediction model, the data usually comes from the version control system and contains the source code and commit messages. Accordingly, an instance can be labelled as defective or non-defective. The instances in the data set are then constructed based on the acquisition of tags and other messages in the version control system, and a defect

prediction model can be built by using a set of training instances. Finally, the prediction model can classify whether a new test instance has a defect or not [16]. Our focus is on comparing prediction methods based on the following two ML models: Naive Bayes (NB) Random forest (RF) [11]

In this paper, we present a systematic literature review and meta-analysis of two machine learning methods for software defect prediction. Our motivation is to help researchers and practitioners in the field of software defect prediction to build a better understanding of these two algorithms that have been applied from a very early stage. The goal is to find out if there is any significance performance difference between NB and RF on software defect prediction in terms of precision, recall and f-measure.

The remaining sections of this paper are organized as follows: In section 2, related works are outlined. Section 3 describes the methodology, systematic literature review and meta-analysis. Section 4 presents the result and the discussion. Section 5 outlines the conclusion.

2 Related Works

Software defect prediction is executed using some software related data, that consists of information about the code for software development, i.e. lines of code, number of methods, inheritance etc. In the paper [9], researchers have discussed the effects of the metric set on software defect prediction, which summarizes that, if we precisely simplify metrics/features of our dataset, then the simplest model will perform better for software defect prediction, meaning that the feature selection plays an important role in software defect prediction.

Machine learning models [5] are used for performing software defect prediction. The researchers [23] discussed the techniques for building an ideal model capable of detecting unseen defects in software. Software defect prediction models only focus on training on a specific domain of data, i.e., open-source projects, whereas an ideal model is trained on commercial projects. Moreover, preprocessing (i.e feature selection, multi-co-linearity) of the dataset for training, can improve the performance of the defect prediction model significantly [23].

Meta-analysis is the procedure of combining different studies for finding out whether the experiments performed in different studies [13] follow the same distribution or not. It helps in validating the results in different studies. The limitations of meta-analysis for defect prediction [20], resulted in a guide for future researchers for meta-analytical studies on software defect prediction.

Software defect prediction can vary based on the knowledge and skills of the researcher [19]. In [19], a meta-analytical experiment helped in finding bias in the results of software defect prediction, comparing 4 different groups (Classifier's effect, Dataset effect, effect of metrics and effect of researchers). The results showed that the effect of the classifier on the result was smaller than the effect of the researcher on the study result.

Soe et al. [22] performed experiments on 12 different datasets, in nine datasets RF performed better than NB with a marginal difference, in one the accuracy

was the same, and in one NB outperformed RF with a difference of 0.07%. The accuracy of NB on the PC3 dataset was i.e 45.8%, while RF achieved 89.76%, which made a huge difference of 43.89%, because of these differences the researchers stated that Random Forest is better. We are testing the claim of [22], using meta-analysis and systematic literature review.

3 Methodology

Our proposed methodology is presented in Fig. 1. It contains two main stages: A) Literature Review and B) Meta-analysis. The following sections outline the details about each stage.

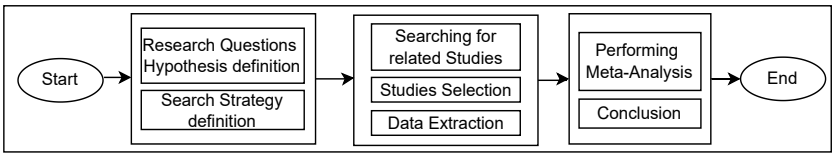


Fig. 1: Methodology Overview

3.1 Literature Review

To test our aforementioned hypothesis using a meta-analytical approach and address the research questions, we conducted a systematic literature review.

Research Questions and hypothesis definition: Research questions (RQ) helps in steering research to valuable and constructive conclusions [2]. We apply PICOC(Population, Intervention, Comparison, Outcomes, and Context) [17] approach to produce five main research questions, which drive the analysis application of machine learning techniques (i.e naive Bayes, Random forest) in software defect prediction.

