Classification of Software Failure Incidents Using SVM

Islam Zada[*], Inayat Khan[†], Taj Rahman[‡], Abid Jameel[§]

Abstract

Software failure in an operational environment can put the performance and quality of service at risk. This research is particularly on software failure incidents. The study involves the basic software engineering process: Classification of software failure incidents through machine learning techniques. The active learning approach is used, which is applied to label only those data which is most in-formative to build models. From all the samples, the sample with higher entropy (randomness) is chosen for labeling. Given a set of labeled observations, we used a classifier that decides the target class label, either "failure" or "no failure". As a classifier mechanism, Support Vector Machine (SVM) is used to classify the data.

Keywords: incident, machine learning, active learning, SVM

Introduction

Software failures can be understood from the two approaches software-centric and system-centric. According to (Chu, Lehner, Martinez-Guridi, & Yue, 2006), the software-centric methodology views "failure" as a property of the software itself (Ogheneovo, 2014). This approach considers the failure only associated with the software itself and has no impact on the entire system. While according to the system-centric approach, software failure not relates to the software itself but also to the entire system. Thus, in the light of these two approaches, failure can be defined as the "failure in the system to perform the required function".
Despite all the efforts, many of the errors remain in the software delivered to the customer, resulting in the software failure. These failures affect users, clients, and the operation handling staff, leading to long-time unavailability of the system, maintenance cost, and low quality. In recent years, many failures occur due to which many people lost their lives, and

[*] Department of Computer Science & Software Engineering, International Islamic University Islamabad, Pakistan. islam.zada@iiui.edu.pk

[†] Department of Computer Science, University of Buner, Buner 19290, Pakistan. inayat_khan@uop.edu.pk

[‡] Qurtuba University of Science & Technology Peshawar, Peshawar 25000, Pakistan. drtaj@qurtuba.edu.pk

[§] Department of Computer Science & Information Technology, Hazara University Mansehra, Pakistan. abidjameel717@gmail.com,

many organizations lost their reputations and suffered a capital loss. For these reasons, recent research investigated the nature of software failures and countermeasures against them.

As noted in (George Box, 2010), discovering the unexpected is more important than confirming the known. Thus, in software development, the "unexpected" one relates to defects. These defects, when unattended, often cause failure (Anwar et al., 2021; Chillarege et al., 1992; Mann, 2002). Thus, software failure is the software's inappropriate behavior and the software's incompetence to conform to the user requirements.

The failure is the software's lack of capability to perform its intended function within a certain environment. To a user, failure is the termination of his/her required function. The user may decide to identify the various levels of failure, such as catastrophic, which could be major or minor, depending on their impacts and consequences on the system, such as monetary value (cost), human life, and property lost (Khan, Ali, & Khusro, 2019; Ogheneovo, 2014). Failures are incorrect external events. In this era, software is inescapable; everyone doing a job in any department or the student studying anything comes in touch with this product. People are attached to it but don't realize it until they face any given problem. The software itself is not easy to develop and maintain, but the most challenging task is to understand, especially for the people who are not IT related. It is among the most labor-intensive, multifaceted, and error-prone technologies in human history (Kumaresh & Ramachandran, 2012). Dijkstra (Dijkstra, Dijkstra, Dijkstra, & Dijkstra, 1976) notes that "the average computer user has been served so poorly that he expects his system to crash all the time, and we witness a massive worldwide distribution of bug-ridden software for which we should be deeply ashamed". Given a system or software, it would be valuable to define its behavior to help set up the rightness of the applications that keep running on it. On the off chance that this forces confinement on the reasonable conduct of the applications, we should see how these limitations can be implemented and the suggestions in debilitating or strengthening them. A helpful strategy for building such a formal depiction regarding adaptation to non-critical failure is to sort the system components as indicated by the sorts of faults they are expected to display. Four conceivable classifications of failures are Omission, Value, Timing, and arbitrary. The reaction from a given segment for given information will be thought to be right if the yield esteem is right and that the yield is delivered on time, i.e., created inside a predefined time restrain. Failures are basically due to system complexity, insufficient testing and/or poor understanding of system dependencies, even if the site owner is unaware of the root causes

of the failures. Other significant software failure causes are system overload, resource exhaustion, and complex fault recovery routines (Gray, 1990; Khan, Khusro, Ali, & Din, 2016). Common causes of software failures are the deficiency of clear specifications and objectives, poor management, poor communication among designers and customers, use of new technology, lack of experience of the designers. Software availability can be enhanced by predicting the software downtime and the errors and faults that lead to software failure. Researches on the prediction of software failure highlighted many causes of software failure. Among them, software aging has gained more attention. According to Hoffman (Hoffmann, Salfner, & Malek, 2011), software aging describes the misbehavior of software that does not cause the component to fail immediately. Software aging does not lead directly to the failure of the component but enhances the chances of system software system failure at a whole. (Garg, Puliafito, Telek, & Trivedi, 1998) proposed a time series-based model for the recognition of software aging.

