Software Defect Prediction Models. A replication study.

1st Nadejde Camelia-Petrina
University Babes Bolyai,
Faculty of Mathematics and Computer Science
Cluj-Napoca, Romania
nadejdecamelia26@gmail.com

2st Vescan Andreea
Babes-Bolyai University
Faculty of Mathematics and Computer Science Romania
Cluj-Napoca, Romania
andreea.vescan@ubbcluj.ro

***Abstract*—Predicting software defects has become an important part of research lately. There are many studies regarding this subject. Software defect prediction has been considered a critical issue for each tech industry. Being able to automatize the process of finding defects in software programs would significantly reduce the faults, time and cost.**

The aim of this paper is to investigate various Machine Learning models applied to software defect prediction. The type of Machine Learning implemented was supervised learning for all models. The models used were a variation of three different algorithms. Three methods are used: Naive Bayes, Decision Tree and Random Forest algorithm. The data sets used for this research were from the NASA MDP Repository. The main purpose was to identify the best feature selection and prediction technique for our problem. The Hybrid Feature Selection method revealed the best optimal set of features. All models used in this research were used before and after we applied the Hybrid Feature Selection method. Moreover, we used different prominent evaluation benchmark to evaluate the models performance, such as: Accuracy and Matthew's Correlation Coefficient (MCC). This study reports a significant performance of 98 percents using Random Forest algorithm on three different data sets. Another goal of this study was to compare the results obtained with other results on the same data sets. In both cases, the Random Forest algorithm gave the best results.

***Index Terms*—Software Defect Prediction, Software Bug Prediction, NASA datasets**

I. Introduction

Defects are currently one of the biggest problems encountered in software development. The goal of every company is to create the highest quality software products and to minimize the number of errors in applications. Of course, this is quite difficult when applications have increasingly new features and functionality. No matter how many programmers and testers handle applications, program errors are inevitable. We can ask ourselves what is a defect? This is an error in the software program that will result in an erroneous result.

The problem of error detection has become quite widespread in recent years. As weaving requires time and resources, and they are sometimes limited, ML models are applied to improve the quality of software applications. Predicting software defects is an important thing in the field of software development, because predicting errors in software programs improves quality and thus reduces the maintenance costs and the effort of testers in finding defects. Predicting errors during software development of programs is also a way to prevent other defects in other features of the program.

One of the motivations of this paper is to facilitate and observe the testing methods of the programs, so that the time invested by both testers and programmers during the development of a product is minimized and streamlined.

Despite the difficulties listed, our aim in this study is to evaluate the accuracy of predictions from different ML models and to observe which certain characteristics affect and in what way the prediction of defects. In this paper, we implemented various prediction models to obtain optimal results for detecting software defects. At the same time, the paper aims to identify an optimal and minimal set of software features that could predict the predisposition of software in NASA aerospace systems with improved accuracy, this being done by preprocessing the data sets. Another objective is to compare the results obtained from the analysis on the data sets from NASA MDP Repository with the results obtained in the research paper [1].

The contributions made in this paper are the following:

- A systematic literature review of the existing approaches regarding prediction of software defects.
- The combination and selection of text processing using Naive Bayes, Decision Tree and Random Forest methods to predict the software defects of a program.
- Applying the mentioned methods to three data sets regarding NASA aerospace system.
- Finding the optimal and minimal set of software features that could predict the predisposition of software defects in NASA aerospace system, this were done by applying the Hybrid Feature Selection.
- Comparison of the results obtained with those of the research paper [1], where it was observed that in both papers, the Random Forest algorithm had the best results on the chosen data sets.

The paper is organized as follows: in Section II, the background material relates to software defect prediction. Section III summarizes a Systematic Literature review over related work. In Section IV, we elaborate the approach and the process of implementing the models. The experiments and their results are described in Section V. Conclusions are provided in Section VI.

II. Background to concepts

This section contains the main concepts that we used in our approach, i.e., software defects, algorithms employed as Naive Bayes, Decision Tree, and Random Forest.

