Analysis of Ensemble Models for Aging Related Bug Prediction in Software Systems

Shubham Sharma and Sandeep Kumar
Computer Science and Engineering Department, Indian Institute of Technology Roorkee, Uttarakhand, 247667, India

Keywords: Aging Related Bugs, Software Fault Prediction, Software Aging, Stacking, Bagging, Boosting.

Abstract: With the evolution of the software industry, the growing software complexity led to the increase in the number of software faults. According to the study, the software faults are responsible for many unplanned system outages and affects the reputation of the company. Many techniques are proposed in order to avoid the software failures but still software failures are common. Many software faults and failures are outcomes of a phenomenon, called software aging. In this work, we have presented the use of various ensemble models for development of approach to predict the Aging Related Bugs (ARB). A comparative analysis of different ensemble techniques, bagging, boosting and stacking have been presented with their comparison with the base learning techniques which has not been explored in the prediction of ARBs. The experimental study has been performed on the LINUX and MYSQL bug datasets collected from Software Aging and Rejuvenation Repository.

1 INTRODUCTION

The complexity of the software system are growing continuously. It has resulted in increasing the rate of software failure causing undesirable behaviour of the system along with poor services and sometimes complete outages. Dealing with software faults is very important task. Faulty modules present in a software deteriorates the quality of the software and also increases the overall cost of the software system (Menzies et al., 2010).

Various studies (Huang et al., 1995; Trivedi et al., 2000) indicated that in many cases software aging phenomenon is responsible for software failures. Software Aging refers to the phenomenon observed in many software systems, such that it results in increased run time of the system as well as increased failure rate.

It is very necessary to identify the aging phenomenon in the early stage of the software development, in order to prevent the damage and extra cost needed to cope up the software aging effect at the later stages of software development. The sooner these aging related bugs are predicted, the lower it will cost to remove or modify these bugs. The objective of the prediction of these aging related bugs is to make them available to the tester, even before the software testing phase takes place.

Software rejuvenation is the mechanism used to counter the effects of the software aging. It is very effective in nullifying the effect of ARBs (Aging Related Bugs) during runtime phase (Vaidyanathan and Trivedi, 2005; Zhao et al., 2008), but it can be more economical and effective to use them at early stage (Matias and Paulo Filho, 2006). Thus, it is very helpful to predict the aging related bugs as early as possible. Thus, we aim to predict the ARB's in a software system in order to minimize their effect.

We may take necessary preventive measures, but our ability to understand the software and predict its future is uncertain and we can approximate it to certain level only. Over the period of time, we will somehow make certain assumptions that will be inconsistent with our initial assumptions. Preventive measures are effective but cannot eliminate the aging completely (Ahmed, 2016).

Various works such as (Cotroneo et al., 2013; Qin et al., 2015) are available which study the use of various learning models for ARB prediction. In addition, works such as (Ahmed, 2016) have explored the use of some ensemble methods for a learning technique for ARB prediction. However, there is a need to study the behaviour of various ensemble methods for different learning models for ARB prediction. In this work, we have presented the

Sharma, S. and Kumar, S.
Analysis of Ensemble Models for Aging Related Bug Prediction in Software Systems.
DOI: 10.5220/0006847702560263
In *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*, pages 256-263
ISBN: 978-989-758-320-9

analysis of ensemble models to predict aging related bugs, which are responsible for the software aging phenomenon. Evaluation has been performed on the datasets available from software aging and rejuvenation repository (Cotroneo et al., 2013). Preliminarily, we have evaluated various machine learning techniques such as Support Vector Machines, Naïve Bayes, Logistic regression, Multilayer perceptron, K Nearest Neighbour and Decision tree to predict the bugs. These techniques are highly explored in the area of software fault prediction but the study of these commonly used techniques in comparison with their ensemble methods has not been explored for the prediction of aging related bugs in software system. Thus, we have developed ensemble models using bagging, boosting and stacking methods on these machine learning techniques. We have compared these ensemble models with each other and also with the participating base learning models.