RQ1. Which repositories are popular for software defect prediction?
RQ2. What kinds of datasets are the most commonly utilized in prediction of software defects?
RQ3. What are the programming languages which commonly in datasets for software defect prediction?
RQ4. What are the common methods that are used to predict software defects?
RQ5. What comparison metrics are commonly used for comparing the software defect prediction performance?

Search strategy: Before beginning the search, it is important to choose a suitable combination of databases to enhance the chances of discovering extremely relevant articles. Digital libraries are the databases where the papers are published, it is an important step to decide which libraries to choose. To have the broadest set of studies possible, it is important to search for literature databases that offer a broad perspective on the field. We used Google Scholar (GS), Research Gate (RG), Springer (SP) as search databases.

Based on our research questions we formulated the following query string: (software OR applicati* OR systems) AND (fault* OR defect* OR quality OR error-prone) AND (predict* OR prone* OR probability OR assess* OR detect* OR estimat* OR classificat*) AND (random* OR forest) AND (naive OR bayes)

The human behavior may differ based on their own selection criteria, so the search string was adjusted based on the trends being used for searches, also the original one was kept. After the search string was adjusted to suit the specific requirements of each database, the databases were searched by title, keyword, and abstract. Search results limited to publications published from 1999 to 2021 were retrieved. Only English journal papers and conference proceedings were included.

Study selection: We will discuss the tools and techniques used for selecting the studies for performing the meta-analytical study. Also in this section we will explain the process of filtering retrieved studies requires outlining inclusion and exclusion criteria. To eliminate bias in the systematic literature review procedure, we defined the inclusion and exclusion criteria using PICOC approach in advance.

- **Inclusion Criteria**:
  We included studies that
  - are discussing NB and RF models in software defect prediction.
  - are with publicly available datasets for software defect prediction.
  - benchmarked performance with appropriate metrics for comparison.
- **Exclusion criteria**
  - Studies that do not include our research objects.
  - Studies with unpublished datasets.
  - Studies that are with no validation of experimental results or experimental process.
  - Non-peer-reviewed studies.
  - Systematic review studies.

Initially, the literature reviews and non peer-reviewed papers were removed. We created groups in Mendeley[25] to collaborate on the screening of studies. Two reviewers independently reviewed all the works, decisions on whether to screen a study were made according to established exclusion and inclusion criteria. When reviewers disagree, a discussion will occur and an explanation for screening will be provided. If there is uncertainty for the final decision, then the literature will be included for analysis.

Selection results: We have collected 62 research studies based on our search query from different databases. On this stage we applied screening based on title(Removal of papers with unrelated titles) and removal of duplicates.

Assessment criteria: We utilized a brief checklist inspired by Ming's [15] work to determine whether the study provides adequate contextual and methodological information, to interpret if the study is sufficient to answer the research questions. After a pilot test, we fine-tuned the criteria for our research questions using the following different stages.

- **Stage 1:** Context Criteria
  - The study's objective and domain should be clearly stated.
  - The programming language for benchmarking must be specified.
  - The source information of publicly available dataset should be provided.
- **Stage 2:** Model building Criteria
  - The features used for training(e.g, software metrics), must be explicit.
  - The dependent variable predicted by the models should be stated clearly.
  - The granularity of the dependent variable should be reported, i.e., whether the predicted outcome is a statistic on LOC, modules, files, or packages.
- **Stage 3:** Data Criteria
  - There should be a report or reference about data collection in order to have a confidence on the dataset.
  - The data transformation or pre-processing techniques should be clearly stated in the study.
- **Stage 4:** Predictive performance Criteria
  - The prediction model must be tested on unseen data after training.

Assessment results: We performed full-text analysis on each study to keep paper, which discussing Naive Bayes and Random Forests. We excluded the studies which were not discussing the specific metrics(i.e. recall, precision, f-measure).

