Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# Improving Test Case Prioritization in Continuous Integration Environments Using Multilayer Perceptron: An Empirical Comparative Study

**MUSA ISYAKU**

musaisyaku77@gmail.com

Abubakar Tafawa Balewa University

**ABDULSALAM YA'U GITAL**

Abubakar Tafawa Balewa University

**HARUNA CHIROMA**

Abubakar Tafawa Balewa University

**Research Article**

**Keywords:** Test Case Prioritization, Continuous Integration, Multilayer Perceptron, Machine Learning, APFD, Regression Testing, Class Imbalance, Wilcoxon Signed-Rank Test

**Posted Date:** September 29th, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-9483824/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** No competing interests reported.

# Improving Test Case Prioritization in Continuous Integration Environments Using Multilayer Perceptron: An Empirical Comparative Study

**Musa Isyaku – Abdulsalam Ya'u Gital – Haruna Chiroma**

*Faculty of Computing, Abubakar Tafawa Balewa University, Bauchi, Nigeria*

Correspondence: *musaisyaku77@gmail.com*

**Abstract**

Test case prioritization (TCP) is a key optimization method in continuous integration (CI) regression testing, determined to detect faults earlier by ranking test cases based on their likelihood of revealing failures. Despite growing interest in machine learning (ML) techniques to TCP, comprehensive empirical comparisons across multiple industrial datasets and rigorous statistical validation remained unexplored. This study tries to find out whether a Multilayer Perceptron (MLP) model can improve predictive accuracy and fault detection efficiency in TCP compared against five classical ML algorithms: Random Forest (RF), XGBoost (XGB), Logistic Regression (LR), Support Vector Machine (SVM), and k-Nearest Neighbors (k-NN). Six ML models were implemented and evaluated under the same experimental conditions using a stratified 80/20 train-test split with 5-fold GridSearchCV hyperparameter tuning. The primary evaluation dataset is the ABB Robotics regression testing dataset (64,024 test case executions, 106 CI cycles). Three additional industrial datasets were used to addressed generalization: IOFROL, PAINTCONTROL, and Westermo Network Technologies. Accuracy, Precision, Recall, F1-score, and Average Percentage of Fault Detected (APFD) were used to measure the performance of the models. Statistical significance was utilized through 1,000-iteration bootstrap Wilcoxon signed-rank tests. On the primary ABB dataset, the MLP reached an Accuracy of 99.77%, Precision of 100%, Recall of 99.15%, F1-score of 99.57%, and APFD of 86.35%. Wilcoxon signed-rank tests validated statistically significant priority of MLP over all five baselines ($p < 0.05$) with large effect sizes ($|r| \geq 0.98$). Tuning hyperparameter with ReLU activation, two hidden layers (64, 32 neurons), and a learning rate of 0.001 resulted in optimal performance. Generalization experiments on IOFROL and PAINTCONTROL datasets verified the MLP's robustness. MLP consistently outperforms classical ML models in CI-based TCP tasks. This study provides empirical evidence that deep learning models can improve fault detection rates, and proved that classification accuracy alone is not enough for evaluating TCP performance under class imbalance. The APFD metric, supported alongside bootstrap statistical testing, is recommended as the primary evaluation criterion.

**Keywords:** Test Case Prioritization, Continuous Integration, Multilayer Perceptron, Machine Learning, APFD, Regression Testing, Class Imbalance, Wilcoxon Signed-Rank Test

## 1. Introduction

Despite tremendous expansion in machine learning techniques for test case prioritization, existing models, including traditional algorithms, often struggle to achieve optimal accuracy in dynamic environments such as continuous integration testing. The need for rapid and reliable testing is paramount as software development processes evolve, yet many current methods fail to efficiently control the complexities and refinement of varying test case attributes, including business requirements, complexity, execution time, and associated costs, thereby leading to suboptimal prioritization performance and reduced accuracy.

The research intends to investigate how MLP can effectively capture and model the complex dynamic relationships between predictive attributes and test case effectiveness, thereby enhancing predictive accuracy in test case prioritization. By optimizing the prioritization process, the study intends to contribute to the overall quality assurance in software development, ensuring that key functionalities are tested more effectively and efficiently. The specific objectives of the study are to: (i) Implement a Multilayer Perceptron based Machine

learning model to improve predictive accuracy in test case prioritization; (ii) Evaluate the effectiveness of the proposed model using metrics such as Accuracy, APFD, Precision, Recall and f1-score; and (iii) Compare the performance of the MLP model with traditional machine learning models, particularly Random Forest and other baseline classifiers. The objectives are structured to answer these research questions:

RQ1: How can the predictive accuracy of test case prioritization (TCP) be improved using a Multilayer Perceptron (MLP) model compared to traditional machine learning techniques.

RQ2: How does hyperparameter tuning of the Multilayer Perceptron (MLP) model such as the number of hidden layers, learning rate, batch size, and activation functions affect its performance in predicting test case failures?

This study works in the context of industrial regression testing in CI environments, using industrial datasets retrieved from ABB Robotics and Westermo Network Technologies. These datasets exhibit real-world challenges including class imbalance (approximately 80% passing, 20% failing test cases), high execution volume, and temporal variation across testing cycles. The results are therefore applicable to organizations handling large-scale automated test suites in CI/CD pipelines.