A. Software defect

Measuring the performance, reliability, or quality of software describes the sequence of actions taken to detect errors in a software product. Defects found during software development have led researchers to develop various methods of error accuracy. However, predicting errors in a software product is costly in terms of time and resources used. Software failure prediction approaches are more expensive but more effective in detecting software defects compared to software testing.

The purpose of software failure prediction is to reallocate software project resources to modules in projects that require more effort. Another ideal would be to predict software defects ahead of time before the product is delivered.

B. Naive Bayes

Naive Bayes (NB) is an efficient and simple probabilistic classifier based on the Bayes theorem with the assumption of nonindependence between characteristics. NB is not a single algorithm, but a family of algorithms based on a common principle, which assumes that the presence or absence of a certain feature is not related to the presence or absence of any other feature. Bayes' theorem is a simple mathematical formula used to calculate conditional probabilities. Conditional probability is a measure of the probability that an event will occur given that another event occurred (by assumption, presumption, assertion, or evidence) [2].

C. Decision Tree

The Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can also be used to solve regression and classification problems [3].

The purpose of using a decision tree is to create a training model that can be used to predict the class or value of the target variable by learning simple decision rules deduced from training data [4].

D. Random Forest

The premise of the Random Forest (RF) is to build a small decision tree with few features, which is therefore computationally inexpensive. Taking into account weak and small decision trees in parallel, the trees can then be combined to form a strong and single decision tree. In addition, RF is often considered the most accurate Machine Learning (ML) model. Thus, building more trees using the Random Forest algorithm is not the only option, but they will be less correlated, which allows the algorithm to perform better [5].

III. Related work

A systematic search and analysis was performed according to the problem studied, using a general method of dividing the question into individual parts and compiling a list of alternative synonyms and spellings, as in the [6] approach.

A. Systematic Literature Review process

We performed a Systematic Literature Review using Kitchenham's approach [6]. In this SLR we analyzed, using software references, studies conducted in the direction of improving the prediction of software defects.

A systematic search and analysis was carried out according to the problem of software defect prediction using a general method of dividing the question into individual parts and drawing up a list of alternative synonyms and spellings. These terms could be used taking into account the titles of topics used in articles and publications.

First, five key words were defined, relevant for the studied problem: *Software Defect Predict, Software Error Predict, Software Bug Predict, Software Defect Estimate, Software Fault Predict*. The criteria for including articles are as follows: the keywords are in the title, the keywords section or the abstract of the paper, and the paper is published in an international scientific database. We included articles from the following databases:

- https://ieeexplore.ieee.org/Xplore/home.jsp
- https://dl.acm.org/
- https://www.webofknowledge.com

To establish this SLR, the aim was to get an answer to the following questions:

1) RQ1. What kind of data sets are most often used to predict software failure?
2) RQ2. Which Deep Learning models are more effective in predicting software defects?
3) RQ3. What metrics can be included in the software failure prediction algorithm?

B. Report of the Systematic Literature Review

This section contains the analysis and findings of the conducted SLR. After applying the additional filters, 30 items meet the first two filters. After reading the abstract, the articles on the research topic were extracted. Thus, the number was reduced to 14 articles. The last step was to remove duplicates, reaching 10 articles.

There are many studies on Software Defect Prediction that use machine learning techniques. For example, in the study [7] proposes Linear Auto-Regression (AR) to predict defective modules. The study predicts future errors based on historical data of accumulated defects. The study also evaluated and compared the AR model with the power model (POWM) using the RMSE (Root Mean Square Error) measure. The study used three sets of data for evaluation, and the results were good.

R. Malhotra in [8] presented a good systematic review of error prediction techniques using ML. The research included all studies from 1991 to 2013, analyzed ML techniques for prediction models, and evaluated their performance, and assessed the strengths and weaknesses of ML techniques.