Data Extraction: We carried out a pilot test on three randomized studies to observe if data extraction meets the needs of analysis. Two group members separately designed the table headers, after the pilot test, they discussed and combined the two tables to ensure the structure consistency. The data was extracted according to the established table structure, the consistency of data was compared to ensure that no mistakes were made in the extraction process.

3.2 Meta-Analysis

In meta-analysis, firstly, we find out the outcome of a study, and based on this outcome, statistics are calculated for each study. Secondly, we combine the statistics of all studies to estimate the summary by using the weighted average of each study. Based on the number of experiments, and the publishing year, the study is

given weights or other factors, i.e. study with fewer experiments will get a lower weight etc. [24]. To understand meta-analysis, we need to define a few terms, which are as follows:

- **Effect Size:** The measure of an outcome to validate/invalidate the hypothesis. We used mean-difference because our study is based on the recall score.
- **Forest Plot:** The Forest plot indicates the variability in the results, based on the forest plot we decide the presence of heterogeneity.
- **Heterogeneity:** The validation criterion to decide whether to employ meta-analysis, i.e. high heterogeneity means the studies are not linking up, and it leads to halt in meta-analysis. In our case, low heterogeneity can be later visualized in the forest plots.
- **Effect Model:** We utilized two effects models, i.e. fixed and random. We will discuss them in the next section.

Fixed vs Random Effect Models Fixed effect model is used for best estimation of the effect size, it is presumed that all studies have a single effect. Whereas random effect model is used for average effect, in random effect it is assumed that each study has a different effect [24]. In order to estimate the individual effects of a assessment criterion and the difference between the experimental and control groups, we calculate the effect size. The below formulas from [4] are used for calculation of the Effect Size g and its standard deviation SE_g:

$$g = J \times d \tag{1}$$

j is Hedge's correction factor: $J = 1 - \frac{3}{4df-1}$ and $df = Numberof Studies - 1$ d is Cohen's standardized difference of the sample means:

$$d = \frac{\bar{X}_1 - \bar{X}_2}{S_{\text{within}}} \tag{2}$$

Here X_1 and X_2 are the means of the control group (Naive Bayes) and the experimental group (Random Forest) respectively, with S_{within} being the pooled within-groups standard deviation (with an assumption that $\sigma_1 = \sigma_2$):

$$S_{\text{within}} = \sqrt{\frac{(n_1 - 1)\, S_1^2 + (n_2 - 1)\, S_2^2}{n_1 + n_2 - 2}} \tag{3}$$

where S_1,S_2 are the standard deviation of control group and experimental group and n_1,n_2 are the number of studies of control group and experimental group respectively.
The variance and standard deviation of each Effect Size g is: $V_g = J^2 \times V_d$ and $SE_g = \sqrt{V_g}$
where V_d is the variance of Cohen's d .Here ,again n_1,n_2 are the number of studies of control group and experimental group respectively:

$$V_d = \frac{n_1 + n_2}{n_1 n_2} + \frac{d^2}{2\,(n_1 + n_2)} \tag{4}$$

Finally, each study's weight is calculated as: $W_i^* = \frac{1}{V_{g_i}^*}$ and $V_{g_i}^* = V_{g_i} + T^2$

where V_{g_i} is the within-study variance for a study i and T^2 is the between-studies variance. In this study, τ^2 is estimated with T^2 using the DerSimonian and Laird method [7]

4 Results and Discussions

The results section is discussing the detailed analysis of the discovered literature, then the answers to the research questions, and at the end of the section we discussed the meta-analytical results, presenting the meta-data and forest-plots.

4.1 Literature Review

We found 62 studies, the repository representation is as follows, 47.37% from PROMISE, 42.10% from NASA and 10.53% from OSS. The answers to the research questions are as follows:

RQ1: Which repositories are popular for software defect prediction?
There were two types of repositories, public and private, in most of the studies the three repositories were discussed, PROMISE, NASA being public and OSS being private. When we analyzed the results, we found out PROMISE, NASA are commonly used, because of public availability.