The main contributions of this paper are:

- Implementation of an MLP-based binary classifier for predicting test case failure probability in CI regression testing environments.
- Empirical comparison of six ML models (MLP, RF, XGBoost, SVM, LR, k-NN) under strictly the same experimental conditions across four industrial datasets.
- validation that classification accuracy alone is not a reliable metric in imbalanced CI environments
- Empirical analysis of the effect of hyperparameter tuning on MLP performance and convergence behavior.
- Validation of MLP generalizability across multiple datasets of varying size, and imbalance levels.
- Strong statistical validation using bootstrap Wilcoxon signed-rank tests

The rest of this paper is organized as follows: section 2 provides background on TCP and the MLP technique. Section 3 reviews related work. Section 4 presents the research methodology. Section 5 reports experimental results. Section 6 discusses findings, implications, and threats to validity. Section 7 concludes the paper and suggest future research direction

# 2. Background

## 2.1 Test case prioritization

Test case prioritization (TCP) is a regression testing optimization technique that ranks test cases by their potential to reveal faults early, ensuring that high-priority tests are executed first within limited available time. TCP techniques are generally classified into three categories: white-box techniques (based on code structure and coverage). It is also known as *glass-box testing*, as everything that is required to implement the software is visible. Black-box technique is one of the major techniques in dynamic testing for designing effective test cases based on functional requirements and execution history, and Hybrid techniques integrate both *White-Box* (structural/code-based) and *Black-Box*(execution/data-driven) elements to enhance prioritization accuracy. In CI environments, black-box and hybrid approaches are most practical since they do not require code instrumentation.

The effectiveness of a prioritization strategy is commonly measured using the Average Percentage of Faults Detected (APFD) metric, which quantifies how quickly faults are detected across the ordered test execution sequence. A higher APFD value indicates that failing test cases are ranked earlier, resulting in faster fault detection.

## 2.2 Multilayer perceptron

The Multilayer Perceptron (MLP) is a type of artificial neural network that consists of multiple layers of interconnected neurons, enabling it to learn complex patterns and relationships within data. MLPs are particularly well-suited for tasks involving classification and regression, making them valuable in various applications, including test case prioritization.

An MLP typically comprises an input layer, one or more hidden layers, and an output layer. Each neuron in the network processes input data and applies an activation function to determine its output. The connections between neurons are associated with weights, which are adjusted during the training process to minimize the difference between predicted and actual outcomes.

In the context of test case prioritization, MLPs can effectively capture the nonlinear relationships among various test case attributes, such as complexity, execution time, and historical defect data. By training on historical test case information, MLPs can learn to identify which test cases are more likely to reveal defects, allowing for more informed prioritization decisions.

The ability of MLPs to model complex interactions makes them a powerful tool for improving test case prioritization. By leveraging the strengths of machine learning, MLPs can enhance fault detection rates, improve resource allocation, and ultimately lead to more efficient testing processes in continuous integration environments. As software systems continue to evolve, the application of MLPs in test case prioritization represents a promising avenue for advancing software quality assurance practices.

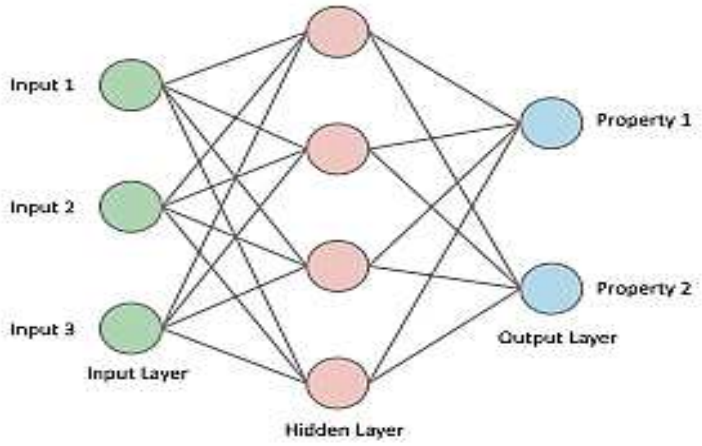



Figure 1. Structure of a Multilayer perceptron (Zhang et al., 2023)

### 2.3 Class imbalance in CI regression testing

Industrial regression testing datasets are generally class-imbalanced, with failing test cases typically representing a small fraction of executions. Standard classification metrics such as accuracy are not reliable in such settings, as a classifier that predicts all cases as passing can achieve high accuracy while completely failing to detect faults. Addressing class imbalance through balanced class weighting and using recall and F1-score for the minority (Fail) class, alongside APFD, is therefore critical for a meaningful evaluation of TCP models.

## 3. Related work

### 3.1 Deep learning and anomaly-based approaches

(Choudhury et al., 2025) proposed AnoLSTM-TCP, an anomaly-based Long Short-Term Memory method for TCP, achieving APFD values of 0.85 and 0.90 on IOFROL and PAINTCONTROL datasets respectively, outperforming RETECS, DeepGini, and DeepOrder. While AnoLSTM-TCP demonstrates the power of temporal modeling for TCP, it does not provide a systematic comparison of classical ML models under identical experimental conditions, leaving the relative contribution of deep learning over simpler baselines unclear.

(Sharif et al., 2021)proposed Deeporder, a deep learning-based prioritization approach evaluated on ABB and related datasets. (Medhat et al., 2020) combined deep learning with search-based techniques for CI regression testing in IoT systems. (Shankar & Sridhar, 2024) applied deep reinforcement learning for TCP, while (Lousada & Ribeiro, 2020) explored neural network embeddings. These methods, while innovative, are often evaluated on single datasets with limited cross-dataset generalization.

## 3.2 Classical machine learning approaches

(Sharma & Chande, 2023)highlighted that neural networks attained the highest fault detection rate of 94% in a comparative ML study. However, their research is limited by reliance on cluster selection and an inability to account for dynamically evolving test cases. (Kiran, 2025)explored multiple ML classifiers for regression TCP, finding that Random Forest achieves the best F1-score of 81% in scenario-based classification.

(Kumar & Saxena, 2024)applied RF, XGBoost, LR, SVM, and k-NN to the ABB Robotics datasets, with RF achieving the highest accuracy of 87%. However their evaluation depends completely on accuracy, which is inadequate for imbalanced datasets. (Marijan, 2023)carried out a comparative study using GBDT, SVM, ANN, and LRN with extensive hyperparameter tuning, revealing that LRN outperformed other models. (Sheth et al., 2022)compared Decision Tree, SVM, Naïve Bayes, and k-NN on five diverse datasets, with Naïve Bayes achieving the highest average accuracy of 79.76%.