The current section is about a brief introduction related to software incidents. Section two present the related work done by different authors. The third section is about the detailed methodology of this study, while the fourth section is about the detailed discussion. The conclusion and future work is presented in section fifth and sixth, respectively.

Related Work

During the last few decades, efforts made to predict failure are remarkable. Failures or failure classification is a broad concept in software engineering and is not just limited to software failure. Failure classification techniques are equally common in both hardware and software. In hardware (e.g., satellite (Meneely & Williams, 2012), cluster computing systems (Li & Lan, 2006), distributed mission-critical systems (Li & Lan, 2006), and telecommunication systems (Baldoni, Lodi, Montanari, Mariotta, & Rizzuto, 2012) these techniques are widely discussed in the literature. But with the increase in complexity of the software systems and the high demand of their failures have been largely shifted to the software (Salfner & Tschirpke, 2008). H. Taherdoost and (Salfner, Lenk, & Malek, 2010) conducted a survey to analyze the failure and the success causes of the different information technology projects. They conducted the survey involving the different factors directly or indirectly related to the causes of failures such as the people, processes, and the technical and non-technical. Large margin classifiers such as SVM classify data using the most useful data points. This makes them natural candidates, for instance, selection strategies. Their underlying basic algorithms describe active learning current methods. Kamal Nigam (Tong & Koller, 2001)

introduced an algorithm using the EM and the naïve Bayes classifier using both the label and the unlabeled data. Lewis (Lewis & Gale, 1994) used the uncertainty sampling method with the Bayesian network and the logistic regression. Here the uncertainty sampling goal was to reduce the folded size while maintaining the accuracy of the results. Colin Cambell (Campbell, Cristianini, & Smola, 2000) also uses the instance selection to train the large margin classifier, the Support Vector Machine (SVM). (Xu, Yu, Tresp, Xu, & Wang, 2003) used active learning for text classification. Representative sampling was purposed for selecting the instances based on the similarity and the informativeness along with the SVM classifier for the text classification. (Lu & Cukic, 2012) used active learning to predict failure and the Naïve-Bayes classifier.

The combination of both approaches opened a new window in this field of prediction. However, due to the computational infeasibility of the naïve-Bayes classifier, this was not the best combinational approach. (Guerra et al., 2011) conducted a study comparing the different machine learning approaches and showed the results of these supervised and unsupervised experiments such as naïve Bayes, K-nn, PCA, etc. Their results clearly show that unsupervised learning couldn't perform well in the case of prediction purposes.

In this research, the classifier SVM is chosen because of its success in classifying the most informative instances. SVM is the large margin classifier which has made it a strong candidate for the instance sampling strategies. SVMs can successfully be applied to solve prediction problems to examine the likelihood (Cao, 2003; Tay & Cao, 2001). Through experiments proved that the SVM outperformed when compared to the BP neural network in terms of the normalized mean square error (NMSE) and mean absolute error (MAE). (Mohandes, Halawani, Rehman, & Hussain, 2004) found that based on the wind speed data, the hourly speed of the wind can be better predicted with the SVM and the Backpropagation model, (Mohandes et al., 2004) investigated the DNA repair pathways and based on the SNP data using the SVM classifier for the prediction of the oral cancer risks. Their results showed that SVM Outperformed both in terms of precision and measure when compared to the other classifiers. Time series ARMA model built to detect the aging and forecast the resource exhaustion times.

Methodology

This study proposed a model for software failure incident classification using active learning and the SVM. Active learning is performed on the dataset that reduced the dataset size and selected the sample out of it which we used as the training set for the SVM classifier.

