CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

A study on Fault Identification in Continuous Integration Pipelines using Machine Learning

Bachelor of Science Thesis in Software Engineering and Management

Zubeen S Maruf

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

A study on Fault Identification in Continuous Integration Pipelines using Machine Learning
In collaboration with Ericsson

Supervisor: Gregory Gay
Examiner: Daniel Strüber

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
University of Gothenburg
Chalmers University of Technology
Gothenburg, Sweden 2023

A study on Fault Identification in Continuous Integration Pipelines using Machine Learning

1st Zubeen S Maruf
University of Gothenburg Gothenburg, Sweden
gusmarzu@student.gu.se

I. Introduction

A. Background

Ericsson is a telecommunication company working with development, network systems, and software development and running operations for telecom service providers. Teams at Ericsson provided end-to-end services for developing telecommunication products and services. Companies like Ericsson develop, deliver and manage products by providing hardware, software, and services to enable customer satisfaction and are expanding rapidly. For such growth and scalability, the demand for machine learning has risen [3] due to the technological shift and increasingly complex systems. The more complex the system, the more time developers spend debugging problems and determining if the problem was caused by hardware or software failure. Machine learning techniques could potentially assist in identifying software or hardware issues for more cost-effective and efficient fault detection and diagnosis methods in such complex systems. RUX is a lab at Ericsson that falls under the radio software group and is focused on radio performance and verification. The lab specializes in detecting and quickly visualizing changes in Radio performance on the module level through the life-cycle of Radio hardware and radio software. Verification engineers working with RUX debug failed test cases running in a Continuous Integration (CI) loop. These failed test cases result from hardware (the testing environment and the radio itself) or software (the framework and radio software) problems.

B. Problem Domain & Motivation

Ericsson has a distributed working environment where end-to-end testing ensures maximum output capacity for product releases. Verification engineers working from multiple countries in one team ensure a full cycle of end-to-end testing in the RUX lab [11]. There are many benefits of working end-to-end to provide maximum support to the project, but there are some challenges when work depends on teamwork among colleagues. Engineers sometimes require consultation from colleagues working in a different time zone to debug issues. To minimize the lead time to wait for the team to start work in a different time zone, a machine learning model could assist the verification engineer in debugging testcases and classify if a test failure in the CI loop was software or hardware-related.

We may categorize RUX failures into two categories hardware(HW) and software(SW) failure, and then we can further divide them into subcategories. Software is further divided into framework and radio software, while the hardware is further divided into the test environment and radio hardware. Software issues can be identified by changing different software versions. Framework(FW) issues can be identified by changing different FW versions. Troubleshooting a failed test case or a failed Jenkins job(automated tests in the CI) in RUX involves manually searching for past results, finding the last successful execution, and manually running tests with different Radio software and framework versions. This takes up a lot of RUX resources, both in terms of engineer hours and equipment. Automation of fault classification of this process would increase the efficiency of the troubleshooting process.

C. Purpose

This study aims to develop a machine-learning model that performs fault classification on the failed test cases in the CI. We have used automated reruns of test cases with different radio software to collect a large dataset. A large dataset was be collected by using CI test results from different types of radios(example 4,8,64 ports) as inputs, and then use machine learning techniques to analyze the data and determine whether the cause of failure was related to the test environment or the software. Additionally, the model is used to investigate whether it's possible to correlate or cluster some data to get more insights on the failure for future use; this would involve identifying the specific type of hardware that was the primary factor in the test environment, for a software failure, which framework version or software version was responsible for the failure. This thesis aims to find resource-efficient implementations of the selected algorithms and investigate the choices of the selected machine learning algorithms. The focus of this thesis is limited to state-of-the-practice deployments

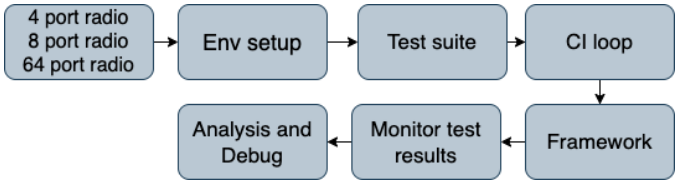

Fig. 1. RUX current workflow for radio performance and verification

of machine learning algorithms adapted to the RUX way of working.

II. Related work

Ericsson is a company working with hardware and software to achieve high radio reliability; the interaction between hardware and software is crucial[9]. The majority of testing procedures and methods are relatively flat from 20 years ago [6]. Some numerous instruments and methods can be used in the modern era for fault detection during testing. Gupta and Bathla [6] explain how debugging is only one aspect of testing. Together with the right techniques, effective testing also demands more than just finding and fixing errors. Additionally, it is utilized in the validation, verification, and reliability measurement processes, and such testing techniques are used to test radio software on different radios.

Automation is an effective approach to reducing cost and time. Even though manual testing is less expensive and less effective than automation, finding an automated process that can examine the vast array of combinations and identify possible failure places is desirable [5]. The testing environment can consist of different types of hardware, which can lead to hardware failures. A study was done to run a simulation to perform hardware failure profiling for each hardware component [7]. The authors discuss the factors that affect this profile. The designed algorithm was able to keep track of the health of 14 hardware samples, alert testers to approaching failures well in advance of them, and give them enough time to address the issue before the failure actually happened[1]. Iyer [8] examines the effectiveness of recovery management and how the operating system handles HW/SW faults. Hardware was discovered to be responsible for close to 35% of all observed software failures. The investigation reveals that the operating system rarely has the ability to recognize that a software fault can be caused by a hardware issue. The effectiveness of system recovery in preventing the spread of HW/SW mistakes is used to gauge the effect of HW/SW errors on the system. HW/SW mistakes have a system failure probability that is almost three times higher than that of general software problems. Given that the observed HW/SW mistakes follow a particular pattern, it is possible to exploit these patterns to anticipate and recover errors intelligently.

