Designing a Multi-class Classification Optimized Model for Requirement Based Test Case Prioritization

Aishwaryarani Behera
Computer Science and Engineering
KIIT Deemed to be University
Bhubaneswar-751024, India
beheraaishwarya2@gmail.com

Arup Abhinna Acharya
Computer Science and Engineering
KIIT Deemed to be University
Bhubaneswar-751024, India
aacharyafcs@kiit.ac.in

Sanjukta Mohanty
Computer Science and Engineering
Odisha University of Research and Technology
Bhubaneswar-751003, India
mailtorani.sanjukta @gmail.com

Namita Panda
Computer Science and Engineering
KIIT Deemed to be University
Bhubaneswar-751024, India
npandafcs@kiit.ac.in

2024 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC) | 979-8-3503-7018-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/ASSIC60049.2024.10507913

***Abstract*— Most of the times we have to test out the entire application functionality, for any code modification done to cater the need of the larger audiences or any bug fixes. This results to consumption of time and the effort to retest the application by executing all of the test suites. In such cases more often regression testing comes to the rescue, where prioritization techniques are being used to overcome the limitations of regression testing. Test Case Prioritization (TCP) usually means categorically ranking some test cases higher than others. The main goal of the TCP is to find the fault early in the testing process by scheduling the test cases with the help of Requirement based Test Case Prioritization Technique (RTCP) in order to increase effectiveness of regression testing. In this research study, we have developed a supervised machine learning based RTCP mechanism where the significant business requirement based relevant feature are considered for scheduling test cases according to their priority label as high, medium and low multi classes. The proposed model is validated with the two datasets collected from internet sources. The machine learning classifier k-Nearest Neighbor (K-NN), Decision Tree (DT), Random Forest (RF) Bagging and Boosting algorithm are utilized to evaluate the features for the test case prioritization. To enhance the performance of the model some hyper parameter settings are altered and found a drastic change in the results. To achieve the low cost and high fault detection rate in RTCP, an optimized model is designed which is validated by datasets and altered parameter settings. The experimental result demonstrates that RF classifier achieved the best performance among the other classifiers for predicting the prioritized test cases early to reduce the cost and time required for regression testing.**

Keywords— ***Machine Learning, multi-class classification, Test case prioritization, Regression testing, Requirement based testing, Hyper-Parameter Tuning***

I. INTRODUCTION

In the field of software development, we always strive for developing high quality application keeping the cost overheads to a minimal. Thus, software testing plays an integral role in ensuring the highest quality of the software [1]. Out of many available software testing practices, regression testing stands out in terms of acknowledging application stability and functionality. Regression testing is a procedure which is used to make sure that any source code changes or modifications have no negative effects, on the existing source code [3]. At the same time, while ensuring the quality we need to keep an eye on the overall software testing cost inflicted in terms of resources and time. Therefore, certain test strategy needs to be implemented in order to minimize testing cost without any compromise on quality. One such strategy is TCP. TCP involves execution of highest priority test scenario first. The traditional methods used in the context of TCP are Coverage based (statement, branch, functional) testing, Model-based testing, History-based testing, Cost-based testing and Time-based testing. In Coverage-based technique a rigorous study of source code is needed which is a very time intensive procedure for TCP [8]. Similarly, the Model-based TCP technique may not be efficient enough in providing satisfactory fault detection results. Apart from that its test selection criteria are not good enough in ordering the test cases [15]. In History-based testing the major drawback is, the fault age gets older if a test run fails to find a flaw [15]. In Time-based test case prioritization, each test case is regarded as autonomous and not reliant on the sequence of execution [8]. According to Zhang et al. [7] the sole disadvantage of ILP-based strategy is time spent in analysis for big test suites. Since TCP implicitly results in cost reduction, so Cost-based TCP need

not considered [15]. Hence to overcome the short comings of different TCP methods, Requirement based TCP (RTCP) is helpful in such cases. The main idea of RTCP is to find the most severe faults early in the testing process without accessing source code. The researchers employed machine learning techniques (MLT) in TCP to increase fault detection rate. With machine learning help using HPT, RTCP becomes easier and simpler to categorize test instances. Few research has been done in RTCP using MLT with HPT [11]. Where clustering technique is used to group the test cases to enhance the TCP. But choosing the clustering size has always been challenging and moreover, no research work has been done till now to classify the test cases in multiple labels like high, low and medium class labels for RTCP. In our proposed method, we classify test cases on the basis of several features [11] in overcoming the shortcomings of the traditional TCP method. To evaluate the features, four supervised machine learning classifiers such as KNN, DT, RF Bagging and Boosting are considered. The proposed approach has been validated with a "MIS with Priority" dataset. From the experiments it is found that among the five classifiers RF achieve the highest accuracy.

The remaining of the paper is organized as follows. In section 2, background information is provided. The related studies are covered in section 3. Section 4 representing findings and comments. Fragment 5 contains the proposed model. The implementation work is presented in section 6. Section 7 represents the conclusion and future work.