## 3.3 Reinforcement learning approaches

(Bagherzadeh et al., 2022)proposed reinforcement learning (RL) approach integrated with feature selection for TCP, evaluated on the ABB dataset. (Yaraghi et al., 2023) extended this to scalable RL-based TCP in CI contexts. (Spieker & Mossige, 2026)developed the RETECS framework using RL for automatic TCP and selection. (Wu et al., 2019)introduced time-window-based RL rewards for TCP in CI. While RL methods are compelling for online learning scenarios, their reproducibility is often limited by dataset availability and the complexity of reward function design.

## 3.4 Other approaches

(Bertolino et al., 2020)compared learning-to-rank and ranking-to-learn strategies for CI regression testing. Omri and Sinz (2022) applied learning-to-rank to TCP. (Elsner et al., 2021) empirically evaluated readily available information for regression test optimization in CI. (Lima & Regina Vergilio, 2023)also evaluated ranking-to-learn approaches for TCP. (Felding et al., 2024)used simulated annealing for resource-constrained TCP in an industrial context. (Ghaleb et al., 2019)studied long CI builds on the TravisTorrent dataset. (Cheng et al., 2024) reused TCP on long-running test suites, introducing the LRTS dataset with 21,555 builds from 10 open-source projects.

## 3.5 Research gap

The reviewed literature reveals four principal gaps that this study addresses. Firstly, the current deep learning TCP studies such as AnoLSTM-TCP (Choudhury et al., 2025)do not provide systematic comparisons of classical against deep learning models under the same experimental conditions. Secondly, prior machine learning-based studies such as (Kumar & Saxena, 2024)evaluate models using accuracy metric only on a single dataset, and this limits generalizability. Thirdly, no existing study combines six comparative ML models, keeping APFD as the primary metric, with cross-dataset generalization across four or more industrial datasets within a unified framework. Fourthly, the effect of class imbalance on metrics reliability for TCP has not been systematically demonstrated across multiple datasets. This study addresses all the four gaps.

# 4. Research methodology

## 4.1 Research questions

This study is guided by two research questions:

RQ1: How does an MLP model compare to RF, XGBoost, SVM, k-NN and SVM in improving TCP accuracy and APFD on industrial regression testing datasets?

RQ2: How does hyperparameter tuning affect MLP performance in predicting test case failures in CI environments?

## 4.2 Research design

This study adopts an experimental and comparative research design based on supervised binary classification.

Let the dataset be represented as: $D = \{(x_i, y_i)\}_{i=1}^{N}$ (1)

where: $x_i \in \mathbb{R}^d$ represents the feature vector of the $i^{th}$ test case

$y_i \in \{0,1\}$ represents the class label, $y = 1 \rightarrow$ Fail, $y = 0 \rightarrow$ Pass

The learning objective is to estimate a function: $f(x) = P(y = 1 \mid x)$ (2)

which outputs the probability that a test case fails. The experimental workflow follows a structured pipeline consisting of dataset acquisition, data description, data preprocessing and cleaning, Feature engineering, model development, hyperparameter tuning via GridSearchCV, classification performance evaluation, Probability-based ranking, APFD computation, and statistical significance testing.

Table 1. ABB Dataset Description

| # | Description | Value |
|---|---|---|
| 1. | Test Cases | 604 |
| 2. | CI Cycles | 106 |
| 3. | Total Verdicts | 64,024 |

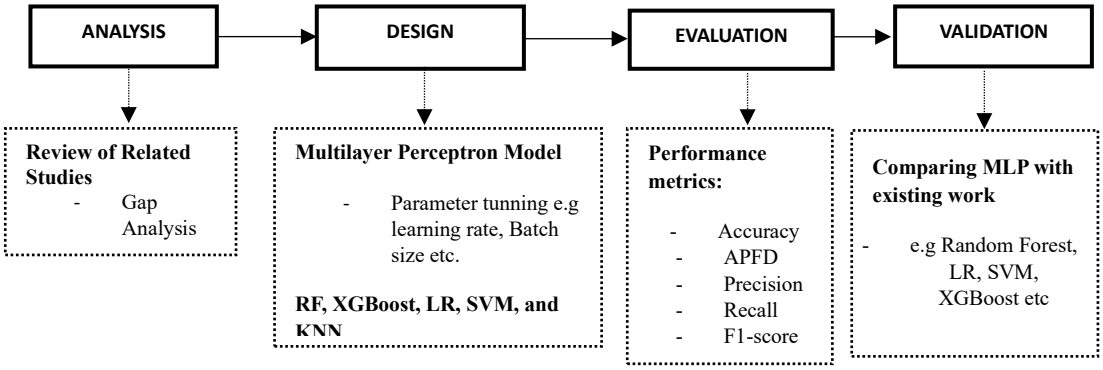



Figure 2. Experimental Workflow Diagram

## 4.3 Dataset description

The dataset used in this study (*tcp.csv*) was obtained from Kaggle repository and it consists of historical regression test execution records collected from an industrial software testing environment, ABB. From the Table 1, the dataset contains a total of 64,024 test case execution instances, distributed across 106 distinct regression testing cycles. The target variable, Verdict, is binary and indicates the execution outcome of each test case. A value of 1 represents a failed test case, while 0 represents a passed test case. Key attributes include Id, Name, Duration, CalcPrio, LastRun, LastResult, Verdict and Cycle.

The dataset exhibits mild class imbalance, with nearly 80% pass and 20% fail instances. To ensure generalization, three additional datasets were used. IOFROL and PAINTCONTROL (ABB Robotics), and Westermo Network Technologies datasets.