The paper is organized as follows: Section 2 presents the related works done regarding software aging. Section 3 describes the work plan followed by us for developing various ensembles for prediction of software aging. Section 4 describes the experimental set up used for performing the experimentation. Section 5 presents experimental results and analysis followed by conclusion in section 6.

2 RELATED WORKS

Andrzejak and Silva (2008) presented an empirical study for prediction of software aging using SVM, Naïve Bayes and decision trees techniques. However, the testing was limited and the authors didn't explore their approach for the cases involving multiple resources and in dynamic environment.

In another work, Cotroneo et al., (2013) tried to find the aging prone source code files and predicted the aging related bugs using predictor variables. Furthermore, Qin et al., (2015) presented a TLAP (Transfer learning based aging bug prediction) approach which provides cross project prediction using transfer learning among different projects. It used the cross project approach, where the dataset of multiple project was used to develop and train a prediction model that can be used to test any of these projects and have shown significant improvement in prediction performance.

In Cotroneo et al., (2011), various works in the field of software aging and rejuvenations were surveyed and it highlighted the aspects that requires special attention and the important trends observed in various experiments performed in software aging and rejuvenation field. It also highlighted the various aging indicators which should be focused during prediction and provide a direction for further study.

It is evident from the works such as Misrili et al., (2011), Wang et al., (2011), Khoshgoftaar et al., (2003) and Aljamann and Elish (2009) that ensemble methods helps in achieving better prediction accuracy in comparison with individual machine learning techniques. But, all these works are used in the field of software fault prediction in order to predict the software modules to be faulty or non-faulty.

But as per the study by Cotroneo et al., (2011), the ensembles methods are rarely used in the field of software aging to find the aging related bugs in the software system and is not fully explored. Thus, it is required to construct and evaluate ensemble methods for finding the aging related bugs. This paper presents the ensemble methods for predicting the aging related bugs in the software system.

3 ENSEMBLE LEARNING BASED ARB PREDICTION

In this section, we present the ensemble based prediction model to predict the aging related bugs. Using this process, various ensemble methods such as bagging, boosting, stacking, etc. are explored to predict the ARB's.

Our main target for prediction is to do prediction at the level of a file with reference to its proneness for ARB. We have to predict whether the file is ARB (Age related bug) prone or ARB-free. Samples are classified into two classes. We are concerned with the presence of ARBs in a file, not their count. Even if file contains one ARB, it is considered as ARB prone, else it is considered as ARB free.

The aging dataset used suffers from the class imbalance problem as the number of bugs belonging to ARB prone class is very less in comparison to ARB free class. So, we perform the class imbalance mitigation procedure to counter the effect of class imbalance problem (Qin et al., 2015). After the class imbalance mitigation procedure, we split the dataset into five parts according to bug proportion and then assign four parts for training the model, while one part is used for testing the model. Each case of experiment is done five times and the mean value of output is used as the final output in order to avoid any occasional error. The final output of the

ensemble model is the mean of five iterations of each case.

The flowchart of the ARB prediction process is shown in Fig. 1.

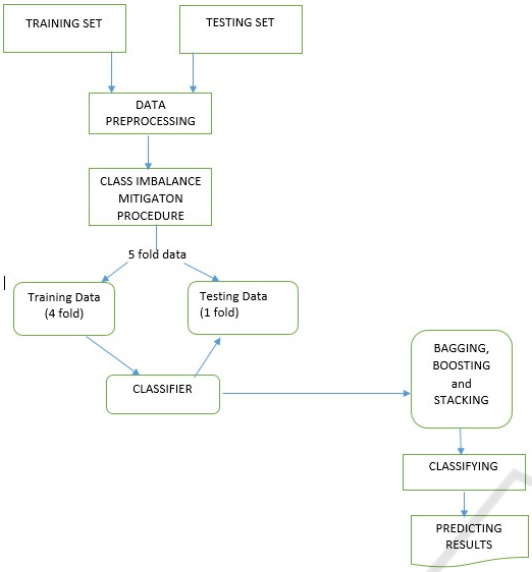

Figure 1: ARB Prediction process.