The selected sample consists of the instances that were unique and informative for training the classifier. Active learning is performed with the clustering technique in this study. To feed the active learning process, k-mean clustering is performed in our approach clustering. Data is first clustered with k-mean clustering strategy, and the clusters representatives were used for the labeling. These instances in the center of the clusters were collected and were manually labeled. This labeled data was used as the training set of the classifier SVM and classification is carried out on this data. The training set obtained after the clustering seemed to be free of any repetitive data and the instances carrying no valuable information. Clustering here in this study was kept constant, and label propagation was not performed given in figure 1 and 2.

The machine learning method used for active learning are many including the SVM (Yousafzai et al., 2021). This method is the most widely used method of machine learning, which has a strong theoretical background. It showed significant performance in image retrieval, text categorization, and handwritten digits recognition. SVM classifiers are most well-suited for active learning due to their appropriate mathematical properties. The goal of SVM is to discover a decision rule with great simplification Capability through choosing some specific subset of dataset called support vectors. In this strategy, an ideal isolating hyperplane is built, mapping the information space nonlinearly into a higher dimensional feature space. Support vector machine is given the labeled data consisting of the two different classes, and then the classifier generates results for that labeled data. The results of the labeled data can be generalized to the unlabeled data.

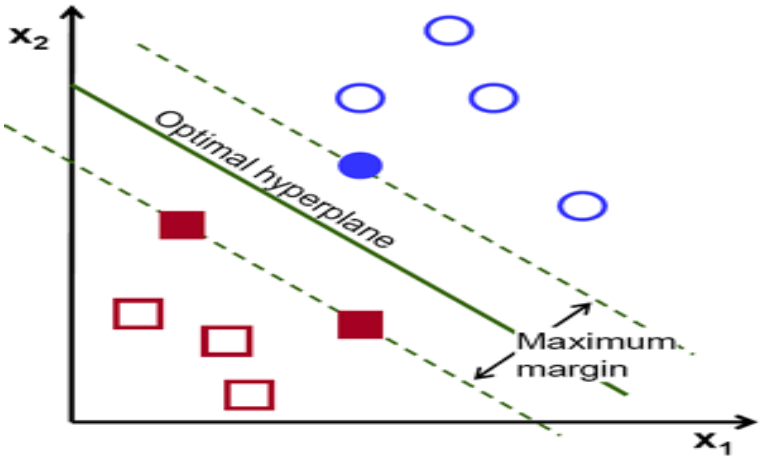

Figure 1.Binary classification, SVM

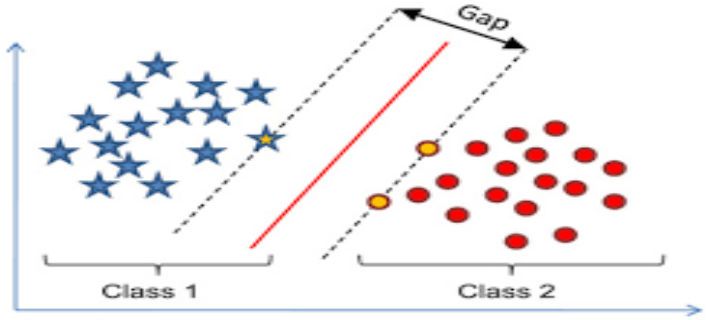

Figure 2. Binary classification, SVM

Kernel tricks make it possible to handle diverse data easily. Support vector machines use the Kernal trick to maximize the distance between the support vectors and hyperplane. Weight functions of the SVMs are inspected to identify the features that significantly influence the whole data set. Those features are then considered to have a significant impact on the classification.

With the success of the SVMs the researchers moved a step ahead and added the extension to the SVM classifier in the form of the capability of solving the multiclass problems depicted in figure 3 and figure 4. Multiclass is the term that means every instance is assigned, one class. In other words, it can be said mutually exclusive. There are several different ways proposed in the literature on how to perform multiclass classification with the SVM. There are two basic ways: Treat the whole dataset as once with all the classes or perform the binary classification on the chunks of the data and then combine the outputs.