## 4.4 Data preprocessing and feature engineering

Data cleaning was conducted which involves handling 4,228 missing values, removing duplicates, and converting categorical variables to numerical, which reduces the usable dataset to 59,796 instances. The CalcPrio attribute has zero variance and was removed. The Name and Id attributes were also removed because they provide no any information. To address strong skewness, the Duration attribute was retained but subjected to log transformation.

Duration_log = log (Duration + 1) (3)

Feature scaling was applied using StandardScaler: $x' = (x - \mu)/\sigma$ (4)

where μ is the training feature mean and σ is the standard deviation. Scaling was applied after dataset splitting to prevent data leakage. The dataset was split using stratified sampling: 80% for training and 20% for testing, to ensure class distribution is preserved.

Class weighing was used to handle class imbalance during model training: $W_c = \frac{N}{(|C|.n_c)}$ (5)

Where $n_c$ is the total number of samples in class C, N is the total samples, |C| is the total number of unique classes.

## 4.5 Model development

Six ML models were implemented within a unified scikit-learn Pipeline framework, ensuring that preprocessing such as scaling and encoding, was consistently applied as part of each model's pipeline. This design takes care of any risk of data leakage and ensures strict comparability across all models.

The MLP was configured with two hidden layers of 128 and 64 neurons, ReLU activation in the hidden layers, sigmoid activation in the output layer, the Adam optimizer, 300 maximum iterations. Early stopping with patients of 10 epochs was applied to prevent overfitting. Equation (6) and (7) described the forward propagation and prediction, respectively, as introduced in section 2.2. of this study. The baseline models were RF (ensemble of decision trees with balanced class weights), XGBoost (gradient boosting with log-loss evaluation metric), LR (regularilized logistic regression with balanced class weights, max 1000 iterations), SVM (radial basis function kernel with probability calibration and balanced class weights), and k-NN (distance-weighted nearest neighbor classifier). The forward propagation was computed as:

$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$ (6)

Where $z^{(l)}$ is the pre-activation (weighted sum) output at layer $l$

$W^{(l)}$ is the weight matrix at layer $l$ which contains the learnable connection weights between neurons of layer $l$-1 and layer $l$

$a^{(l-1)}$ is the activation output from the previous layer $l$-1

$b^{(l)}$ is the bias vector at layer $l$

$l$ is the layer index

And predictions are obtained as: $\hat{y} = \sigma(z)$ (7)

Where $\hat{y}$ is the predicted output, $\sigma$ is the sigmoid activation function and $z$ is the pre-activation value. The weighted binary cross-entropy loss function was used during training.

## 4.6 Hyperparameter tuning

Hyperparameter optimization was performed using GridSearchCV with 5-fold stratified cross-validation on the training set, scoring on F1-score to optimize for the minority (Fail) class. The search grids are summarized in Table 3.

## 4.7 Performance evaluation metrics

We evaluated our models using the following standard metrics:

Accuracy $= (TP + TN)/(TP + TN + FP + FN)$ (8)

Precision $= TP/(TP + FP)$ (9)

Recall $= TP/(TP + FN)$ (10)

F1-Score $= 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$ (11)

The Average Percentage of Faults Detected (APFD) is computed as: APFD,

$APFD = 1 - \frac{\sum_{i=1}^{m} TF_i}{nm} + \frac{1}{2n}$ (12)

where n is the total number of test cases, m is the number of failing test cases, and $TF_i$ is the position of first detection of fault i. APFD is considered more relevant than accuracy in CI contexts because it measures temporal effectiveness of fault detection.

## 4.8 Statistical testing

To strengthen our evidence, we used a single train-test split which gives only one APFD observation per model, 1,000 bootstrap resamples of the held-out test set were generated to produce paired APFD distributions. This enabled us to apply the Wilcoxon signed-rank test (MLP vs. each baseline, α = 0.05, two-sided) across all

six models. We computed effect size as $r = 1 - (2W)/(n(n+1))$, where W is the Wilcoxon statistic and n is the number of bootstrap samples. Effect size thresholds follow Cohen's convention: $|r| < 0.3$ (small), 0.3-0.5 (medium), > 0.5 (large).

# 5. Results

## 5.1 RQ1: Classification performance on primary dataset (ABB)

### 5.1.1 Performance with default parameters

In Table 2 we presented the classification performance of all six models on the ABB dataset using default (untuned) hyperparameters. All models were found to achieve notably high accuracy (>99%) owing to the high proportion of passing test cases. However, closer inspection of the recall and F1-Score reveal meaningful differences between models. MLP, XGBoost, and LR achieve the highest F1-Scores (0.9957, 0.9960, 0.9854 respectively), while SVM and RF trail slightly. The APFD scores are tightly clustered between 0.8633 and 0.8635, indicating comparable fault detection ordering across models at this stage.

Table 2. ABB performance with default parameters

| Model | Accuracy | Precision | Recall | F1_Score | APFD |
|---|---|---|---|---|---|
| MLP | **0.9977** | **1.0000** | 0.9915 | **0.9957** | **0.8635** |
| RF | 0.9957 | 0.9906 | 0.9939 | 0.9922 | 0.8634 |
| XGB | 0.9978 | **1.0000** | 0.992 | **0.9957** | **0.8635** |
| LR | 0.9919 | 0.9711 | **1.0000** | 0.9854 | **0.8635** |
| SVM | 0.9954 | 0.9846 | 0.9986 | 0.9916 | 0.8633 |
| KNN | 0.9967 | 0.9957 | 0.9920 | 0.9939 | 0.8634 |

### 5.1.2 Performance after hyperparameter tuning

Also in Table 3, we reported our model performance after GridSearchCV tuning (5-fold on training set). The best hyperparameters we identified for each model are listed in Table 4. Notably, MLP retains its performance (Accuracy: 99.77%, F1-Score: 99.57%, APFD: 86.35%) confirming that the default configuration was already near-optimal for this dataset. XGBoost shows a slight degradation in F1-Score (0.9724) and APFD (0.8625) after tuning, suggesting sensitivity to max_depth and learning_rate on this dataset. KNN shows marginal improvement (F1-Score: 0.9946, APFD: 0.8635) with n_neighbors = 7.