II. BACKGROUND DETAILS

In this section, the concepts pertaining to requirement-based test case prioritization and machine learning techniques are explained in details

A. Software Testing

It is a technique for ensuring that a software programmed compiles with the requirements [1]. The intent of doing software testing is to make application bug free, to minimize the cost and time needed for development and enhance the performance. Different types of software testing are adopted for fulfilling the specific objective and regression testing is one of them which is the most essential testing strategy for validating a modification introduced during the maintenance stage [2].

B. Regression Testing

Regression testing is a crucial software quality assurance procedure to ensure that modifications to the software code have not negatively impacted already-existing functionalities. There are three forms of regression testing. Retesting all test cases, regression test selection, and prioritizing test cases. In this research we have emphasized on the prioritization of test cases as the objective is to detect the fault as early as possible [2].

C. Test Case Prioritization

As software retesting is a time intensive process so, to get the better of this limitation a good process of test case prioritization is required which can rank the test cases in order of importance according to some criteria [1]. Prioritizing test cases has one benefit over selection that it enables continual testing while constantly concentrating on the most crucial test cases, even when resources run out or all test cases are processed. Different techniques are available for prioritizing the test cases from the large test suite which are well explained in related work section but out of all the existing methodologies the machine learning approach is the best method as it predicts the most prioritized test suite early without going in details to the source codes.

D. Machine Learning

The term "machine learning (ML)" refers to teaching a machine to act like a human being. The application of ML approaches is optimization jobs, when a specific optimization goal shall be accomplished effectively. There are various kinds of ML approaches. supervised, unsupervised and reinforcement ML technique. supervised and unsupervised relay on the idea of training data. These situations in supervised learning include labels. Whereas unlabeled data is used in unsupervised approaches. One form of unsupervised method is clustering signal [5]. The input data is transformed into a feature matrix representation before using a particular machine learning technique. Each feature has a unique quality, such as the sender of spam email. A particular data instance is represented by the set of all features. The goal of ML algorithms is to identify patterns in the data based on the feature representation.

E. Hyper -Parameter Tuning

The performance of ML algorithms for prediction can be significantly impacted by HP tweaking. A ML algorithm's HPs are often configured through a process of trial and error. Finding a decent set of numbers manually might take a lot of time, depending on how long the ML algorithm being used needs to train. Because of this, current research on HP for ML algorithms has focused on improving HP tuning strategies. The HP process is typically viewed as an optimization (black box) problem, with the algorithm's objective function being the model's capacity for prediction [15].

III. RELATED WORK

In this section we have explored the most common methods, designed by research practitioners to prioritize the test cases. The different approaches used are: 1. Code based technique, 2. Model based Technique, 3. Requirement based Technique, 4. History based technique, and 5. Machine learning based technique.

Aggrawal et al. [9] has suggested a paradigm for version-specific regression testing that best achieves 100% code coverage. C++ has been used to implement the algorithm. Also, they have suggested a prioritization method that produces changed code coverage as quickly as is practical.

Srikant at al [12] has proposed a prioritization technique based on requirement and risk factor. They have extended the work of previous paper [2]. They have taken a small software with a pseudo-code as a case study. They have Proposed a novel approach which explore two prioritization factors.

Arafeen et al. [11] has checked whether the requirement-based clustering strategy may increase the efficiency of the test case prioritization technique. Here the authors have taken several java programs that have numerous version and requirement document. Clustering the requirement, they have used k-means algorithm. Result indicates that prioritizing test case

using requirement-based clustering approach boost the performance of test case order with respect to detect the fault early.

Mohanty et al. [2] has identified the current method and problems with model-based priority in CBSS. To do test case ranking, the authors have examined test case prioritization in the code base, model base, requirement base, and its application. Nine items were chosen through a multistage screening process. the (APFD) metric was applied here. According to the experimental findings, prioritization strategy may increase the fault rate identification and the speed at which the defect can be discovered. The concept of code-based testing, component behavior, interaction, and compatibility have all been discussed in detail by the author.

Younghwan et al. [13] has proposed a technique called AFSAC, a test case prioritizing method established on historical information, which may be used to efficiently gather failure data. They have categorized the technique into two phases. To validate their strategy, they have conducted an empirical study on two Apache freely available software initiatives, namely Camel and Tomcat. The empirical study's findings demonstrate that their technique more successfully informs testers and developers about failures than alternative methods of prioritization.

Fan et al. [3] has proposed an idea spontaneous path generation. Basically, independent path generation algorithm and improvised depth -first-search algorithm plays the vital role to automate generating the test path as well as optimized the path. As a result, the redundancy and cycle path problem have been solved. Test path is also used for Prioritizing the design. the future work can be reduced the number of invalid test path.

Sharif et al. [5] has developed a new method for early-stage fault prediction in CI. The gap in the previous paper is that the existing test cases prioritization techniques are not going to handle this enormous executing history of data. This kind of limitation reduces the effectiveness of TCP. To overcome this problem the author has proposed deep order, deep learning approach in CI. For test result analysis the author has checked fault detection effectiveness, time effect, effect of longer test history and prediction accuracy. They have mentioned two types of related work ML based TCP and non-ML based TCP.