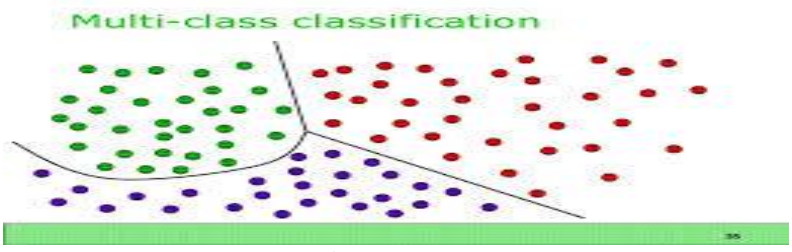

Figure 3.Multiclass classification, SVM

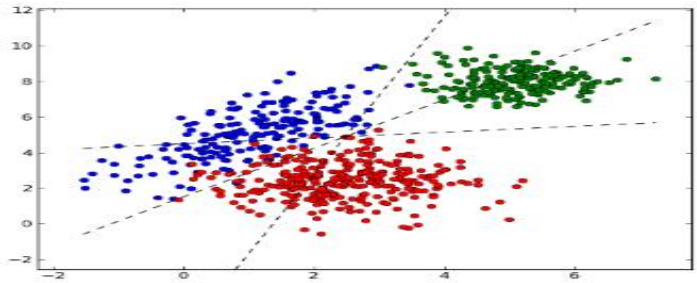

Figure 4. Multiclass classification, SVM

Software packages for implementing the Support Vector Machines are easily available and can be downloaded freely. This research is prediction-oriented. Software package capable of implementing the both with same success rate was not easy to choose so many of the software packages were observed for prediction problems. Their capabilities, accuracy, and speed were tested before choosing one of them. The mentioned packages were evaluated:

Libsvm, Svmtorch, Sequential Minimal Optimization (SMO).

In this research the SVM package used for the classification is the ***SMO*** due to its benefits over other packages such as SMO requires no additional framework stockpiling by any means. In this way, extensive SVM preparing issues can fit within the memory of a common PC or workstation. Since no lattice calculations are utilized as a part of SMO, it is less vulnerable to numerical accuracy issue.

Discussion

The proposed model showed 84% accuracy and classified most of the instances correctly while just incorrectly classifying the two. The test model figure 4.13 is the detailed accuracy measure of the model with different terms such as f-measure, precision etc. Before discussing the metric, some terms need to know, such as the True positive Rate, FP Rate, Precision, Recall, F-Measure, MCC, ROC Area, and PRC Area.

True positive (TP)

Values, which are observed positive, and are predicted positive. In our model the TP rate for "failed" level is "1.000" for status-ok "0.5", for error "1.00", and for the warning its "0.833".

False positive (Fp)

Observed value is negative, but the prediction result is positive. In our case its "0.08" for failure level, "0.111" for error level.

Precision

Precision is the measure of the correctly identified positive events divided by the total predicted events. For failure, it is "0.5", status-ok "1.000", for error the precision rate is "0.8", and for warning its "1.00" given in figure 5.

Precision = True Positive /True Positive +False Positive

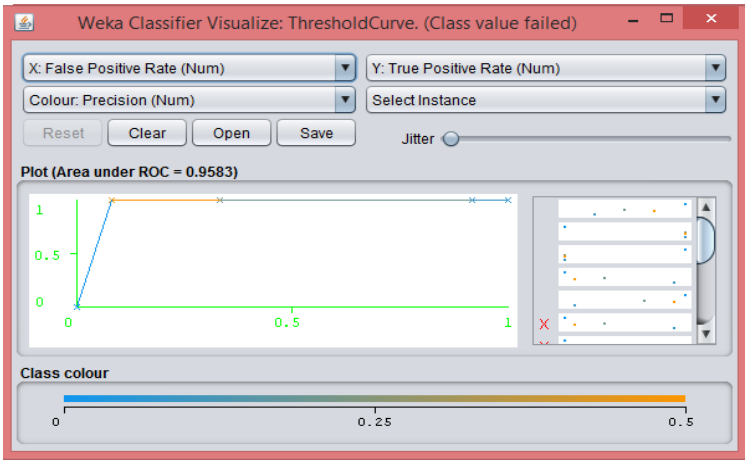

Figure 5. Failure Precision (Plot under ROC)

Recall

Recall is the correctly forecasted positive values divided by all observations. Recall, as mentioned, is the ratio of the true positive observation and the total observation in our case the. For failure it is "1.000", status-ok "0.5", for error the recall is "1.000", and for warning its "0.833".

Recall = True positive / True positive +False Negative

F-measure

This the average of both precision and recall. In our model for failure it is "0.667", status-ok "0.677", for error the f-measure is "0.84", and for warning its "0.85" mentioned in figure 6 and table 1.