A. Algorithm selection

The selected algorithms are required to operate right after a test suite is finished running in order to provide valuable predictions for the verification team to analyze results. Multiple strategies can be applied to achieve this conclusion, varying from the different algorithms used. Models need to be of sufficiently small size in order to make predictions that can fit between CI tests. The training can, however, easily be parallelized to perform distributed training on separate cores, as described by George et al [12]. A KNN model can also be useful and can be found in popular papers such as [13]. Decision tree is a well-known technique for a given large dataset like this one that was first introduced by Svetnik et al. [14]. Decision trees can expand quickly, which has an impact on their memory footprints. Decision trees can, like neural networks, rapidly grow in size, which affects their memory footprints. There are many tricks to reduce the size of the decision trees in Decision trees, such as Feature-budgeted Decision trees as proposed by Nan et al. [15] or the Optimally Pruning Decision Tree Ensembles With Feature Cost also by Nan et al.[16].

B. Overcoming gaps and limitations

The study of an Ml model in this paper directly impacts the internal workflow of the radio software lab at Ericsson. To maximize customer value with products designed to cost[18] is one of the main goals of Ericsson; in order to have cost-effective ways and excel in total development efficiency, we may ask the following questions:

- What did we develop?
- How do we develop?

To answer what we develop? We developed an ML model in the RUX CI loop, which would help verification engineers categorize hardware and software failures during the development of new radios. This efficient working method would allow projects to reach product release deadlines without delays. To answer How we develop? With the cooperation of verification engineers, analyze CI test results and create the dataset for model training. Limitations include the number of CI test runs daily. Insufficient historical data to add to the dataset for model training.

III. Methodology

This study analyses a case study for fault classification ML models using data from Ericsson. Test suites are run on CI loops to test radio software. There may be failed test cases during test runs, which can be classified as hardware or software, as mentioned above in the background section. The research aims to classify these failures using machine learning; the study utilizes Ericsson RUX test results for model training and Ericsson verification engineers to label data using a list of questions provided and explained in the sections below. To create the case study, the instructions offered by Runesson and Höst [19] were taken into account. Case studies give a deeper insight of the phenomenon being studied than controlled experiments, even while they don't get the same conclusions[19] similarly we conducted this study to understand what could be done with the CI test results which we used to create a fault classification model utilising different techniques and finding the best results which is shown below in the document. Data collection was done through surveys[19], and during the compilation of the dataset for our study, we worked closely with the issue lists shown in subsection C and had extensive discussions with the verification engineers.

A. Case study context

Currently, when a failure occurs, engineers raise a JIRA ticket and work to identify whether the problem is software-related or hardware-in-nature. As they solve the issue, they

TABLE I
OVERVIEW OF OUR ACTION PLAN FOR THIS STUDY

Action	Description
Objective:	Explore CI test results from training fault classification ML model to conclude if the cause of failure was software or hardware-related
Case:	Classify testcase failures as Software or Hardware
Theory:	Fault classification models
Research Questions:	RQ1, RQ2
Methods:	Tool evaluation
Selection Strategy:	Data collected from Ericsson RUX CI loop

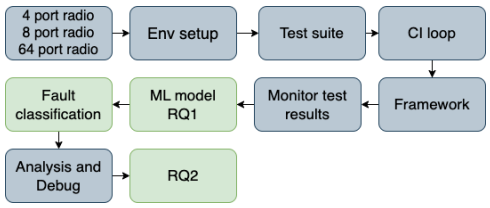

Fig. 2. Overall pipeline showing Multiple radios with an environment setup running test-suites on different radio SW in the CI loop.

document their findings in the ticket and provide an overview of the issue and how they resolved it. We have implemented machine learning techniques to classify the failure caused by the radio software or environment. By doing so, verification engineers can save significant amounts of time in the debugging process and more quickly pinpoint the root cause of the issue. With a machine learning model in place, engineers can rely on a more efficient and accurate method of classifying failures, potentially resulting in faster resolution times and contributing to better software testing practices.

B. Research Questions

The following research questions are based on the purpose of the study:

RQ1) How does the performance of different machine learning techniques vary when applied to fault classification using the current dataset, and what are the factors that could affect their performance?

Hypothesis - The machine learning model's performance varies on the dataset's size and quality. The purpose of these new radio products is introduced; therefore, test parameters can vary, leading to changes in the dataset. This variation has led to a change in performance for the machine learning model, as new products do not have historical data.

RQ2) Can the collected dataset be clustered or correlated to identify patterns and correlations that may indicate a specific cause of failure, and how reliable are the results obtained?

Hypothesis - The collected dataset can be clustered and correlated to identify different patterns and correlations that may indicate a specific cause of failure. The results obtained

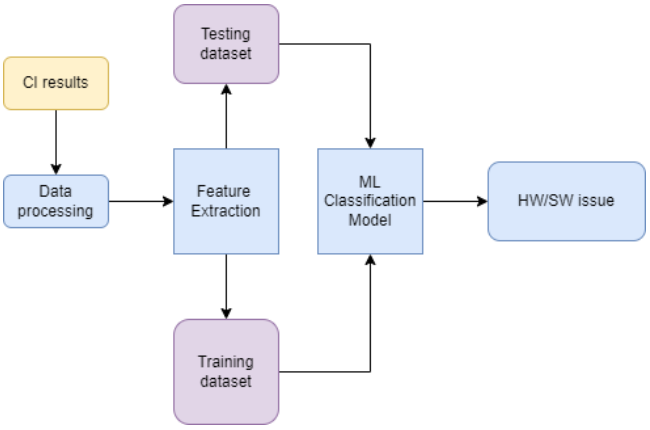

Fig. 3. CI results used as input to create the dataset. The dataset is split into training and testing 80/20. The ml model provides a prediction of HW/SW failure from the given dataset.

may provide valuable information for improving the reliability of the radio software and persistent environment/hardware failures.