4 EXPERIMENTAL SETUP

4.1 Background

4.1.1 Machine Learning Methods Used

We have used Naïve Bayes, SVM, Logistic Regression, KNN, MLP and Decision Tree, MLP for the prediction of aging related bugs. These are highly explored machine learning techniques in the area of software fault prediction and are rarely explored in prediction of ARB's. In this section, we have discussed these techniques in brief.

Naïve Bayes: Naïve Bayes classifier is a classification technique, which works on the independence between the predictor variables. It uses Bayes theorem, which finds conditional probabilities among variables. A Naïve Bayes classifier considers various features and checks their contribution independently to the probability of occurrence of any event (Dumais et al., 1998).

Support Vector Machines: Support vector machines (SVM's) are linear classifiers which is generally used for two class problem. Its working is based on margin maximization principle. It classifies the data optimally into two categories by constructing hyperplanes in higher dimensional space. The idea of SVM is to select the hyperplane that separates the two classes with maximum margin from the hyperplane (Adankon and Cheriet, 2009). We have used SVM with radial kernel basis function for this implementation.

Logistic Regression: It is a statistical model which uses fitting of logistic curve to the dataset. It is generally preferred when the outcome variable or target variable is dichotomous. It is used for estimation of discrete values based on the given independent variables. It provides output in terms of probability. Generally, it can be used for both two class problem as well as for multi class problem (Cohen et al., 2009).

K Nearest Neighbour (KNN): It is a classification algorithm, which is non-parametric in nature. It selects an unlabelled dataset point and assigns it to the nearest set of already classified set of points by majority vote of its nearest neighbours. It is well suited for multi-modal classes but can perform on two class problem too. It needs many iterations and is lazy in nature. Its efficiency is dependent on selecting a good value of 'k', where 'k' refers to the number of nearest neighbour points which are used for voting while classifying a data point into one of the classes (Cover, 1968).

Multilayer Perceptron (MLP): Multilayer Perceptron (MLP) or neural network (NN) is a machine learning algorithm that works on the principle of biological neural network (Kotsiantis, 2007). It consist of series of processing layers interconnected, with each connection possessing some weight. During training, based on the knowledge of domain, it develops a representation that maps input space to output space. MLP uses supervised learning technique called back-propagation to train the network.

Decision Tree: Decision tree are the trees that classify the example by sorting them on the basis of their feature value. Decision Tree classifier generates a tree based on C4.5 algorithm (Quinlan, 2014). Each node represents one of the feature of the example to be classified and the branch from that node represents a value that can be assumed by that node. The feature that divides the training examples most appropriately would be used at root node. Similarly at each level, the feature that best divided the training data represents the node at that level (Kotsiantis, 2007).

4.1.2 Ensemble Methods Used

The ensemble methods used to predict the aging related bugs are Bagging, Boosting and Stacking. In this section, we have discussed these methods in brief.

Bagging: Bagging uses Bootstrap Aggregating algorithm for classification process (Breiman, 1996). In bagging, various classifiers used as members of ensemble are constructed from different training datasets, and the combined prediction is usually the uniform averaging or voting over all the members. Each classifier is trained on sample training example take from training dataset with replacement such that size of each sample equals the size of actual training dataset.

Boosting: Boosting is one of the popular ensemble technique, which generally uses AdaBoost algorithm (Freund and Schapire, 1996). It uses sequential training of models and performs training using multiple iterations, with a new model used for training in each iteration. It constructs an ensemble by using different example weights at different iterations. The prediction output is combined using voting on output of each classifier.

Stacking: Stacking is an ensemble technique which needs two level of models to construct an ensemble (Wolpert, 1992). These are base models, also called as level-0, and meta-model or level-1 model. The level-0 models uses bootstrap samples of the training dataset, and the output of these base models are fed as input to a level-1 model. The meta-model aims to classify the target correctly by combining the outputs of base models and correcting the mistakes made by base models.

4.1.3 Evaluation Metrics

There are several performance measures to evaluate the performance of the classifier and some of the most common performance measures are: Accuracy, Recall, F-measure, and Precision.