RQ2: What kinds of datasets are the most commonly utilized in prediction of software defects?
It can be visualized from the fig [fig.2], that the researchers mostly used PC3, KCI, PC1, PC4. There is a marginal difference between the usage dataset, also the datasets are used in a combination with other datasets, but we can see that jEdit, KC3 and ANT were least commonly used.

RQ3: What are the programming languages which commonly in datasets for software defect prediction?
In the studies we observed that there were only four languages used, while Java and C were most common, whereas Perl was the least used Fig.3.

Fig. 2: Most common datasets

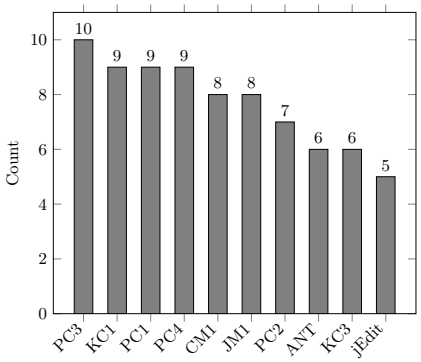
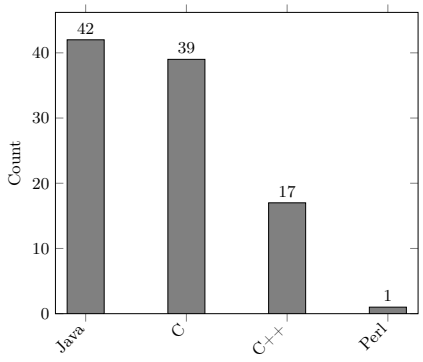


Fig. 3: Dataset languages



RQ4: What are the common methods that are used to predict software defects? As our research query only pinpoints the Naive Bayes and Random Forest, so these were the most common, besides these models we found out that Linear Regression and Decision Tree are also commonly used and Multi-layer perceptron was least used in Fig.5.

RQ5: What comparison metrics are commonly used for comparing the software defect prediction performance?
The top discussed metrics are Accuracy, F-measure, Recall, and precision, one of the reason of these being top was the search query, and it is a known fact that accuracy is the common metric when we discuss ML models. Whereas, it was interesting to observe that precision was used fewer times, the least used metrics were MAE and ROC (Figure.4).

Fig. 4: Common Metrics

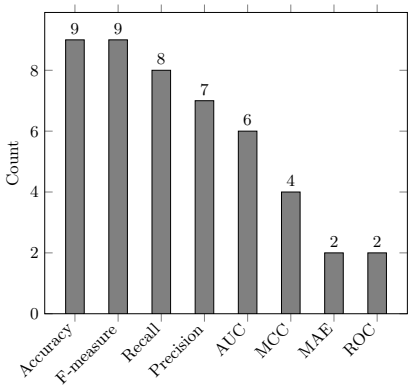
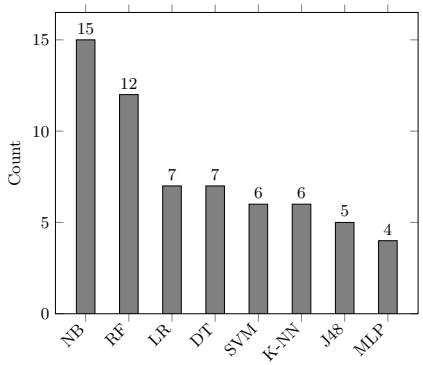


Fig. 5: Common Methods



4.2 Meta-Analysis

Meta-analysis is performed on the data we retrieved after literature review, the studies selected for meta-analysis are presented in Table 1. In the table, NB experiments represents the number of experiments performed for Naive Bayes, and RF for Random Forest. Y represents the presence of a specific measure in the study, and N means absence.