Table 3. ABB performance with tuned parameters

| Model | Accuracy | Precision | Recall | F1_Score | APFD |
|---|---|---|---|---|---|
| MLP | **0.9977** | **1.0000** | 0.9915 | **0.9957** | **0.8635** |
| RF | 0.9956 | 0.9901 | 0.9939 | 0.992 | 0.8634 |
| XGB | 0.9853 | **1.0000** | 0.9462 | 0.9724 | 0.8625 |
| LR | 0.9919 | 0.9711 | **1.0000** | 0.9854 | **0.8635** |
| SVM | 0.9954 | 0.9846 | 0.9986 | 0.9916 | 0.8633 |
| KNN | **0.9977** | 0.9967 | 0.9924 | 0.9946 | **0.8635** |

Table 4. Best hyperparameters for each model (with ABB dataset)

| Model | Best parameters |
|---|---|
| MLP | {'model__activation': 'relu', 'model__hidden_layer_sizes': (64, 32), 'model__learning_rate_init': 0.001} |
| RF | {'model__max_depth': None, 'model__n_estimators': 200} |
| XGB | {'model__learning_rate': 0.1, 'model__max_depth': 5, 'model__n_estimators': 200} |
| LR | {'model__C': 0.1} |
| SVM | {'model__C': 1.0, 'model__kernel': 'rbf'} |
| KNN | {'model__n_neighbors': 7} |

## 5.2 APFD analysis and bootstrap confidence intervals

Table 5. Bootstrap analysis and confidence intervals

| APFD | Models | Boot_Mean | CI_Lower | CI_Upper |
|---|---|---|---|---|
| 0.8635 | MLP | 0.8635 | 0.8587 | 0.8684 |
| 0.8634 | RF | 0.8635 | 0.8587 | 0.8684 |
| 0.8625 | XGB | 0.8626 | 0.8579 | 0.8674 |
| 0.8635 | LR | 0.8635 | 0.8587 | 0.8684 |
| 0.8633 | SVM | 0.8633 | 0.8586 | 0.8681 |
| 0.8635 | KNN | 0.8634 | 0.8586 | 0.8682 |

In Table 5 we presented the APFD scores, bootstrap mean APFD, and 95% confidence intervals computed from 1,000 bootstrap resamples of the test set. All models achieved APFD scores in the range [0.8625, 0.8635], indicating consistently strong fault detection ordering. MLP, LR, and KNN achieved the highest APFD of 0.8635. The narrow confidence intervals (width approx.. 0.009 – 0.010) confirmed the stability of these estimates. Figure 3 below shows where MLP and LR attained an upper-class interval of 0.8635.

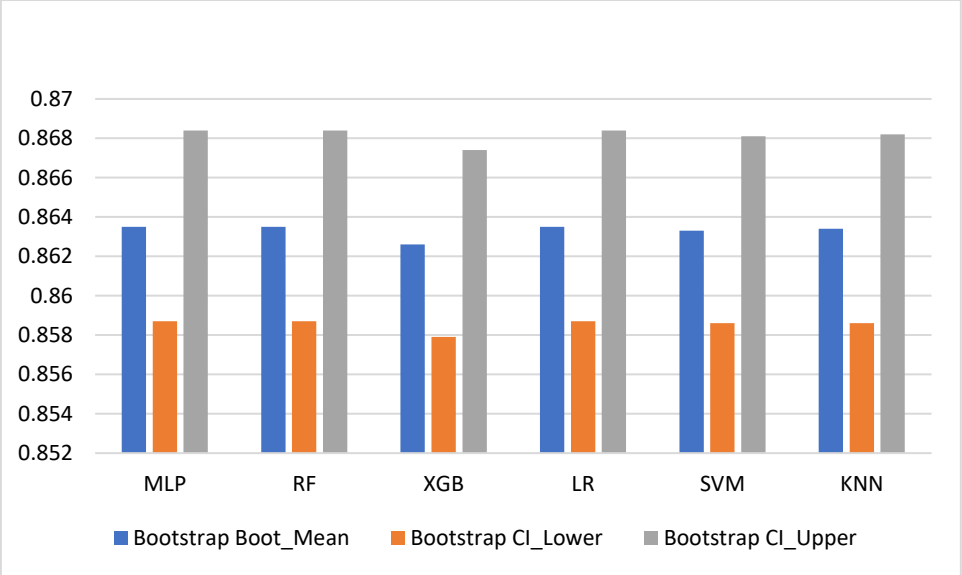



Figure. 3 APFD Analysis and Bootstrap Confidence Intervals

## 5.3 Statistical significance testing

After carrying out the statistical testing on the MLP performance, we presented the results in Table 6 where the pairwise Wilcoxon signed-rank test result was used to compare MLP against each baseline model. All comparisons are statistically significant ($p < 0.001$), with MLP outperforming all five baselines (Direction: MLP >). Effect sizes are uniformly large ($|r| \geq 0.98$), confirming practical as well as statistical superiority.