C. Dataset creation

To build a prediction model, we created a dataset of cases labeled with the corresponding cause of failure.

- 0 = Pass
- 1 = Software
- 2 = Hardware

The difficulty of going through and extracting information from Jira tickets to determine whether the failure was hardware or software can be troublesome [10]. As a result, we sent a list of issues to the verification team, and this assisted us in understanding many SW/HW problems they troubleshoot regularly. The issues were as follows, and the list includes the details and solutions offered by the verification team.

- Name of Radio and position in the lab
- Test suite/ Build number
- Cause of possible failure
- Link to Jira ticket
- Json file of completed test run if possible

A Jira ticket contains the raised issues and how the issue was resolved but does not give any direct answer if the issue was hardware or software-related. Five verification engineers received a list of issues assigned to them on the Jira ticket every week. We have collected two weeks of test run data from one 4-port radio. The radio software test suite runs a nightly CI loop, therefore, 14 test runs. Performing such labeling through a list gave us an answer to the issue, where the verification engineer could state if the issue were hardware or software. This assisted us in answering RQ1 once the model was operational and the results were compared. Using this input from the verification engineers as a subject helped us by applying different classification models such as the K-means algorithm, support vector machine, tree-based algorithm, and naive bias. Moreover, we have planned to apply unsupervised

algorithms such as k-means clustering to find out the patterns of the individual feature compared to our target variable. By applying those aforementioned methodologies, we have better understood our curated dataset[6]. This enabled us to respond to RQ1 and RQ2. Therefore, we believe the case study methodology best suits this research. The following **Table VI** shows the initial creation of the dataset. The name of the testcase is represented by the test case group, and the event name shows the radio's port number together with the test case code. Finally, the result is the output value of the radio port running the radio software. Min and max are the maximum and lower limits for the output value range where the test case is supposed to pass. Test description refers to the many parameters linked to the testcase. As the test run is completed, a determination is made as to whether the testcase passed or failed. Finally, the failure column refers to the labeling we established.

D. Data analysis

After completing the initial data collection phase, a thorough Exploratory Data Analysis (EDA) was conducted, as depicted in **Figure 4**. The analysis aimed to gain insights into the collected dataset by examining its characteristics and distribution. Upon analyzing the dataset **Table VI**, it was observed that a significant class discrepancy became evident when the dataset was divided into three distinct classes, namely pass cases, software failures, and hardware failures. This discrepancy implies that there were notable differences in the distribution and occurrence of these classes within the dataset. However, by modifying the dataset division to include only two classes shown on **Table II**, specifically software failures and hardware failures, while excluding the remaining success data, the observed class discrepancy appeared to be relatively smaller as shown in **Figure 5**. This suggests that the disparities between software failures and hardware failures were more pronounced in comparison to the successful cases. It is worth noting that these initial findings highlight the importance of considering the class distribution within the dataset during subsequent analyses and modeling. The observed class discrepancy can potentially impact the performance and generalizability of any predictive or classification models that may be developed based on this dataset. Further investigations and appropriate strategies may be required to address this class discrepancy and mitigate potential biases in subsequent analyses.

The choice of performance metrics for research questions **(RQ1)** depends on the class distribution observed in the dataset. If the class distribution remains imbalanced, the F1 score, particularly the macro F1 score, should be chosen as the primary performance metric. The F1 score considers precision and recall, providing a balanced predictive model performance assessment. Calculating the F1 score per class and taking the average, we have identified potential biases or weaknesses in the model's performance across different classes. If the two-class dataset exhibits a relatively balanced distribution, accuracy can be chosen as the main performance metric. Accuracy measures the overall correctness of the model's predictions and is suitable when the class distribution is fairly uniform.

E. Model training

In accordance with the study conducted by Emiris and Dimitrios [4], our research has adopted a similar approach to identify the most informative parameters within the executed test cases, thereby facilitating effective classification. Consequently, we present the results of our feature extraction and data pre-processing procedures, which are essential for our investigation.

For the initial phase of our analysis, we have opted to utilize four distinct algorithms: K-nearest neighbor, tree-based algorithms like Decision tree, support vector, and naive bias. These algorithms have demonstrated superior performance when applied to tabular data, making them well-suited for our research objectives[20][21]. Model training for all models listed below was conducted using data collected from 14 test runs the dataset shown in **Table II**.

It is important to note that this selection is based on their proven track record and notable success in handling similar data scenarios, and the choice of these algorithms serves as a strategic decision within our research framework. We anticipate their implementation has given insightful results and contributed to achieving our research objectives.

For the models, we Separate features (X) and class labels (y) in this case

- X = [['Min', 'Max','Result']]
- y = ['Failure']

In order to prove the case study, we have designed an experiment to test four distinct algorithms: K-nearest neighbor, tree-based algorithms like Decision tree, support vector, and naive bias. We started by pre-processing the data. The data was pre-processed to ensure that Min, Max, and Result values are interpreted as floats for calculations and modeling. We used normalization as part of the pre-processing to ensure feature scaling. We used one hot encoding for categorical features to transform categorical data into numerical data for the above-chosen algorithms. To measure the model's performance, we have performed these two techniques in terms of dataset split: train-test split and k-fold cross-validation. Train-test split involves splitting the available dataset into two parts, potentially split into training and testing 80/20. K-fold cross-validation is a more reliable as our dataset size is small. So we have used k-fold cross-validation with a k=5 . Train the KNN model with k = 1-10 number of neighbors to consider.