For classification problem, with data points belonging to two classes positive and negative class, TP defines Positives correctly predicted, TN- Negatives correctly predicted, FP- Positive wrongly predicted as Negative and FN – Negative falsely predicted as positive.

Accuracy – Accuracy performance measure calculates the number of data points correctly predicted. It is calculated by the equation-1;

$$Accuracy=(TP+TN)*100/(TP+FN+FP+TN) \quad (1)$$

Recall (True Positive Rate (TPR)) – It calculates the fraction of positive class data points predicted as positive. The equation-2 is used to calculate it;

$$Recall(TPR) = TP / (TP + FN) \quad (2)$$

Precision- Precision is the fraction of predicted positive class that is actually positive class. It is calculated by equation-3:

$$Precision = TP / (TP + FP) \quad (3)$$

F-measure - F-measure depends on recall and precision and is defined by equation-4:

$$F\text{-}measure=2*Prec.*TPR/(Prec.+TPR) \quad (4)$$

4.2 Class Imbalance Mitigation Procedure

Class imbalance is one of the major challenges and it represents the cases where the examples in one class is very less as compared to other classes. The class with higher size of data is majority class, while the class with smaller size is considered as minority class.

Minority class is unable to draw attention of classifiers in comparison to the majority classes.

Let us assume that D is a dataset with N samples and out of it a part of dataset $D_{min,}$ includes the examples belonging to the minority class (ARB prone class in our case) with size N_{min}, and another part of dataset, D_{max} contains the N_{max} examples belonging to the majority class (ARB free classes).

We calculate $R = N_{max} / N_{min}$, termed as size ratio to calculate the relation between the size of minority and majority class. Consider that O_{min} and O_{max} are the sizes of the parts of datasets after the mitigation process. During mitigation, we keep the majority class as it is, such that size of O_{max} equals N_{max} while we change the minority class by enlarging it such that size of $O_{min} = R * N_{min}$. Thus, both the majority and minority classes are now equally attended and considered by the classifier (Qin et al., 2015).

4.3 Dataset and Tools Used

An overview of software aging datasets are shown in Table 1 (Cotroneo et al., 2013) above. We have used two real software system datasets: Linux and MySQL. We have performed analysis on four subsystems of Linux project and three subsystems of MySQL project as described above in Table 1 (Cotroneo et al., 2013).

Table 1: An overview of software aging datasets.

Project	Subsystem	ARB's	Files	ARB- prone files	% ARB-prone files
Linux	Network Drivers	9	3400	20	0.59%
	Other Drivers	4			
	Filesystem	5			
	IPv4	2			
MySQL	Storage Engine	6	470	39	8.3%
	Duplicate files	5			
	Optimization files	5			

The datasets consists of aging-related metrics (Huang et al., 1995) namely AllocOps, DeallocOps, DerefSet and other memory usage related metrics (Cotroneo et al., 2013) along with Halstead and McCabe complexity metrics. Tai et al., (1997) provide the details about all these metrics.

Tools Used:

All the implementation in this work have been done in Python programming language version 3.5.0.

5 EXPERIMENTAL RESULTS

In this section, we have presented experimental results by performing ARB prediction by using SVM, KNN, Decision Tree, Multi-Layer Perceptron (MLP), Logistic Regression and Naïve Bayes and subsequently by applying stacking, bagging and boosting based ensemble models on each of these learning models. The comparative analysis and the conclusive observations are also discussed. In this section, we have presented experimental results by performing ARB prediction by using SVM, KNN, Decision Tree, Multi-Layer Perceptron (MLP), Logistic Regression and Naïve Bayes and subsequently by applying stacking, bagging and boosting based ensemble models on each of these learning models. The comparative analysis and the conclusive observations are also discussed.

In this experimentation, AdaBoost algorithm (Freund and Schapire, 1996) is used to model the boosting based ensemble method. Also, for implementing bagging and stacking approaches, we have used bootstrap aggregating algorithm (Breiman, 1996) and stacking algorithm (Wolpert, 1992) respectively.