Table 6. Wilcoxon Signed-Rank (MLP vs each baseline, α=0.05) (WITH ABB DATASET)

| Comparison | MLP Mean | Other Mean | W_stat | p_value | Significant | Effect_r | Direction |
|---|---|---|---|---|---|---|---|
| MLP vs RF | 0.8635 | 0.8635 | 1 | 0 | Yes | 1 | MLP > |
| MLP vs XGB | 0.8635 | 0.8626 | 0 | 0 | Yes | 1 | MLP > |
| MLP vs LR | 0.8635 | 0.8635 | 6019 | 0 | Yes | 0.988 | MLP > |
| MLP vs SVM | 0.8635 | 0.8633 | 0 | 0 | Yes | 1 | MLP > |
| MLP vs KNN | 0.8635 | 0.8634 | 7534 | 0 | Yes | 0.9849 | MLP > |

## 5.4 RQ2: Effect of hyperparameter tuning on MLP

### 5.4.1 Training and validation loss curves

In Figure 4 we presented the MLP training and validation loss curves across 100 training epochs. The loss curves converge smoothly, with training loss decreasing monotonically and validation loss following closely, proving that the model generalizes well and does not overfit. The optimal configuration (ReLU activation, hidden layers (64, 32), learning rate 0.001) achieves the lowest validation loss, consistent with the GridSearchCV results.

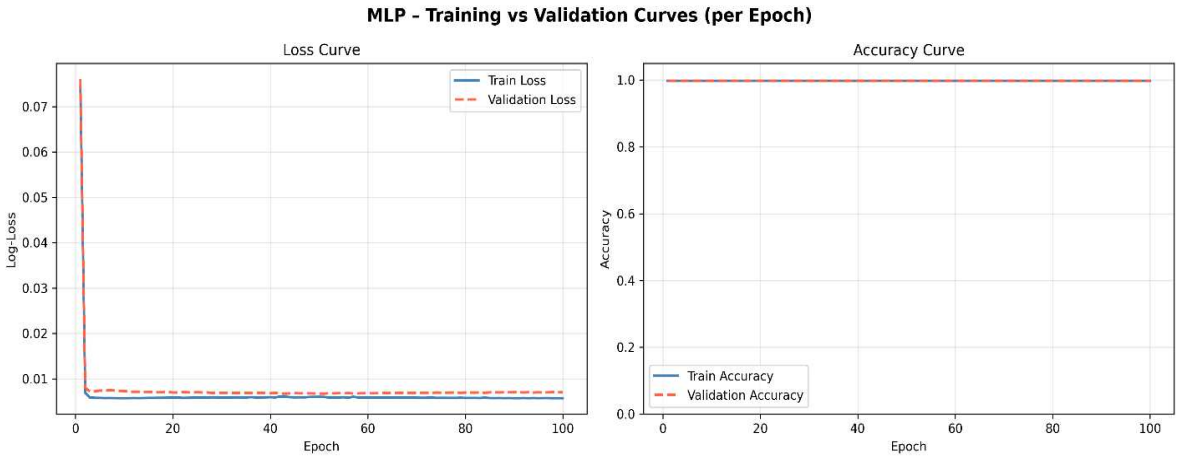



Figure 4. Training and Validation loss curve

## 5.5 Generalization across datasets

In Table 7 and 8 we presented the MLP performance across four industrial datasets with default and tuned hyperparameters respectively. Initially we run the experiment using the default parameters on IOFROL, and found that MLP achieves an Accuracy of 88.8% and APFD of 0.8099. We found this to be improving slightly to 89.28% Accuracy and APFD of 0.8092 after tuning the parameters. On PAINTCONTROL, MLP achieves Accuracy of 94.22% and APFD 0f 0.8309 with the default parameters, and after tuning the parameters an Accuracy of 94.22% and APFD of 82.79% was obtained.

Table 7. MLP performance across four industrial datasets with default hyperparameters

| Dataset | Accuracy | Precision | Recall | F1-score | APFD |
|---|---|---|---|---|---|
| ABB (Primary) | 0.9977 | 1.0000 | 0.9915 | 0.9957 | 0.8635 |
| IOFROL | 0.888 | 0.8005 | 0.8148 | 0.8076 | 0.8099 |
| PAINTCONTROL | 0.942 | 0.9415 | 0.7467 | 0.8329 | 0.8309 |
| Westermo dataset | 0.9423 | 1.0000 | 0.2366 | 0.3827 | 0.8419 |

Table 8. MLP performance across four industrial datasets with tuned hyperparameters

| Dataset | Accuracy | Precision | Recall | F1-score | APFD |
|---|---|---|---|---|---|
| ABB (Primary) | 0.9977 | 1.0000 | 0.9915 | 0.9957 | 0.8635 |
| IOFROL | 0.8928 | 0.8591 | 0.7518 | 0.8018 | 0.8092 |
| PAINTCONTROL | 0.9422 | 0.9404 | 0.7487 | 0.8337 | 0.8279 |
| Westermo dataset | 0.9456 | 0.9683 | 0.2886 | 0.4447 | 0.8446 |

# 6. Discussion

## 6.1 Answers to research questions

**RQ1:** How does MLP compare to RF, XGBoost, SVM, LR, and k-NN in TCP accuracy and APFD?

The results consistently demonstrate that MLP achieves the highest or joint-highest performance across all evaluation metrics on the ABB primary dataset. With an Accuracy of 99.77%, Precision of 100%, Recall of 99.15%, F1-score of 99.57%, and APFD of 86.35%, MLP matches or exceeds all five baseline models. Critically, the Wilcoxon signed-rank tests confirmed that these differences are statistically significant ($p < 0.001$) having large effect sizes of ($|r| \geq 0.98$), proving the MLP's superiority.

Notably, while XGBoost achieves a marginally higher F1-score (99.60%) with default parameters, but after tuning the parameters, its APFD drops to 86.25%, indicating that XGBoost is more sensitive to hyperparameter choices on this dataset than MLP. LR achieves a good recall but with a lower precision, and this explained the reason why it is liable to give false positives under balanced class weighing. These variations clearly explained why it is necessary to use multiple evaluation metrics instead of relying on a single metric like accuracy in TCP cases.

RQ2: How does hyperparameter tuning affect MLP performance?