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

The Euclidean distance calculation is performed implicitly by the KNeighborsClassifier algorithm. The dataset created had a bias toward a higher pass rate; therefore, for fairness of data output, as Joymallya [17] stated the rebalancing of internal distributions. To reduce bias, we combined the undersampled indices with the indices of the other classes and then randomly chose the same number of pass samples

TABLE II
FINAL DATASET FOR ONE RADIO PORT COMPILED FROM 14 TEST RUNS

Test Case Group	Event Name	Min	Max	Result	Test Description	Test Pass	Failure
TX_MOD_ACC_0001	TX_MOD_ACC_0001_1	0	12,5	4,53	¡AB_W_6_Dd_MODtm1¿	FALSE	1
TX_MOD_ACC_0001	TX_MOD_ACC_0001_4	0	12,5	1,62	¡TRmodsigtm4¿;¡TRmodsigtm4¿	FALSE	2
TX_MOD_ACC_0001	TX_MOD_ACC_0001_9	-20	20	2,78	¡TRmodsigtm5¿;¡TRmodsigtm5¿	FALSE	2
TX_LTE_MODACC_0014	TX_LTE_MODACC_0014_4	0	9	9,05	¡AB_L200SUB50_2_Dd_BOFF-1.8dB	FALSE	1
TX_LTE_MODACC_0016	TX_LTE_MODACC_0016_4	0	9	9,08	¡BA_L200SUB50_2_Dd_BOFF-1.8dB	FALSE	1

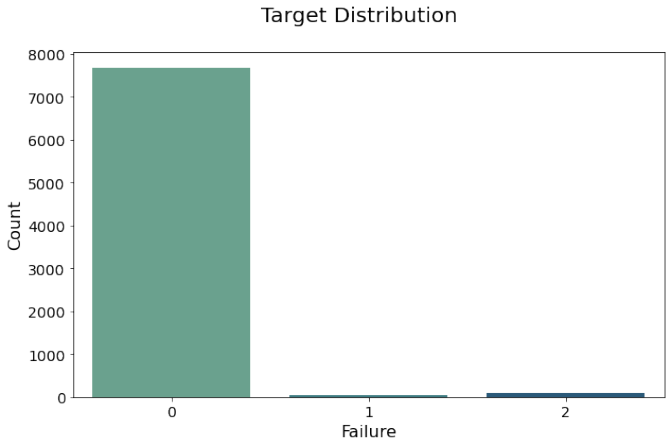

Fig. 4. Illustration of class frequency/distribution of the entire collected dataset. Here, 0 = Pass test, 1 = Software failure, 2 = Hardware failure.

Fig. 5. Illustration of class distribution after refining the collected dataset.

as the other classes to obtain the undersampled dataset. The second algorithm to train was a naive bias algorithm, P(A—B) = P(B—A) * P(A) / P(B); the algorithm works best with smaller datasets which is why we implemented it after the dataset division to include only two classes removing the pass testcases **Figure 5**. We have also implemented a Support Vector Machine using the hyperparameters C and gamma; after they have been defined in combination with the grid search algorithm, we have determined the accuracy and F1 score for SVM. Finally, we employed decision tree learning, where each leaf node represents a classification or a choice and the tree starts with a root node with no incoming branches. The potential outcomes that can be reached from the features are contained in the leaf nodes.

To ascertain the validity of our predictive model and address the research question pertaining, we have employed statistical significance testing to evaluate the feature selection process. We can determine whether the observed outputs are statistically significant by conducting statistical tests, such as calculating p-values or confidence intervals.

By calculating p-values, we can determine the probability of obtaining results as extreme as the ones observed, assuming that the null hypothesis is true. A low p-value indicates that the observed results are unlikely to occur by chance alone, thus providing evidence to reject the null hypothesis and support the alternative hypothesis. Alternatively, confidence intervals provide a range of plausible values within which the true effect or relationship in the population is likely to lie. If the calculated confidence interval does not include the null value (e.g., zero), it suggests that the observed effect or relationship is statistically significant.

Through the application of these statistical tests, we can ascertain whether the selected features exhibit statistically significant associations with the target variable. This analysis is crucial for ensuring the reliability and validity of our predictive model, enabling us to draw meaningful conclusions and make informed decisions based on the obtained results.

RQ2 We have used K-means clustering, an unsupervised machine learning algorithm that aims to partition a dataset into a specified number of clusters. The algorithm works by iteratively assigning each data point to the cluster whose centroid is closest to it, and then updating the centroids to be the mean of all the points in the cluster. This process continues until the cluster assignments no longer change or a maximum number of iterations is reached. We have utilized K-means clustering to calculate the inertia and silhouette scores for each value of k = 2-10, which can be seen in **Figure 6** and **Figure 7**. By following elbow method we have found out that the kink point is 3 (as shown in **Figure 7**) however we have found out the optimal result when cluster=4. The result can be found in **section IV**

IV. RESULTS

To establish a benchmark for our acquired dataset, we have harnessed the potential of four prominent machine learning algorithms: K-NN, Decision Tree, Naive Bayes, and Support Vector Machine (SVM). During the model training phase, k-fold cross-validation with five folds was implemented. Our

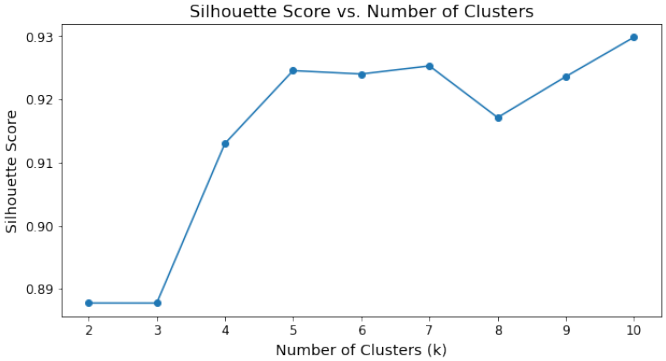

Fig. 6. Illustration of how Silhouette score measures

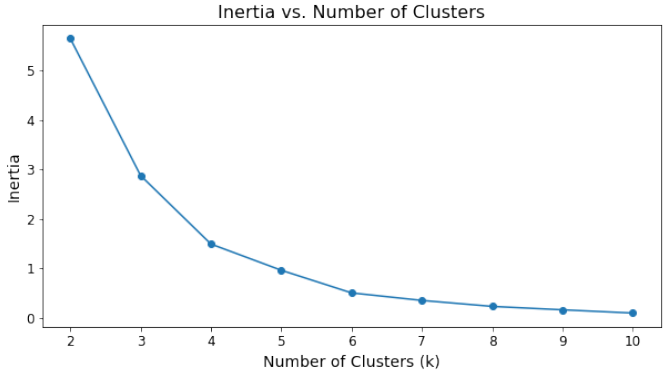

Fig. 7. Illustration of how Inertia score measures

dataset underwent division through the utilization of a train-test split, allocating 20% of the data for testing while dedicating the remaining 80% for model training via the k-fold technique. The assessment of model performance transpired using the reserved 20% test data.