Tables 2,3,4,5 shows the detailed result of the experiments performed in terms of accuracy, recall, precision and f-measure.

Table 2 shows the detailed analysis of machine learning algorithms such as SVM, Decision Tree, KNN, MLP, Naïve Bayes and Logistic regression. SVM performs best in terms of accuracy and recall, while logistic regression is performing worst. In terms of precision and f-measure, decision tree performs best, whereas Naïve Bayes shows the worst performance.

Table 3 contains the detailed analysis of Bagging based ensembles.SVM performs best in terms of accuracy, recall, precision and f-measure. Logistic Regression performed worst in terms of accuracy and recall, while Naïve Bayes performed worst in terms of precision and F-measure.

Table 4 shows the detailed analysis of boosting based ensembles Decision tree performs best in terms of accuracy, recall, precision and f-measure and SVM performs worst in every performance parameter.

Table 5 shows the performance of stacking based ensemble performed on top three performing learning models: SVM, KNN and Decision tree. The combination stack of SVM, KNN and Decision tree outperformed other ensembles in terms of accuracy, recall, precision and f-measure.

From the analysis of these results, following observations are drawn:

1. The accuracy for the used machine learning techniques are generally high (generally >0.75). TNR is also quite high. Recall is comparatively low in range of 0.57-0.96.
2. The classification accuracy of SVM is very high (>0.95 in most of the cases) and it predicts best as compared to other classification techniques.
3. The ensemble learning methods improve the prediction accuracy of the machine learning algorithms in almost all the cases for ARB prediction as shown in figure 3 and 4.
4. The performance of the weak learners such as Logistic regression, Naïve Bayes and MLP is significantly improved by the useof ensemble models as shown in figure 3 and 4.

Table 2: Comparative analysis of different classifiers.

Technique used	Accuracy	Recall	Precision	F-measure
SVM	0.9891	0.9892	0.8461	0.8523
Naïve Bayes	0.6894	0.8496	0.4620	0.5994
Logistic Regression	0.6296	0.5789	0.9823	0.7320
Decision Tree	0.9807	0.9654	0.9545	0.9484
Multi-Layer Perceptron	0.7173	0.6476	0.9962	0.7849
KNN	0.9193	0.8545	0.9509	0.8882

Table 3: Comparative analysis of different classifiers using Bagging approach.

Technique used	Accuracy	Recall	Precision	F-measure
Bagging on SVM	0.9961	0.9952	0.9937	0.9961
Bagging on Naïve Bayes	0.7364	0.8550	0.5570	0.6731
Bagging on LR	0.6746	0.5985	0.9512	0.7121
Bagging on Decision Tree	0.9960	0.9922	0.9845	0.9960
Bagging on KNN	0.9712	0.9556	0.9862	0.9706
Bagging on MLP	0.7476	0.6368	0.9112	0.7758

Table 4: Comparative analysis of different classifiers using Boosting approach.

Criterion	Accuracy	Recall	Precision	F-measure
Boosting on SVM	0.5661	0.5652	0.6137	0.5561
Boosting on Naïve Bayes	0.7990	0.7850	0.7870	0.7731
Boosting on LR	0.5646	0.5385	0.8852	0.5721
Boosting on Decision Tree	0.9757	0.9632	0.9889	0.9758
Boosting on KNN	0.9244	0.8448	0.9459	0.8912
Boosting on MLP	0.7194	0.6884	0.9084	0.7842

Table 5: Comparative analysis of different classifiers using Stacking approach.

Criterion	Accuracy	Recall	Precision	F-measure
Stack of SVM	0.9881	0.9622	0.9871	0.9861
Stack of KNN	0.9771	0.9785	0.9855	0.9731
Stack of DT	0.9655	0.9641	0.9489	0.9717
Stack of (SVM+KNN+DT)	0.9889	0.9778	0.9685	0.9888

Based on this experimental analysis, the following research questions can be answered as below:

RQ1: How do ensemble methods perform in the prediction of aging related bugs?

Answer: It has been found that discussed ensemble methods shows good prediction accuracy, precision, and recall for almost every dataset.