Title	Year	NB Experiments	RF Experiments	Recall	Precision	F-Measure	Total Models	Total Metrics
S1 [10]	2017	4	4	Y	Y	Y	11	4
S2 [18]	2018	7	7	Y	Y	Y	6	5
S3 [6]	2018	3	3	N	N	Y	4	2
S4 [1]	2019	24	24	Y	Y	Y	10	6
S5 [21]	2020	3	3	Y	N	N	5	2

Table 1: Meta-Data

Further, we drew forest plots to analyze the meta-data. Forest plot consists of 10 columns, where each row represents a study, the authors are the identifiers. Total is the number of experiments in the study, and mean and SD refer to the mean and standard deviation of that performance measure. The Standardized Mean Difference refers to the standardized difference between the Mean performance of both groups, and the values are in SMD column. The positions of the shaded rectangles represent the standardized mean differences for a study instance, while their size represents the effect weights.

95% CI represents the confidence interval, verifies the hypothesis for SMD. Weights are the contribution of the study to the meta-analysis. Diamond represents the combined effect of all the studies. Heterogeneity identifies the model (random or fixed effect). Test for overall effect shows overall effect based on degrees of freedom. The forest plots are represented in Fig.6, Fig.7 and Fig.8.

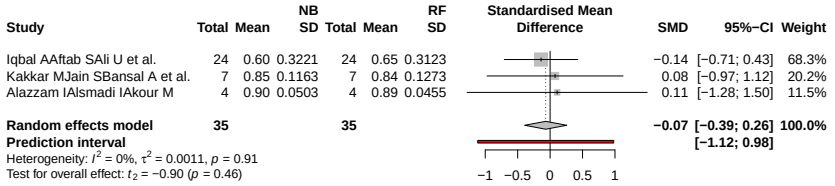


Fig. 6: Forest plot for precision

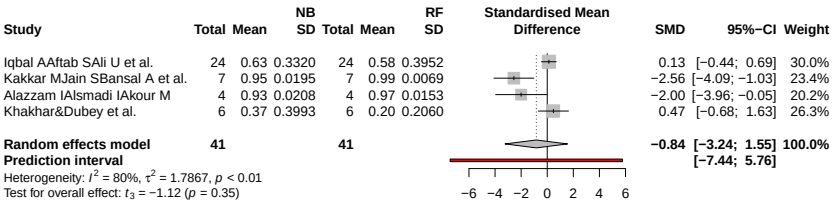


Fig. 7: Forest plot for recall

The standardized mean difference in the study is represented by the horizontal line of a forest plot. When the horizontal and vertical lines of a forest plot intersect, it means that there is no statistically significant difference between the research groups. These horizontal lines create a diamond, that symbolizes the combined impact of all the studies. There is no statistically significant difference between the groups for the studies when the diamond intersects the vertical

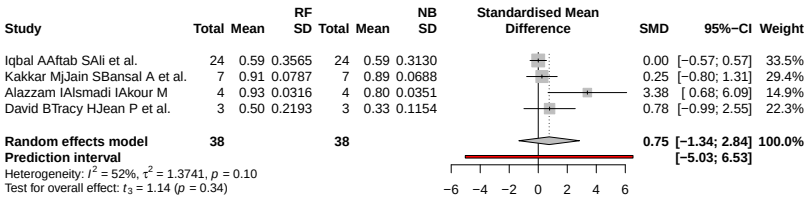


Fig. 8: Forest plot for F-measure

line. From the forest-plots above, we can observe that all the diamonds are intersecting the vertical line, meaning that, the effects of Naive Bayes and Random Forests on software defect prediction are the same. The implementation details are publicly available in Github[1].

5 Conclusions

In this study, we thoroughly reviewed literature assessment and performed a meta-analysis to determine if naive Bayes and random forest are more effective at predicting software problems. We collected the related studies, and presented statistics about databases, datasets, languages, repositories, machine learning models, comparison metrics. We retrieved 62 studies that address Naive Bayes and Random Forest, and five of them were chosen for meta-analysis based on our criteria. We conclude from the meta-analysis that there is no statistically significant difference between random forest and naive Bayes in terms of recall, precision, and f-measure for software defect prediction.