The comprehensive dataset performance is presented in **Table III**. Analysis of the results showcased in **Table III** highlights the favorable performance of the K-NN and SVM models within the set of four configurations. The process of determining the optimal parameters for these models involved the application of the grid search algorithm. Subsequent experimentation led to the observation that the K-NN model exhibited superior performance with an optimal value of n_neighbors = 4, while the SVM model demonstrated its best performance with the parameter values C = 100, gamma = 0.1, and a kernel of 'rbf'.

Furthermore, an in-depth exploration of the confusion matrices for all four models has been conducted. As depicted in **Figure 8**, the analysis reveals that in the cases of K-NN and SVM, the cumulative count of True Positives (TP) is 14, while the tally of True Negatives (TN) amounts to 29, surpassing the performance of the alternative configurations. The assessment of False Positives (FP) and False Negatives (FN) demonstrates the superiority of K-NN and SVM. Notably, our empirical investigation of the test set indicates that both K-NN and SVM exhibit the prediction of only a solitary FP and FN each.

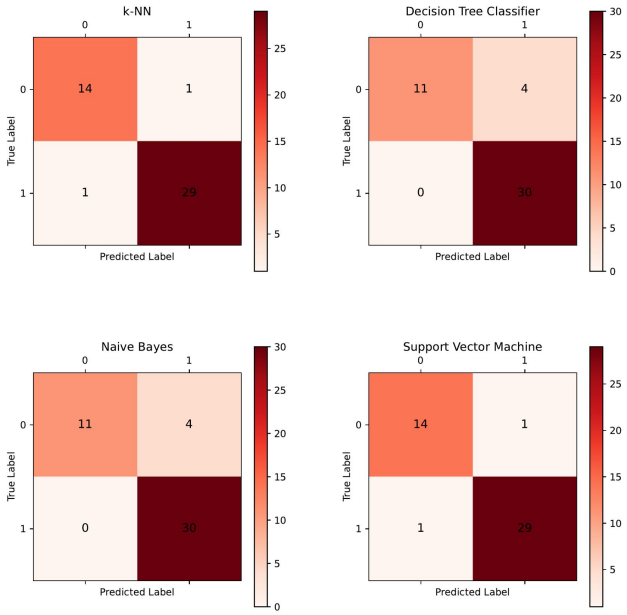

Fig. 8. Confusion matrix of all the four models. From the confusion matrix also see that k-NN and SVM outperform other configurations.

Regarding the analysis of the K-NN algorithm, our investigations have determined that the optimal value for the hyperparameter n_neighbors is 4. To comprehensively evaluate its behavior, we conducted a series of experiments covering n_neighbors values within the range of 1 to 10. Illustrated in **Figure 9**, the performance of the K-NN algorithm under varying n_neighbors values is presented. This visual representation reaffirms our empirical observation that K-NN achieves its peak performance, boasting a 0.93 f1-score, when n_neighbors is set to 4.

To assess the statistical significance of the conducted test, we select two optimal models derived from our preceding experimental endeavors and subject them to a t-test analysis. Here, the null hypothesis is - that there is a significant difference in performance between the KNN and SVM models, whereas the alternative hypothesis is there is no significant difference in performance between the KNN and SVM models. As depicted in **Table IV**, both models exhibit p-values less than the predefined threshold of α, compellingly rejecting the null hypothesis. This substantiates that the observed outcome is improbable to have arisen purely by random chance and there is no significance between KNN and SVM models in terms of performance under this dataset. The implications of this outcome suggest a statistically meaningful variance or influence amid diverse segments of k-fold cross-validation. However, it is prudent to acknowledge that the p-value, despite its interpretative utility in establishing statistical significance, does not inherently convey insights regarding the practical import of the identified effect.

In addressing **RQ2**, we have effectively employed the K-

TABLE III

EVALUATION OF OUR DATASET PERFORMANCE REVEALS THAT K-NN AND SVM OUTPERFORM OTHER CONFIGURATIONS. THE OPTIMAL PARAMETERS WERE DETERMINED THROUGH THE UTILIZATION OF THE GRID SEARCH ALGORITHM.

Model	**Best Params**	**f1(K-fold)**	**f1(Test set)**	**Accuracy(Test set)**
K-NN	n_neighbors=4	**0.85**	**0.93**	**0.95**
Decision Tree	max_depth=2	0.83	0.84	0.91
Naive Bayes	alpha =10, fit_prior=False	0.83	0.85	0.91
Support Vector Machine	C=100, gamma=0.1, kernel='rbf'	0.84	**0.93**	**0.95**