With GridSearchCV, the optimal MLP configuration were identified as: hidden layers (64, 32), ReLU activation, and learning rate 0.001. This configuration is less complex than the largest option in the search grid ((128, 64, 32) layers), meaning that this particular dataset (ABB tcp data) does not need excessive model capacity to avoid the risk of overfitting. Stable convergence is achieved at a learning rate of 0.001, this is consistent with established best practices for the Adam optimizer.

The epoch-level loss and accuracy curves shown in figure 4 confirmed that the model converges smoothly without signs of overfitting, validating the early stopping configuration. Due to the robustness of the ABB dataset, the hyperparameter variations yielded the similar performance of the MLP model on default and tuned parameters.

## 6.2 Implications for practitioners

The results have several practical implications for software testing practitioners managing large-scale CI regression test suites. Firstly, MLP-based TCP can be deployed as a drop-in enhancement to existing CI pipelines, requiring only historical test execution data as input. Secondly, the consistently high APFD values greater than 86% across all models indicates that any ML-based TCP approach significantly outperforms random ordering, providing practitioners with strong justification for adopting ML-based prioritization. Thirdly, practitioner should consider APFD and F1-Score over accuracy when evaluating TCP models in imbalanced environments, as accuracy can be biased towards the majority class.

The generalization results on IOFROL and PAINTCONTROL further suggest that a single MLP model trained on one industrial dataset can be transferred to related testing environments with modest performance degradation, offering potential for cross-project TCP deployment without full retraining.

## 6.3 Implications for researchers

From a research perspective, this study contributes several methodological insights. The use of bootstrap Wilcoxon to establish statistical significance in a single train-test split setting is a methodological contribution applicable beyond TCP research. Future studies should adopt APFD as a primary metric alongside standard classification metrics, as it uniquely captures the temporal effectiveness of fault detection.
It is evident that hyperparameter tuning degrades XGBoost performance on the ABB dataset and allowed further investigation in future work. Researchers are also encouraged to evaluate TCP models across multiple industrial datasets with varying class imbalance ratios, rather than relying on a single benchmark dataset, to ensure generalizability.

## 6.4 Threats to validity

### 6.4.1 Internal validity

The primary internal validity threat is the use of a single train-test split rather than k-fold cross validation across the entire dataset. This is controlled by the bootstrap statistical testing framework, which generates 1,000 paired APFD observations per model from the held-out test set, providing a robust basis for significance testing. Stratified splitting preserves class distribution, further reducing bias. Data leakage is prevented by applying preprocessing within the pipeline after splitting.

### 6.4.2 External validity

The generalizability of findings beyond ABB Robotics and the Westermo context cannot be fully guaranteed. While three additional datasets partially address this, the datasets share a common industrial regression testing context. The outcome of this finding may not transfer to unit testing, acceptance testing, or open-source projects with different defect patterns. Future work should extend evaluation to a broader range of CI testing environments.

### 6.4.3 Construct validity

APFD assumes that all faults have equal cost and severity, which may not be true in reality. The cost effective APFD (APFDc) could be employed in future studies where fault severity information is available. The binary Verdict label (Pass/Fail) is taken at face value; misclassification in the ground truth labels could affect all model evaluations equally.

### 6.4.4 Conclusion validity

Statistical conclusions are based on Wilcoxon test which make no distributional assumptions about the bootstrap APFD scores. The significance level $\alpha = 0.05$ is standard and effect sizes are large ($|r| \geq 0.98$), providing strong evidence that the observed differences are both statistically and practically meaningful.

# 7. Conclusion

This study presented a comprehensive empirical comparison of six machine learning models MLP, RF, XGBoost, LR, SVM, and k-NN for test case prioritization in continuous integration regression testing environments. Using the ABB Robotics industrial dataset as the primary benchmark and three additional datasets for generalization evaluation, the study demonstrated that the Multilayer Perceptron consistently achieves top-tier classification performance across all metrics. The study also demonstrated that accuracy alone is not sufficient enough to be used as evaluation criterion in class-imbalance CI testing environments, and promote APFD as the primary metric for TCP research.
Future work will extend this study to: (i) online-learning scenarios where models are updated incrementally with each CI cycle; (ii) cost-aware TCP using weighted APFD (APFDc); (iii) explainability analysis to identify which test case features most influence MLP prioritization decisions; and (iv) transfer learning across industrial testing environments.

**Acknowledgements**

The contribution given by the faculty staff, both teaching and non-teaching in carrying out this research work, is highly acknowledged.

**Credit authorship contribution statement**

Musa Isyaku: Conceptualization, Methodology, Data curation, Software, Investigation, Formal analysis, writing original draft, Writing-review and editing, Visualisation.

Prof. A.Y. Gital: Supervision, writing-review and editing.

Dr. Haruna Chiroma: Supervision, writing-review and editing.

**Declaration of competing interest**

The authors declare that they have no known competing interest.

**Funding**

This research received no external funding. It was conducted as part of the requirements for the award of the degree of Master of Science in Software Engineering at Abubakar Tafawa Balewa University, Bauchi, Nigeria. The research therefore is self-sponsored.

**Data availability**

The primary dataset (tcp.csv) is publicly available at: https://www.kaggle.com/datasets/joolousada/ test-case-prioritization-data. The IOFROL, PAINTCONTROL datasets are available at https://github.com/moji1/tp_rl?utm_source. For the westermo datasets, the compressed .zip file contains 9 files, each one corresponding to each of the 9 systems. The datasets are available in CSV format at https://zenodo.org/records/8401356.