Singh and Chug in [9] discussed five popular ML algorithms used to predict software defects, namely: Artificial Neural Networks (ANNs), Particle Swarm Optimization (PSO), Decision Tree (DT), Naive Bayes (NB), and Linear Classifiers (LC).

The study showed important results, including the fact that ANN has the lowest error rate following DT, but LC is a better algorithm in terms of error prediction.

NASA datasets have also been widely used in software failure prediction experiments [10]. There are currently 13 data sets in the NASA Metrics Data Program used to predict software defects. The biggest problem found in NASA's original datasets is when they are used in a machine learning context, there is the possibility of repeating data points that can lead to an invalid experiment by inadvertently appearing training data included in test sets [1], [10].

In Boetticher 2006 [11], five sets of NASA data were used for various classification experiments. The author states that "data processing removes all duplicate data from each data set along with those data that have ambiguous values." The author gives details about the repeated data, stating that "in order to avoid the construction of artificial models, the best approach is not to allow duplicates in the data sets." One of the experiments in this study was to show the effect of duplicate data on the 5 NASA data sets used. This was in the context of the 10-fold cross-validation classification experiment with a C4.5 decision tree. The result claimed that data sets with duplicate values included performed better than those without.

Classification experiments using probabilistic results were performed in Bezerra et al., 2007 [12]. Here the authors used all 13 original NASA datasets and claim to have eliminated both redundant and inconsistent models. Inconsistent data is another problem with NASA datasets. These occur when duplicate metrics describe different class labels. Thus, in this field, they appear where the same set of values is used both to describe the modules labeled as "defective" and for those labeled as "good". The elimination of such cases was first carried out in Khoshgoftaar and Seliya 2004 [13]. In [11], the authors rely on data analysis and preprocessing. The authors' study also demonstrates the poor quality of NASA data sets.

To summarize, there has been very substantial research and effort in predicting software errors. Most of the mentioned works proposed several ML models and different data sets. Some of the previous studies focused mainly on the values that make software failure prediction as effective as possible, while others have proposed different methods for predicting software defects instead of ML techniques.

C. Answers to the Research Questions

Following the SLR study, we can now state the answers to the research questions set at the beginning of this paper.

1) RQ1. What kind of data sets are most often used to predict software failure?
   Most of the summary studies applied the NASA and PROMISE data sets. However, applying many datasets and comparing the results among them is a good indicator of the effectiveness of the models. Public data sets were also used, but they had some issues, such as lack of data quality. Therefore, the results of the proposed model may be inaccurate.
2) RQ2. Which Deep Learning models are more effective in predicting software defects?
   We find that some studies use only one ML model, while other studies use a combination of ML models. Moreover, the most used ML models used to predict software defects were: Convolutional Neural Network (CNN), Artificial Neural Network (ANN), Naive Bayes, SVM and Decision Tree. Moreover, the hybrid models performed better than individual ML models.
3) RQ3. What metrics can be included in the software failure prediction algorithm?
   The metrics are used to compare the proposed models with the other basic models. Most studies applied a single evaluation measure, while other studies used several metrics: AUC, F score, ROC, Accuracy, Average Precision, Precision and Recall. However, using multiple evaluation metrics is better than using a single metric (F or AUC). Several evaluation metrics must be used to validate the applicability of the models and to verify the performance of the models.

IV. Software defect prediction models

The aim of this paper is to exploit three different ML models for defect prediction.

A. Replication study

Experimentation plays an important role in scientific improvement, so replication is an important part of experimental methods, namely, experiments will be repeated to verify the results. For an experiment to be considered a replication, the following elements must be met [14]:

- The authors must explicitly state which original experiment they wish to reproduce,
- validation of the experiment, the way we interpret the results
- both experiments must have common research questions or hypotheses
- the analysis must contain a comparison with the original and new results to conform or refute the initial experiment.