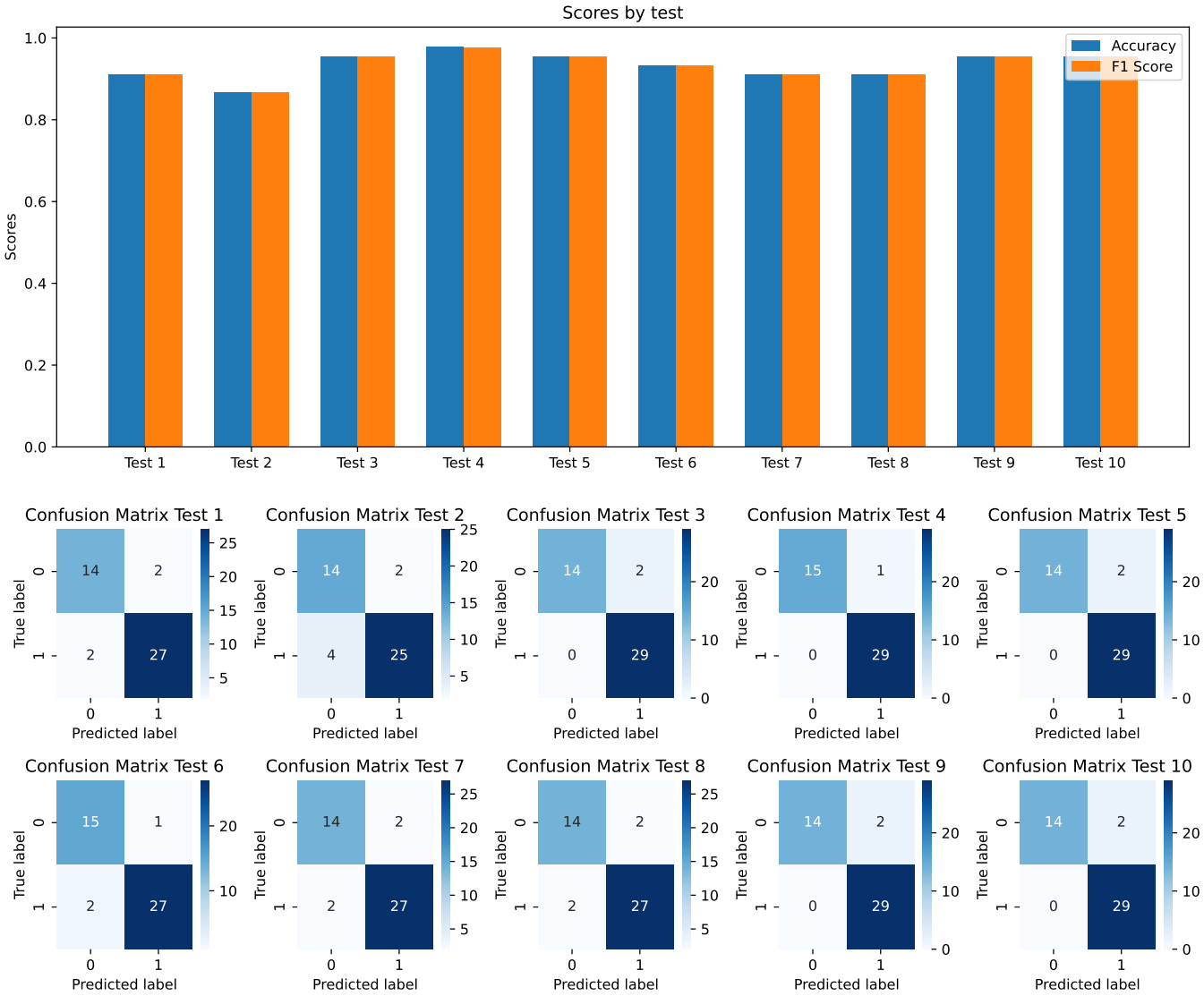

Fig. 9. Illustration of the analysis of K-NN algorithm with different n_neighbors value from (1 - 10). We can observe that from Test 4 where the n_neighbor = 4 gets the maximum results.

TABLE IV
P-VALUE OF K-NN AND SVM

Model	p_value(α=0.05)
K-NN	0.0013
SVM	0.00000173

TABLE V
PERFORMANCE EVALUATION OF K-MEANS ALGORITHM

number of clusters	accuracy (in %)
3	16
4	21

means clustering algorithm to partition our assembled dataset. To ascertain the most suitable count of clusters, we have methodically employed the elbow method (refer to **Figure 7**), revealing a noticeable inflection point at a cluster count of 3. Subsequently, we evaluated the classification performance, yielding an accuracy score of 16% as detailed in **Table V**. However, it's noteworthy that the highest accuracy, amounting to 21%, was achieved when the number of clusters was set at 4. Consequently, we assert that a cluster count of 4 is the optimal choice for this dataset.

V. DISCUSSION

The findings of this study have given us insights into how machine learning is used in the radio software lab at Ericsson. In this section, a brief overview of the research procedure is provided. Machine learning techniques have been used to classify failures caused by radio software or environment, saving time in debugging and enabling faster identification of root causes. This efficient and accurate method of classifying failures could lead to faster resolution times and improved software testing practices, benefiting verification engineers.

Due to the dataset being relatively small, as there was not much historical data present and occurrences of failed testcases we have tried different methods to increase the size of the dataset. A script that would create test results in JSON while making a test fail by, for example, setting the result to 0.9 x MINIMIT, or to 1.1 x MAXLIMIT. So, from one original JSON, we use it as a template to generate a number of failed JSON files. This approach was excessively complicated, and there were no patterns to use when choosing which parameters to fail. This raised concerns because it was difficult to construct synthetic test cases without knowing which tests to fail; therefore, the proposal was abandoned. As we monitored nightly, CI test runs for up to 14 days referred to 14 test runs while keeping in mind that each test run produced different findings and that the number of failures varied sometimes. As a result, we chose to work with a smaller dataset, fine-tuning the hyperparameters. Finding the problem area and building a dataset from scratch were the biggest challenges for this research paper, which required excellent teamwork and cross-functional abilities to complete. Continuous meetings with engineers to understand workflow and potential implementation areas.