RQ2: How do ensemble methods perform in comparison to basic machine learning techniques?

Answer: When compared with the individual learning techniques, we found that ensemble methods have either performed better or at least comparable with that of individual base learning techniques in every parameter.

RQ3: How does the ensemble methods affect the performance of the learning models?

Answer: It is observed from the experimental study that ensemble methods improved the performance of most of the base learners. Weak learners like Naïve Bayes, Logistic regression and MLP show significant improvement, while SVM, KNN and Decision Tree performed same or have shown little improvement in every parameter.

RQ4: Are the accuracies of various used ensembles comparable?

Answer: From Fig. 3 and Fig. 4, it can be observed that the accuracy obtained from the various ensembles are comparable with each other.

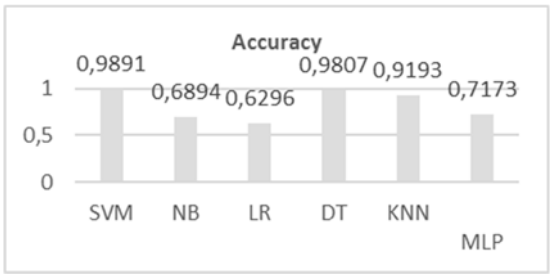

Figure 2: Prediction Accuracy for various classifiers used.

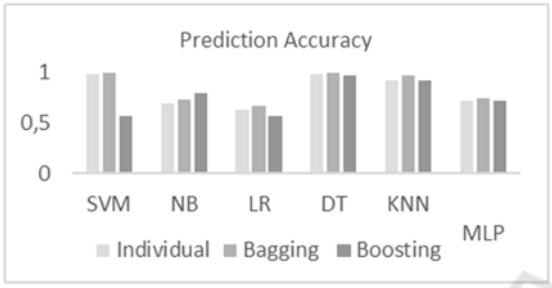

Figure 3: Comparison of Prediction Accuracy of Bagging and Boosting ensemble with Base Learning Techniques.

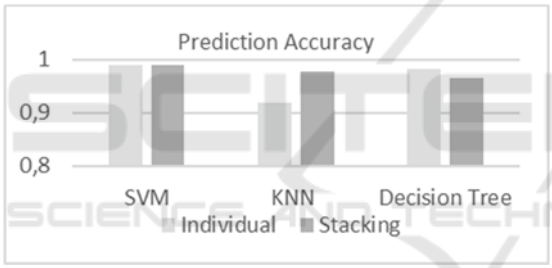

Figure 4: Comparative analysis of performance of base learning techniques with stacking approach.

6 CONCLUSIONS

In this paper, we have investigated the use of some machine learning techniques and ensemble methods such as bagging, boosting and stacking for the prediction of ARBs by building a classifier from the examples of ARBs; and then using it to classify new files as “ARB prone” or “ARB free”. The study was performed on the LINUX and MYSQL bug datasets available in IEEE software aging and rejuvenation repository. For this purpose, we have used six highly explored machine learning techniques such as SVM, KNN, Decision Tree, Naive Bayes and Logistic Regression and evaluated their performance along with their ensembles. This approach have produced significant results in software fault prediction, but not explored in prediction of ARB’s. The experimental analysis highlights that the use of ensemble models is effective in improving the prediction accuracy of the base learning techniques and have significant improvement with prediction accuracy close to 0.99 in some cases. The high accuracy using ensemble methods can be helpful in identifying ARB’s accurately at early stages and can reduce the cost and damage caused due to software aging. In the future, we can evaluate the performance of these ensembles with different combination rules.

ACKNOWLEDGEMENTS

This publication is part of the R&D work undertaken in the project under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

REFERENCES

Adankon, M. M. and Cheriet, M. (2009). Support vector machine. In *Encyclopedia of biometrics*, pages 1303–1308. Springer.

Ahmed, A., (2016). Predicting software aging related bugs from imbalanced datasets by using data mining techniques. In *IOSR Journal of Computer Engineering (IOSR-JCE)*, volume 18, pages 27-35.