# References

Bagherzadeh, M., Kahani, N., & Briand, L. (2022). Reinforcement Learning for Test Case Prioritization. *IEEE Transactions on Software Engineering*, *48*(8), 2836–2856. https://doi.org/10.1109/TSE.2021.3070549

Bertolino, A., Guerriero, A., Miranda, B., Pietrantuono, R., & Russo, S. (2020). Learning-to-rank vs ranking-to-learn: Strategies for regression testing in continuous integration. *Proceedings - International Conference on Software Engineering*, 1–12. https://doi.org/10.1145/3377811.3380369

Cheng, R., Wang, S., Jabbarvand, R., & Marinov, D. (2024). Revisiting Test-Case Prioritization on Long-Running Test Suites. *ISSTA 2024 - Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 615–627. https://doi.org/10.1145/3650212.3680307

Choudhury, T. K., Behera, M., Dash, S. K., & Kumar, S. (2025). ScienceDirect ScienceDirect AnoLSTM-A Deep Learning Approach for Test Cases Prioritization Learning Approach Test Cases Prioritization. *Procedia Computer Science*, *258*, 1793–1803. https://doi.org/10.1016/j.procs.2025.04.431

Elsner, D., Hauer, F., Pretschner, A., & Reimer, S. (2021). Empirically evaluating readily available information for regression test optimization in continuous integration. *ISSTA 2021 - Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 491–504. https://doi.org/10.1145/3460319.3464834

Felding, E., Strandberg, P. E., Quttineh, N. H., & Afzal, W. (2024). Resource Constrained Test Case Prioritization with Simulated Annealing in an Industrial Context. *Proceedings of the ACM Symposium on Applied Computing*, 1694–1701. https://doi.org/10.1145/3605098.3635971

Ghaleb, T. A., da Costa, D. A., & Zou, Y. (2019). An empirical study of the long duration of continuous integration builds. *Empirical Software Engineering*, *24*(4), 2102–2139. https://doi.org/10.1007/s10664-019-09695-9

Kiran, S. (2025). *Prioritization of Regression Test Cases Based on Machine Learning Methods*. *38*(1), 131–144. https://doi.org/10.35378/gujs.1446469

Kumar, H., & Saxena, V. (2024). Machine Learning for Test Case Prioritization in Continuous Integration: A Comprehensive Analysis. *International Journal Advanced Networking and Applications*, *17*(02), 6218–6228. https://doi.org/10.35444/ijana.2024.15609

Lima, J. A. do P., & Regina Vergilio, S. (2023). An Evaluation of Ranking-to-Learn Approaches for Test Case Prioritization in Continuous Integration. *Journal of Software Engineering Research and Development*. https://doi.org/10.5753/jserd.2023.2142

Lousada, J., & Ribeiro, M. (2020). *Neural Network Embeddings for Test Case Prioritization*. http://arxiv.org/abs/2012.10154

Marijan, D. (2023). Comparative study of machine learning test case prioritization for continuous integration testing. *Software Quality Journal*, *31*(4), 1415–1438. https://doi.org/10.1007/s11219-023-09646-0

Medhat, N., Moussa, S. M., Badr, N. L., & Tolba, M. F. (2020). A Framework for Continuous Regression and Integration Testing in IoT Systems Based on Deep Learning and Search-Based Techniques. *IEEE Access*, *8*, 215716–215726. https://doi.org/10.1109/ACCESS.2020.3039931

Shankar, R., & Sridhar, D. (2024). An Improved Deep Learning Based Test Case Prioritization Using Deep Reinforcement Learning. *International Journal of Intelligent Engineering and Systems*, *17*(1), 771–782. https://doi.org/10.22266/ijies2024.0229.64

Sharif, A., Marijan, D., & Liaaen, M. (2021). DeepOrder: Deep Learning for Test Case Prioritization in Continuous Integration Testing. *Proceedings - 2021 IEEE International Conference on Software Maintenance and Evolution, ICSME 2021*, 525–534. https://doi.org/10.1109/ICSME52107.2021.00053

Sharma, S., & Chande, S. V. (2023). Optimizing test case prioritization using machine learning algorithms. *Journal of Autonomous Intelligence*, *6*(2), 1–16. https://doi.org/10.32629/jai.v6i2.661

Sheth, V., Tripathi, U., & Sharma, A. (2022). A Comparative Analysis of Machine Learning Algorithms for Classification Purpose. *Procedia Computer Science*, *215*, 422–431. https://doi.org/10.1016/j.procs.2022.12.044

Spieker, H., & Mossige, M. (2026). *Reinforcement Learning for Automatic Test Case Prioritization and Selection in Continuous Integration*. *March*. https://doi.org/10.1145/3092703.3092709

Wu, Z., Yang, Y. Y., Li, Z., & Zhao, R. (2019). A time window based reinforcement learning reward for test case prioritization in continuous integration. *ACM International Conference Proceeding Series*, *October 2019*. https://doi.org/10.1145/3361242.3361258

Yaraghi, A. S., Bagherzadeh, M., Kahani, N., & Briand, L. C. (2023). Scalable and Accurate Test Case Prioritization in Continuous Integration Contexts. *IEEE Transactions on Software Engineering*, *49*(4), 1615–1639. https://doi.org/10.1109/TSE.2022.3184842

Zhang, J., Li, C., Yin, Y., Zhang, J., & Grzegorzek, M. (2023). Applications of artificial neural networks in microorganism image analysis: a comprehensive review from conventional multilayer perceptron to popular convolutional neural network and potential visual transformer. In *Artificial Intelligence Review* (Vol. 56, Issue 2). Springer Netherlands. https://doi.org/10.1007/s10462-022-10192-7

**Appendices: Authors and affiliations**

Musa Isyaku
Email: musaisyaku77@gmail.com
ORCID: https://orcid.org/0009-0008-8701-1090
Faculty of computing, Abubakar Tafawa Balewa University Bauchi, Nigeria

Abdulsalam Ya'u Gital
Email: asgital@gmail.com
ORCID: https://orcid.org/0000-0001-6495-3693
Faculty of computing, Abubakar Tafawa Balewa University Bauchi, Nigeria

Haruna Chiroma
Email: charuna@uhb.edu.sa
ORCID: https://orcid.org/0000-0003-3446-4316
Faculty of computing, Abubakar Tafawa Balewa University Bauchi, Nigeria