In the article [14], the authors consider that an experimental configuration has four dimensions:

- **Operationalization** - Description act of translating a construct into its manifestation. In a controlled experiment, we find the cause and effect. In the context of our research, the present replication is similar to the original experiment, as similar ML models and the same data sets are used.
- **Population** - There are two types of populations in controlled experiments for which the results should be verified in replication: subjects and objects. If the experiment is conducted with subjects, the replication should study how easily the results are influenced by the properties of the subjects. In the context of our research, the objects of the experiment itself, in our case the data sets, were selected for the experiment. Each ML model was applied

to one data set, using both the entire data set and only certain metrics in it.

- **The protocol** contains the apparatus, materials, experimental objects, forms, and procedures used in the experiment. The protocol is the configuration of all these elements used to observe a certain effect. In the context of our research, three ML models were applied: Naive Bayes, Decision Tree and Random Forest. These were applied to both the entire dataset and only certain features. The best features were selected using the Hybrid Feature Selection method.
- **Researchers** - This dimension refers to the number of people involved in the experiment. In the context of our research, for this dimension, which refers to the people included in the experiment, the researchers are different, the experiment being performed by the authors, compared to those in previous articles that used the same methods.

B. Dataset

Three different datasets are used. All data sets are from NASA MDP Repository. These three data sets contain static code measures, along with their numerical format failure rates. Values are based on product size, complexity, and vocabulary.

The data sets used in this study belong to NASA MDP Repository. NASA MDP is a database that stores problems, products, and data values. The data sets are as follows: PC2, PC3, PC4. The files are in ARFF format. In the Table I the datasets with their number of attributes and instances and the description are provided.

TABLE I
DESCRIPTION OF NASA DATASETS

Dataset	Number of attributes	Number of instances	Description
PC2	37	1493	Dynamic simulator for altitude control systems
PC3	38	1099	Flight software for Earth orbit
PC4	38	1379	Flight software for Earth orbit

In our datasets we have: PC2 - 37 columns, PC3 - 38 columns, and PC4 - 38 columns. We need to decide which are the important features that have a big impact on the data set. The correlation refers to some statistical relationships that involve dependence between variables. A heatmap shows the relative intensity of a value in an array. In Figure 1 it is provided the correlation matrix of the PC2 dataset. The heatmaps were created for all three datasets.

C. Data Pre-Processing

In our dataset, the attributes of each NASA dataset were in ARFF format and investigated with WEKA-Explorer. Missing values were eliminated in the data processing process, which resulted in the data sets being cleaned and ready for use in algorithms.

The authors of the research paper [1] tried to investigate the ability to select features using the HFS (Hybrid Feature Selection) method to extract "good" features for predicting software defects. This step involved executing the HFS method proposed by Ramani and Jacob [15] to try to automate the process of finding the optimal set of metrics, by combining feature selection with a subset of features that are highly correlated with the class and least correlated with each other.

The HFS method was performed on all three data sets, and it was observed that the number of metrics decreased considerably. The number of important metrics and their description can be seen in Table II.

TABLE II
NASA DATASETS BEFORE AND AFTER HFS

Dataset	Total number of Attributes	Number of attributes after HFS	Remaind number of attributes
PC2	37	5	Loc_Comments, Cyclomatic_Density, Halstead_Content, Modified_Condition_Count, Percent_Comments
PC3	38	7	Loc_Blank, Loc_Code_And_Comment, Loc_C,Per_Comments, Halstead_Content, Halstead_Length, Num_Unique_Operands,
PC4	38	4	Loc_Code_And_Comment, Condition_Count, Essential_Complexity, Percent_Comments

An analysis of the related metrics was also done using WEKA Explorer, where $AttributeEvaluator$ was set to $CfsSubSetEval$ and $SearchMethod$ to $GreedyStepwise$. After this analysis, it was observed that in both variants, and by the HFS method, and using WEKA Explorer, the results were similar.