Acknowledgements

This research is supported by the Russian Science Foundation, Grant No. 22-21-00494

References

[1] U Ali A Iqbal S Aftab. “Performance analysis of machine learning techniques on software defect prediction using NASA datasets”. In: (2019).
[2] J Agee. “Developing qualitative research questions: a reflective process”. In: *International Journal of Qualitative Studies in Education* (2009).
[3] Saiqa Aleem, Luiz Fernando Capretz, and Faheem Ahmed. “Benchmarking machine learning technologies for software defect detection”. In: (2015).
[4] Michael Borenstein et al. “Identifying and Quantifying Heterogeneity”. In: *Introduction to meta-analysis*. 2009. Chap. 16, pp. 107–125.

[1] https://github.com/cm-awais/SDP_NB_RF_meta_analysis

[5] Venkata U.B. Challagulla. "Empirical Assessment of Machine Learning based Software Defect Prediction Techniques". In: *Acp j club* (2005).
[6] Jean Petric David Bowes Tracy Hall. "Software defect prediction: do different classifiers find the same defects?" In: *Software Qual J* (2018).
[7] Rebecca DerSimonian and Nan Laird. "Meta-analysis in clinical trials". In: *Controlled clinical trials* (1986).
[8] Sunita Devnani-Chulani. "Modeling software defect introduction". In: *Proc. California Software Symposium*. 1998.
[9] Li Bing He Peng. "An empirical study on software defect prediction with a simplified metric set". In: *Information and Software Technology* (2015).
[10] M. Akour I. Alazzam I. Alsmadi. "Software fault proneness prediction: a comparative study between bagging, boosting, and stacking ensemble and base learner methods". In: (2017).
[11] Shomona Gracia Jacob et al. "Improved random forest algorithm for software defect prediction through data mining techniques". In: *IJCA* (2015).
[12] Divya Kumar and KK Mishra. "The impacts of test automation on software's cost, quality and time to market". In: (2016).
[13] G Leandro. "Meta-analysis in medical research: The handbook for the understanding and practice of meta-analysis". In: *Acp j club* (2008).
[14] William E Lewis. *Software testing and continuous quality improvement*. CRC press, 2017.
[15] Ming Li et al. "Sample-based software defect prediction with active and semi-supervised learning". In: *Automated Software Engineering* (2012).
[16] Zhiqiang Li, Xiao-Yuan Jing, and Xiaoke Zhu. "Progress on approaches to software defect prediction". In: *IET Software* (2018).
[17] H Roberts M Petticrew. "Systematic reviews in the social sciences: A practical guide". In: *Acp j club* (2008).
[18] S. Jain M. Kakkar. "Is Open-Source Software Valuable for Software Defect Prediction of Proprietary Software and Vice Versa?" In: (2018).
[19] David Bowes Martin Shepperd and Tracy Hall. "Researcher Bias: The Use of Machine Learning in Software Defect Prediction". In: (2016).
[20] James Miller. "Applying meta-analytical procedures to software engineering experiments". In: *Journal of Systems and Software* (2000).
[21] R. Dubey P. Khakhar. "The integrity of machine learning algorithms against software defect prediction". In: (2020).
[22] Yan Naung Soe and Khine Khine Oo. "A Comparison of Naıˆve Bayes and Random Forest for Software Defect Prediction". In: ICCA. 2018.
[23] Khari Manju Son Le Hoang Pritam Nakul. "Empirical study of software defect prediction: A systematic mapping". In: *Symmetry* (2019).
[24] University of York. "CRD's guidance for undertaking reviews in health care". In: *Introduction to meta-analysis*. 2009. Chap. 1, pp. 54–55.
[25] Holt Zaugg et al. "Mendeley: Creating communities of scholarly inquiry through research collaboration". In: *TechTrends* (2011).
[26] Horst Zuse. *Software complexity: measures and methods*. Walter de Gruyter GmbH & Co KG, 2019.