F measure = 2(Recall x Precision) / (Recall x Precision)

Table.1

Detailed Accuracy

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.083    0.500      1.000    0.667      0.677    0.958     0.500     failed
                 0.500    0.000    1.000      0.500    0.667      0.677    0.977     0.833     status ok
                 1.000    0.111    0.800      1.000    0.889      0.843    0.944     0.800     error
                 0.833    0.000    1.000      0.833    0.909      0.854    0.917     0.910     warning
                 0.000    0.000    0.000      0.000    0.000      0.000    ?         ?         critial
Weighted Avg.    0.846    0.041    0.900      0.846    0.847      0.810    0.938     0.833
```

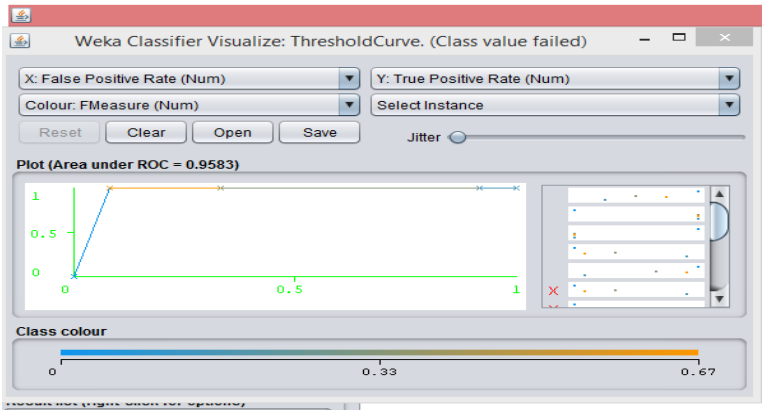

Figure 6. Failure F-measure (Plot under ROC)

Confusion Matrix

The confusion matrix or the error matrix is the visualization of the technique's performance or can say, the algorithm used. Rows contain the predicted instances, and the columns represent the actual instances mentioned in table 2.

Table.2.

Confusion Matrix