VI. THREATS TO VALIDITY

1) Threats to External validity: The actions and artifacts of Ericsson and, more specifically, the RUX lab are the subject of the case study. A significant drawback of this isolation is that the research's results and conclusions will only apply to Ericsson and be difficult to generalize as it is working with a problem specific to a certain business area. The conclusion drawn from the case study may not be generalized to other businesses or areas outside of Ericsson, but a similar way of working can be followed to determine fault classification in other use cases. The proposed solution of determining (HW/SW failure) to mitigate the risk of human error may introduce new sources of bias.

2) Threats to Internal validity: The research heavily depends on obtaining data from verification engineers' list of issues; human error risks can lead to inaccuracy. To mitigate such risks, we will continue working with the team to develop automated labeling techniques for extracting information from Jira tickets.

3) Threats to Construct validity: The model developed will aid verification engineers in their daily debugging tasks, but this can lead to users being too dependent on the given output. This risk will be reduced by explicitly informing every subject that the model is based on historical data from previous run test suites; it will not account for environmental factors such as power loss in the lab and physical shifting of or unrecorded changes in the testing environment.

VII. CONCLUSION

As technology develops, Ericsson's business is expanding and introducing new machine-learning techniques for a better way of working in daily tasks that make life easier for employees. In this case study, we classified failed test cases as hardware or software failures using K-NN, Decision Tree, Support Vector Machine and Naive bias methods. Despite lacking historical data and no data labeling methods prior to this study, we have implemented these machine learning algorithms to answer the following research questions: **RQ1) How does the performance of different machine learning techniques vary when applied to fault classification using the current dataset, and what factors could affect their performance?**
RQ2) Can the collected dataset be clustered or correlated to identify patterns and correlations that may indicate a specific cause of failure, and how reliable are the results obtained?

For **RQ1)**, The K-NN and SVM models performed well in four different configurations, according to the dataset performance study[13]. The grid search algorithm was used to find the ideal settings. The SVM model performed best with parameters C = 100, gamma = 0.1, and a kernel of 'rbf', whereas the K-NN model performed best with an ideal value of n_neighbors = 4. We chose the two best models obtained from our prior experimental endeavors and put them into a t-test analysis to evaluate the statistical significance of the

conducted test. Both models provide p-values that are below the predetermined cutoff of $alpha$, strongly rejecting the null hypothesis. This proves that the observed result is unlikely to have occurred by pure chance. The ramifications of this result point to a statistically significant variance or influence among various k-fold cross-validation segments. However, it is prudent to recognize that the p-value does not automatically transmit information about the practical significance of the identified effect despite its interpretive utility in determining statistical significance.

To Answer **RQ2)**, we have tried K means clustering; the algorithm considers all the available information in the data when determining the cluster assignments. By using all the fields, the algorithm can consider the relationships and interactions between all the features in the data, which can result in more accurate and meaningful clusters. The optimal number of clusters was 4.

VIII. Future work

In this thesis, we implemented four machine learning algorithms, Naive Bayes, SVN, Decision tree, and KNN model, to classify whether a RUX CI loop failure is a hardware or software failure. Additional research in this field may use other methods for extracting parameters from the testcases. Testcases could be dissected and further examined to determine which parameters are beneficial and have the potential to be added to the dataset. Identify if we can correlate/cluster the data if there is a correlation between the day of the week and the number of failed tests. As more data is collected and a rich history is created, further correlations between the number of failed tests per each radio and month of the year could also be a possibility to be visualized enabling verification engineers to track issues and help them debug further. While the machine learning model only supports one radio port, handling multiple radio ports and types with additional study and data processing is possible.

References

[1] Chigurupati, Asha and Thibaux, Romain and Lassar, Noah. (2016). Predicting hardware failure using machine learning. 1-6. 10.1109/RAMS.2016.7448033.

[2] Concea, Andreea and NIȚESCU, Tudor-Alin , SGÂRCIU, Valentin. (2022). TEST AUTOMATION FOR CONTINUOUS INTEGRATION IN SOFTWARE DEVELOPMENT. UPB Scientific Bulletin, Series C: Electrical Engineering. 84. 95-106.

[3] Elcio Tarallo, Getúlio K. Akabane, Camilo I. Shimabukuro, Jose Mello, Douglas Amancio, Machine Learning in Predicting Demand for Fast-Moving Consumer Goods: An Exploratory Research,IFAC-PapersOnLine,Volume 52, Issue 13,2019,Pages 737-742.

[4] Emiris, Dimitrios. (2023). Fault Classification and Diagnosis for Rotating Equipment using Machine Learning Algorithms. 10.5957/SOME-2023-025.

[5] Gregory Gay, Tim Menzies, Misty Davies and Karen Gundy-Burlet. Automatically Finding the Control Variables for Complex System Behavior.West Virginia University, Morgantown, WV, USA, NASA Ames Research Center, Moffett Field, CA, USA.

[6] Gupta, Ms, Bathla, Dr. (2022). Comparative Study of Software Testing Technique using Manually and Automated Way. International Journal of Scientific Research in Science and Technology. 384-394.

[7] Huang, Bing Li, Xiaojun Li, Ming Bernstein, Joseph Smidts, Carol. (2005). Software-Specific Hardware Failure Profile. Proc. of The 41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit (AIAA 2005), Tucson, AZ (USA). 10.2514/6.2005-4483.

[8] Iyer, Ravishankar Velardi, Paola. (1985). Hardware-Related Software Errors: Measurement and Analysis. IEEE Trans. Software Eng.. 11. 223-231. 10.1109/TSE.1985.232198.

[9] John, Yakubu Sanusi, Abdullahi Yusuf, Ibrahim Modibbo, Umar. (2022). Reliability Analysis of Multi-Hardware-Software System with Failure Interaction.

[10] Montgomery, Lloyd Lüders, Clara Maalej, Walid. (2022). Jira: An Alternative Issue Tracking Dataset.

[11] Bosch Sijtsema, Petra M ,Ruohom?ki, Virpi,Vartiainen, Matti Journal of knowledge management, 22 Oct 2009, Vol. 13, Issue 6, pages 533 - 546

[12] D George, M Alan, and N Tia. "Parallelizing neural network training for cluster systems". In: Book Parallelizing neural network training for cluster systems, Series Parallelizing neural network training for cluster systems (2008).