Aljamaan, H. I. and Elish, M. O. (2009). An empirical study of bagging and boosting ensembles for identifying faulty classes in object-oriented software. In *Computational Intelligence and Data Mining, 2009. CIDM’09. IEEE Symposium on*, pages 187–194. IEEE.

Andrzejak, A. and Silva, L. (2008). Using machine learning for non-intrusive modeling and prediction of software aging. In *Network Operations and Management Symposium, 2008*, pages 25–32. IEEE.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Cohen, I., Chase, J. S., Goldszmidt, M., Kelly, T., and Symons, J. (2004). Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *OSDI*, volume 4, pages 16–16.

Cotroneo, D., Natella, R., and Pietrantuono, R. (2013). Predicting aging-related bugs using software complexity metrics. *Performance Evaluation*, 70(3):163–178.

Cotroneo, D., Natella, R., Pietrantuono, R., and Russo, S. (2011). Software aging and rejuvenation: Where we are and where we are going. In *Software Aging and*

Rejuvenation (WoSAR), 2011 IEEE Third International Workshop on, pages 1–6. IEEE.

Cover, T. M. (1968). Rates of convergence for nearest neighbor procedures. In *Proceedings of the Hawaii International Conference on Systems Sciences*, pages 413–415.

Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155. ACM.

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156. Bari, Italy.

Huang, Y., Kintala, C., Kolettis, N., and Fulton, N. D. (1995). Software rejuvenation: Analysis, module and applications. In *Fault-Tolerant Computing, 1995. FTCS-25. Digest of Papers., Twenty-Fifth International Symposium on*, pages 381–390. IEEE.

Khoshgoftaar, T. M., Geleyn, E., and Nguyen, L. (2003). Empirical case studies of combining software quality classification models. In *Quality Software, 2003. Proceedings. Third International Conference on*, pages 40–49. IEEE.

Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24.

Matias, R. and Paulo Filho, J. (2006). An experimental study on software aging and rejuvenation in web servers. In *Computer Software and Applications Con ference, 2006. COMPSAC'06. 30th Annual International*, volume 1, pages 189–196. IEEE.

Menzies, T., Milton, Z., Turhan, B., Cukic, B., Jiang, Y., and Bener, A. (2010). Defect prediction from static code features: current results, limitations, new approaches. *Automated Software Engineering*, 17(4):375–407.

Mısırlı, A. T., Bener, A. B., and Turhan, B. (2011). An industrial case study of classifier ensembles for locating software defects. *Software Quality Journal*, 19(3):515–536.

Qin, F., Zheng, Z., Bai, C., Qiao, Y., Zhang, Z., and Chen, C. (2015). Cross-project aging related bug prediction. In *Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on*, pages 43–48. IEEE.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Singh, A., Thakur, N., and Sharma, A. (2016). A review of supervised machine learning algorithms. In *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on*, pages 1310–1315. IEEE.

Tai, A. T., Chau, S. N., Alkalaj, L., and Hecht, H. (1997). On-board preventive maintenance: Analysis of effectiveness and optimal duty period. In *Object-Oriented Real-Time Dependable Systems, 1997. Proceedings., Third International Workshop on*, pages 40–47. IEEE.

Trivedi, K. S., Vaidyanathan, K., and Goseva-Popstojanova, K. (2000). Modeling and analysis of software aging and rejuvenation. In *Simulation Symposium, 2000.(SS 2000) Proceedings. 33rd Annual*, pages 270–279. IEEE.

Vaidyanathan, K. and Trivedi, K. S. (2005). A comprehensive model for software rejuvenation. *IEEE Transactions on Dependable and Secure Computing*, 2(2):124–137.

Wang, T., Li, W., Shi, H., and Liu, Z. (2011). Software defect prediction based on classifiers ensemble. *Journal of Information & Computational Science*, 8(16):4241–4254.

Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.

Zhao, L., Song, Q., and Zhu, L. (2008). Common software aging-related faults in fault-tolerant systems. In Computational *Intelligence for Modelling Control & Automation, 2008 International Conference on*, pages 327–331. IEEE.