```
=== Confusion Matrix ===

 a b c d e   <-- classified as
 1 0 0 0 0 | a = failed
 0 1 1 0 0 | b = status ok
 0 0 4 0 0 | c = error
 1 0 0 5 0 | d = warning
```

Conclusion

In general, every failure is essential regarding both security and cost. Enhanced and improved maintenance schedules can be acknowledged with forecasting techniques. Failures prediction helps in predicting the maintenance times, which counteract both security richness and diminish costs. This study proposed a model for predicting failures using the machine learning methods clustering and the classification of the selected instances through the *support vector machine (SVM).* SVM calculation is capable of anticipating, yet it is not clear to decide the parameter esteems that will yield an agreeable outcome. Then again, it is difficult to model a function for the transformative calculations to be utilized to determine that needs the blend of the considerable number of conceivable outcomes. This research classified the failures of the software to optimize the maintenance schedules. The study showed and anticipated failures of complex software systems. Log documents are gathered of the 4 characteristics and 100 examples. Event-driven error log records are displayed with grouping and SVM. The outcomes demonstrate the predominant execution of active learning and the SVM. Expecting that all figure failures can be maintained a strategic distance from our displaying strategies may prompt a great change of framework accessibility. The purposed show accomplishes the best execution with the most instructive information.

Future work

This research is particularly focused on the use of the SVM to classify the failure of software, and it might have left any question in the

readers and researchers such as the preferred usage of the ML techniques such as SVM and active learning.

Future work will incorporate research on the legality of our model and its comparison with the other proposed models for the failure classification. We have provided an overview of the previous approaches for failure classification, but not in terms of the TP Rate, FP Rate, Precision-Recall, F-Measure, MCC, ROC Area, and PRC Area. The future study will provide an analytical evaluation of the machine learning techniques for prediction purposes. The results of the different techniques will be generated with the same set of data by using differently supervised and unsupervised techniques and classifiers.

References

Anwar, K., Rahman, T., Zeb, A., Saeed, Y., Khan, M. A., Khan, I., . . . Abdollahian, M. (2021). Improving the Convergence Period of Adaptive Data Rate in a Long Range Wide Area Network for the Internet of Things Devices. *Energies, 14*(18), 5614.

Baldoni, R., Lodi, G., Montanari, L., Mariotta, G., & Rizzuto, M. (2012). *Online black-box failure prediction for mission critical distributed systems.* Paper presented at the International Conference on Computer Safety, Reliability, and Security.

Campbell, C., Cristianini, N., & Smola, A. (2000). *Query learning with large margin classifiers.* Paper presented at the ICML.

Cao, L. (2003). Support vector machines experts for time series forecasting. *Neurocomputing, 51*, 321-339.

Chillarege, R., Bhandari, I. S., Chaar, J. K., Halliday, M. J., Moebus, D. S., Ray, B. K., & Wong, M.-Y. (1992). Orthogonal defect classification-a concept for in-process measurements. *IEEE Transactions on software Engineering, 18*(11), 943-956.

Chu, T., LEHNER, J., MARTINEZ-GURIDI, G., & YUE, M. (2006). *A review of software-induced failure experience*: Brookhaven National Laboratory.

Dijkstra, E. W., Dijkstra, E. W., Dijkstra, E. W., & Dijkstra, E. W. (1976). *A discipline of programming* (Vol. 613924118): prentice-hall Englewood Cliffs.

Garg, S., Puliafito, A., Telek, M., & Trivedi, K. (1998). Analysis of preventive maintenance in transactions based software systems. *IEEE transactions on Computers, 47*(1), 96-107.

George Box, S. H. (2010). 25 Great Quotes for Software Testers. [Press release]

Gray, J. (1990). A census of Tandem system availability between 1985 and 1990. *IEEE Transactions on reliability, 39*(4), 409-418.

Guerra, L., McGarry, L. M., Robles, V., Bielza, C., Larrañaga, P., & Yuste, R. (2011). Comparison between supervised and unsupervised classifications of neuronal cell types: a case study. *Developmental neurobiology, 71*(1), 71-82.

Hoffmann, G. A., Salfner, F., & Malek, M. (2011). *Advanced failure prediction in complex software systems*: Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät ….

Khan, I., Ali, S., & Khusro, S. (2019). *Smartphone-based lifelogging: An investigation of data volume generation strength of smartphone sensors.* Paper presented at the International Conference on Simulation Tools and Techniques.

Khan, I., Khusro, S., Ali, S., & Din, A. U. (2016). Daily Life Activities on Smartphones and Their Effect on Battery Life for Better Personal Information Management: Smartphones and Their Effect on Battery Life for Better Personal Information Management. *Proceedings of the Pakistan Academy of Sciences: A. Physical and Computational Sciences, 53*(1), 61–74-61–74.

Kumaresh, S., & Ramachandran, B. (2012). Defect prevention based on 5 dimensions of defect origin. *International Journal of Software Engineering & Applications (IJSEA), 3*(4).

Lewis, D. D., & Gale, W. A. (1994). *A sequential algorithm for training text classifiers.* Paper presented at the SIGIR'94.

Li, Y., & Lan, Z. (2006). *Exploit failure prediction for adaptive fault-tolerance in cluster computing.* Paper presented at the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06).

Lu, H., & Cukic, B. (2012). *An adaptive approach with active learning in software fault prediction.* Paper presented at the Proceedings of the 8th International Conference on Predictive Models in Software Engineering.

Mann, C. C. (2002). Why software is so bad. *Technology Review, 105*(6), 33-38.

Meneely, A., & Williams, O. (2012). Interactive churn metrics: socio-technical variants of code churn. *ACM SIGSOFT Software Engineering Notes, 37*(6), 1-6.

Mohandes, M. A., Halawani, T. O., Rehman, S., & Hussain, A. A. (2004). Support vector machines for wind speed prediction. *Renewable energy, 29*(6), 939-947.

Ogheneovo, E. E. (2014). Software dysfunction: Why do software fail? *Journal of Computer and Communications, 2014*.

Salfner, F., Lenk, M., & Malek, M. (2010). A survey of online failure prediction methods. *ACM Computing Surveys (CSUR), 42*(3), 1-42.

Salfner, F., & Tschirpke, S. (2008). *Error Log Processing for Accurate Failure Prediction.* Paper presented at the WASL.

Tay, F. E., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *omega, 29*(4), 309-317.

Tong, S., & Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of machine learning research, 2*(Nov), 45-66.

Xu, Z., Yu, K., Tresp, V., Xu, X., & Wang, J. (2003). *Representative sampling for text classification using support vector machines.* Paper presented at the European conference on information retrieval.

Yousafzai, B. K., Afzal, S., Rahman, T., Khan, I., Ullah, I., Ur Rehman, A., . . . Cheikhrouhou, O. (2021). Student-Performulator: Student Academic Performance Using Hybrid Deep Neural Network. *Sustainability, 13*(17), 9775.