D. Employed Prediction Models

In this study, the following ML models were used to detect software defects:

- Naive Bayes
- Decision Tree
- Random Forest

All these three methods were applied on our data sets, before and after HFS, and then the results obtained were observed. For the evaluation metrics, two were chosen:

- Accuracy
- MCC (Matthew's correlation coefficient)

In order to apply the MCC evaluation metric, it was necessary to implement it. In Listing IV-D it is provided the implementation of the MCC function where we applied the formula of this metric, namely:

$$MCC = \frac{TN * TP - FN * FP}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}}. \quad (1)$$

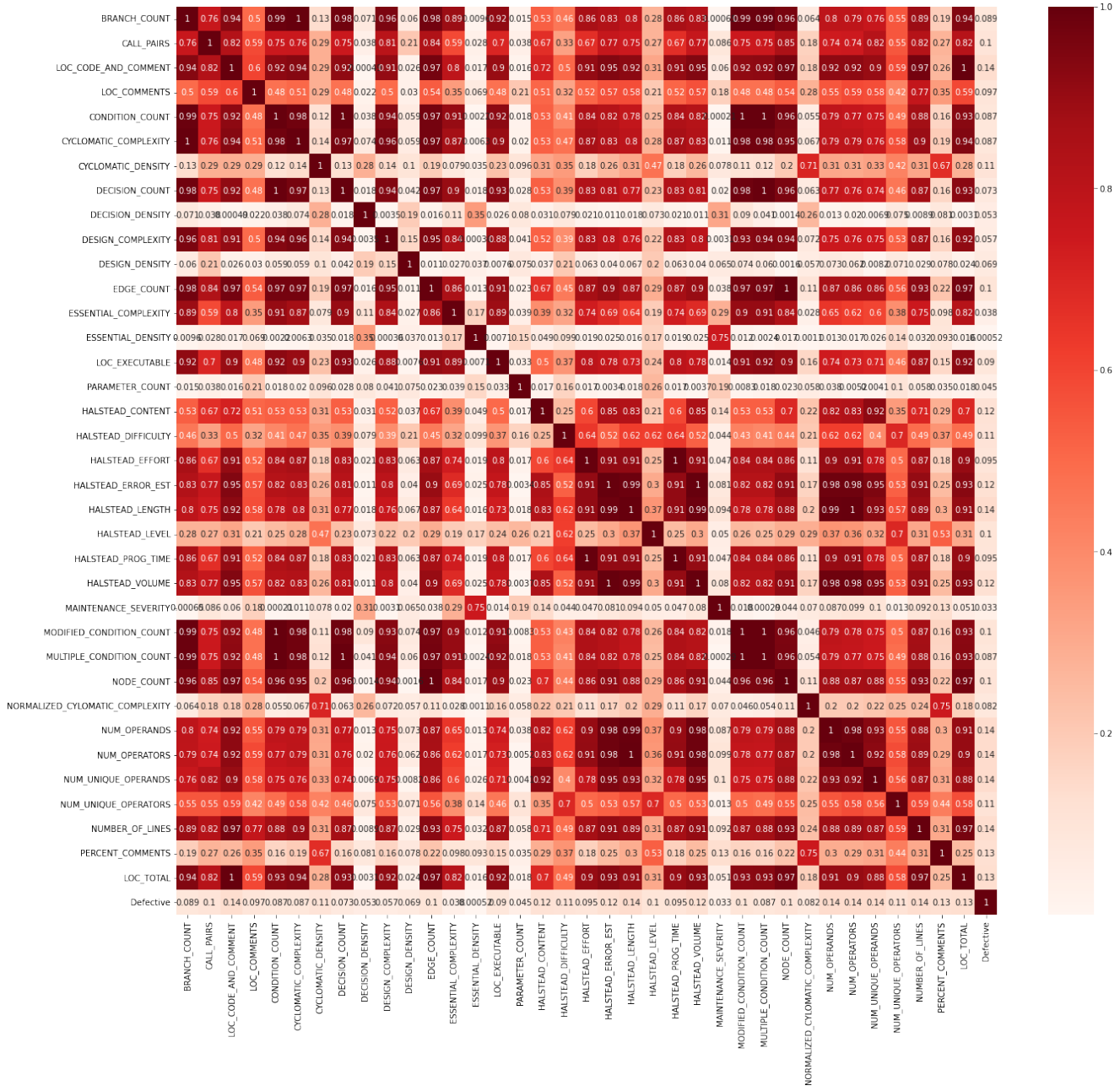

Fig. 1. PC2 dataset correlation matrix

MCC evaluation metrics range from $+1$ to -1:

- $+1$ is the best value between predicted and actual values;
- 0 not a good value. That is, the prediction will be made at random.

Listing 1. Implementation of MCC evaluation metrics

```
def MCC(yt,yp):
    tn, fp, fn, tp = confusion_matrix(yt, yp).ravel()
    den=(tp+fp)*(tp+fn)*(tn+fp)*(tn+fn)
    if den==0:
        den=1
    return (tp*tn-fp*fn)/den**0.5
```

E. Experiments

The selected models (Naive Bayes, Decision Tree, and Random Forest) were applied to both the entire data set and the resulting HFS data set. In the following part, we will discuss the ML models applied and their role in predicting software defects.

All models were imported from the *sklearn* library from Google Colaboratory. We applied a few functions for training and predicting as follows: fit function for our data training, and $predict()$ function, where we will perform the classification on the test vector matrix given by the input variable.

V. Results

In the following, we will observe the accuracy table based on the models applied before and after the application of the HFS method, and we will make a comparison between the results obtained in this paper and the results obtained in the research paper [1].

Once the models have been trained on our data sets, they can be used to make predictions about new datasets. In the case of the models we use, we can conclude based on the data on Table III:

- The Naive-Bayes model had the highest accuracy for the $PC2$ data set, after applying the HFS method: 0.96; and the highest value for the MCC measure was for the $PC4$ data set, after applying the HFS method: 0.40.
- The Decision Tree model had the highest accuracy for the $PC2$ data set, after applying the HFS method: 0.97; and the highest value of the MCC measure was for the $PC3$ data set, after applying the HFS method: 0.30.
- The Random Forest model had the highest accuracy for the $PC2$ data set before the HFS method was applied: 0.98; and the highest value of the MCC measure was for the $PC4$ data set before the HFS method was applied: 0.48.

TABLE III
Model results before and after applying the HFS method.

Dataset	Feature selection	Evaluation Metrics	Naive Bayes	Decision Tree	Random Forest
PC2	EFS	Accuracy	93.30	96.87	98.66
		MCC	0.130	-0.0157	0.0
	HFS	Accuracy	95.53	97.76	98.66
		MCC	-0.020	0.277	0.0
PC3	EFS	Accuracy	61.41	83.95	87.34
		MCC	0.247	0.242	0.200
	HFS	Accuracy	82.71	85.49	87.03
		MCC	0.234	0.308	0.161
PC4	EFS	Accuracy	88.11	85.27	91.47
		MCC	0.344	0.264	0.480
	HFS	Accuracy	90.69	83.97	89.40
		MCC	0.400	0.272	0.422

In Table IV we can observe the results obtained on the same data sets used by us, with the same models. These results were published in the research paper [1].

By comparing our results with those of Shomona Jacob and Geetha Raju, we can draw the following conclusions:

- For the Naive Bayes model, the highest accuracy value was for the $PC2$ data set, after applying the HFS method: 0.95; and for the MCC measure, the highest value was recorded for the $PC4$ data set, after applying the HFS method: 0.40
- For the Random Forest model, the highest value of accuracy was recorded for the $PC2$ data set, after applying the HFS method: 0.98; and for the MCC measure, the highest value was for the $PC4$ data set before the HFS (Hybrid Feature Selection) method was applied: 0.54.

As can be seen from the results, the best values were for the Random Forest model in both cases.