[13] Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K. (2003). KNN Model-Based Approach in Classification. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds) On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. OTM 2003. Lecture Notes in Computer Science, vol 2888. Springer, Berlin, Heidelberg.

[14] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. "Random forest: a classification and regression tool for compound classification and QSAR modeling". In: Journal of chemical information and computer sciences 43.6 (2003), pp. 1947–1958.

[15] Feng Nan, Joseph Wang, and Venkatesh Saligrama. "Feature-budgeted random forest". In: arXiv preprint arXiv:1502.05925 (2015).

[16] Feng Nan, Joseph Wang, and Venkatesh Saligrama. "Optimally Pruning Decision Tree Ensembles With Feature Cost". In: arXiv preprint arXiv:1601.00955 (2016)

[17] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. 2021. Bias in machine learning software: why? how? what to do? In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2021). Association for Computing Machinery, New York, NY, USA, 429–440.

[18] Shehab EM, Abdalla HS. A design to cost system for innovative product development. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture. 2002;216(7):999-1019.

[19] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," Empirical software engineering, vol. 14,

[20] Ravid Shwartz-Ziv and Amitai Armon,"Tabular data: Deep learning is not all you need",Information Fusion, vol 81, pages 84-90,2022

[21] Rich Caruana and Alexandru Niculescu-Mizil. 2006. An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd international conference on Machine learning (ICML '06). Association for Computing Machinery, New York, NY, USA, 161–168.

Appendix

Accuracy and F1 score for respect K values

K values	Accuracy	F1
1	0.91111	0.91111
2	0.86666	0.86820
3	0.95555	0.95481
4	0.97777	0.97760
5	0.95555	0.95481
6	0.93333	0.93375
7	0.91111	0.91111
8	0.91111	0.91111
9	0.95555	0.95481
10	0.95555	0.95481

TABLE VI
INITIAL DATASET FOR ONE RADIO PORT

Test Case Group	Event Name	Min	Max	Result	Test Description	Test Pass	Failure
TX_MOD_ACC_0001	TX_MOD_ACC_0001_1	0	12,5	4,53	¡AB_W_6_Dd_MODtm1¿	False	1
TX_MOD_ACC_0001	TX_MOD_ACC_0001_2	-100	-37	-70,24	¡AB_W_6_Dd_MODtm1¿	True	0
TX_MOD_ACC_0001	TX_MOD_ACC_0001_3	-20	20	2,72	¡AB_W_6_Dd_MODtm1¿	True	0
TX_MOD_ACC_0001	TX_MOD_ACC_0001_4	0	12,5	1,62	¡TRmodsigtm4¿;¡TRmodsigtm4¿	False	2

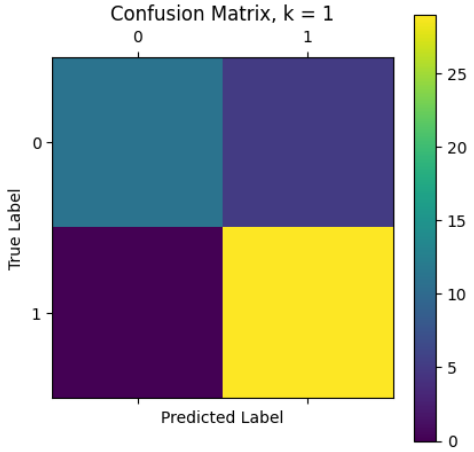

Fig. 10. K value chosen 1

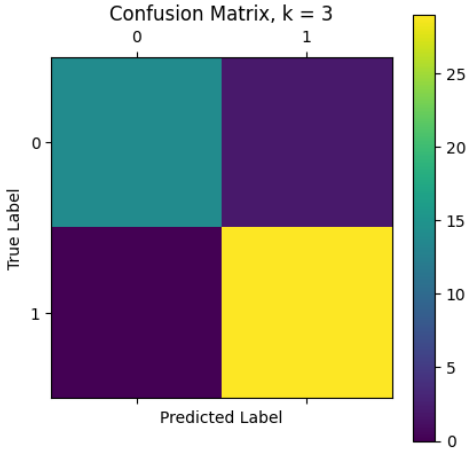

Fig. 12. K value chosen 3

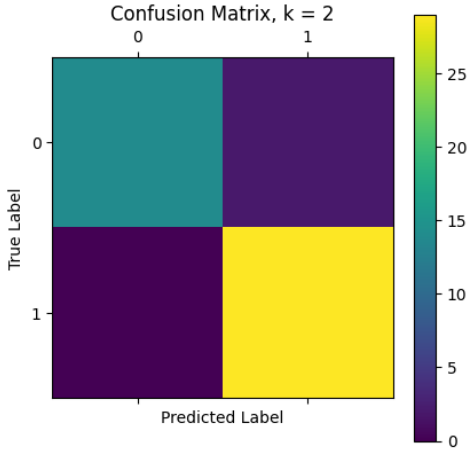

Fig. 11. K value chosen 2

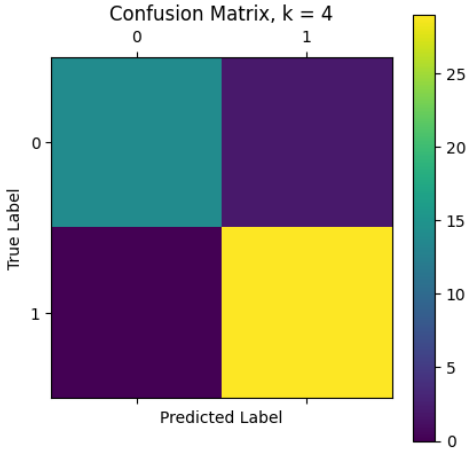

Fig. 13. K value chosen 4