TABLE IV
Model results from the article [1]

Dataset	Feature selection	Evaluation Metrics	Naive Bayes	Decision Tree	Random Forest
PC2	EFS	Accuracy	95.5	99	98.9
		MCC	0.078	0	-0.0
	HFS	Accuracy	95.8	99	98.9
		MCC	0.114	0	0.00
PC3	EFS	Accuracy	32.6	85.4	87.6
		MCC	0.124	0.2	0.275
	HFS	Accuracy	82.4	87.6	87.8
		MCC	0.293	0	0.295
PC4	EFS	Accuracy	87.3	88.6	90.6
		MCC	0.364	0.465	0.543
	HFS	Accuracy	88.6	88.8	89.3
		MCC	0.401	0.36	0.503

Result Comparison Between Original and Replicated Studies. The results of this replication remains consistent with the results of the original study. Therefore we build knowledge to support the original study findings.

This study has validated the result of the original study by using the same models and with the same dataset. The results confirmed the findings in the original study: the Random Forest model obtained the best results.

On the other hand, the knowledge built by this study refers to the optimized values for the metrics to be used in prediction using the HFS method.

VI. Conclusions

Software defect prediction is an essential aspect when delivering a high qualitative software system.

Three Machine Learning models, i.e., Naive Bayes, Decision Tree and Random Forest, were investigated and applied to software defect prediction in this paper. The datasets used for this research were from the NASA MDP Repository. The aim of this paper was twofold: to identify the best feature selection and the best defect prediction model. The Hybrid Feature Selection method revealed the best optimal set of features. Moreover, we used different prominent evaluation benchmarks to evaluate the models performance, such as: Accuracy and Matthew's Correlation Coefficient (MCC).

This study reports a significant performance of 98 percents using Random Forest algorithm on three different data sets. Another goal of this study was to compare the results obtained with other results on the same data sets. In both cases, the Random Forest algorithm gave the best results.

References

[1] S. Jacob and G. Raju, "Software defect prediction in large space systems through hybrid, feature selection and classification," *The International Arab Journal of Information Technology*, vol. 14, no. 2, pp. 208–214, 2017.

[2] T. Arnold, "A Machine Learning Approach for Coreference Resolution," Master's thesis.

[3] E. LEarned-Miller, "Supervised Learning and Bayesian Clasification," september 2011.

[4] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA: MIT Press, 2012.

[5] T.Mitchell, "Discipline of Machine Learning," 2006.

[6] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Tech. Rep., 2007.
[7] L. Zhao, Z. Shang, L. Zhao, A. Qin, and Y. Y. Tang, "Siamese dense neural network for software defect prediction with small data," *IEEE Access*, vol. 7, pp. 7663–7677, 2019.
[8] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing*, vol. 27, pp. 504–518, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494614005857
[9] P. Deep Singh and A. Chug, "Software defect prediction analysis using machine learning algorithms," in *2017 7th International Conference on Cloud Computing, Data Science Engineering - Confluence*, 2017, pp. 775–781.
[10] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "The misuse of the nasa metrics data program data sets for automated software defect prediction," in *15th Annual Conference on Evaluation Assessment in Software Engineering (EASE 2011)*, 2011, pp. 96–103.
[11] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the nasa software defect datasets," *IEEE Transactions on Software Engineering*, vol. 39, no. 9, pp. 1208–1215, 2013.
[12] M. E. R. Bezerra, A. L. I. Oliveira, and S. R. L. Meira, "A constructive rbf neural network for estimating the probability of defects in software modules," in *2007 International Joint Conference on Neural Networks*, 2007, pp. 2869–2874.
[13] T. Khoshgoftaar and N. Seliya, "The necessity of assuring quality in software measurement data," in *10th International Symposium on Software Metrics, 2004. Proceedings.*, 2004, pp. 119–130.
[14] M. Shepperd, N. Ajienka, and S. Counsell, "The role and value of replication in empirical software engineering results," *Information and Software Technology*, vol. 99, pp. 120–132, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584917304305
[15] R. R. and J. S.