Spieker et al. [6] has developed RETES method for prioritizing and selecting in CI, so that the time will be minimized among developer response on fail test cases and code comments. here the author has used the reinforcement machine learning technique to prioritized the test cases using failure history, previous last execution and their duration. In essence, the author has concentrated on ranking the testcases, as well as test case scheduling inside a single CI cycle.

Lachmann et al. [7] has focused different “machine learning” technique in TCP. The author has utilized Support Vector Machine (SVM) rank and test case description to prioritize the test cases. The author has evaluated their approach using two different subject systems. (a) body comfort system (BCS) and (b) automotive Industry data. They proved that their approach shows better result comparison to any other methods. Also, to check the efficiency of their approach they have applied the APFD and K-fold techniques.

Lachmann et al. [8] have focused on how different machine learning techniques were utilized to rank the test instances and validate the model using three datasets. The absence of source code is this paper’s greatest strength. To judge the technique’s effectiveness, the authors have applied APFD technique. They discovered that their results outperformed those of any other strategy having accuracy is of 75%.

TABLE I. SUMMARY OF LITERATURE REVIEW OF TCP USING MACHINE LEARNING TECHNIQUES

Author details	Techniques used	Advantages	Limitation
Aggrawal et al. [9], 2004	Code based, Regression Test Selection Algorithm, c++	It avoids the expense and labor of running additional test cases.	Lack of code prevents execution.
Srikanth et al. [12], 2015	Requirement based, factor-based algorithm, APFD	It enhances the fault detection rate.	If your project is very large, it can be very difficult to keep track.
Mohanty et al. [2], 2011	Model based, APFD, RPd	Shows compatibility of components.	Integrating the component s which may leads to affect the quality of software.
Younghwan et al. [13], 2016	History based, AFSAC	limitation	There are not many subject projects.
Arafeen et al. [11], 2013	Requirement based, k-means clustering.	effectiveness of test case order in terms of early fault detection.	The no. of cluster selected may have negative impact.
Lachmann et al. [7], 2016	SVM Rank, APFD, k-fold	Run time is good, detect fault early	Lack of test case description gives very low result.
Lachmann et al. [8], 2018	SVM Rank, KNN, logistic regression, ensemble learning, neural network, APFD.	Source code is not required.	Absence of project specific feature may affect the result.
Spieker et al. [6], 2017	RETES, reinforcement ML.	From failure history and previous last execution can prioritize the test cases.	Difficult to find out appropriate feature.
Sharif et al. [5], 2021	deep learning, APFD	enormous execution history of data.	Very complex method
Fan et al. [3], 2021	activity diagram, improved depth-first-search algorithm.	automate generating the test path as well as optimized the path.	Number of invalid test path are there.

IV. DISCUSSION AND FINDINGS

We have included a lot of current prioritization techniques in our literature review, both with and without the use of machine learning technique. From the comparative study, it was found that a lot of research has previously been done on the subject of traditional prioritization method [17-19]. The disadvantage of the traditional methods is unable to prioritize the test cases without presence of the origin code. We have studied that black box testing is a different field. It includes

requirement- and history-based testing. Code is not necessary in this method to order the test cases. This motivated us to deep dive into this topic by using requirement factors required for ranking the test cases using Hyper-Parameter Tuning (HPT). The remainder of the section we will explain in detail how requirement testing is prioritized using hyper-parameter tuning in machine learning technique.

V. PROPOSED TEST CASE PRIORITIZATION APPROACH

The detailed explanation of the suggested methodology is provided in this section. A machine learning model using Hyper-Parameter Tuning (HPT) is used to determine the most important test cases. With training data as the input, classifiers are applied, followed by hyper parameter adjustment to choose the optimum model, accuracy is calculated, and ultimately prioritized test cases are scored as high, medium, or low. The proposed methodology for TCP is depicted in Fig.1

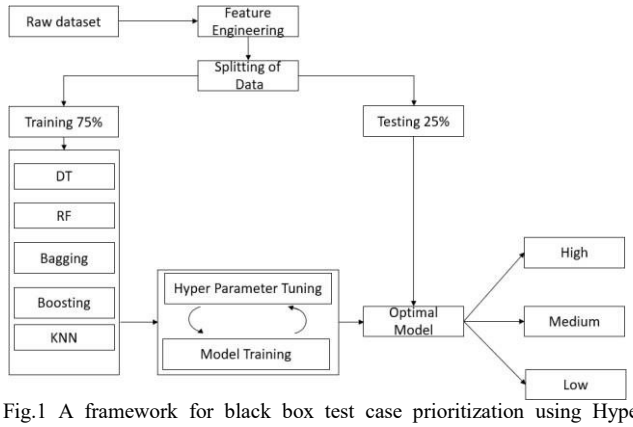

Fig.1 A framework for black box test case prioritization using Hyper-Parameter Tuning in Machine Learning

A. Dataset

In this paper, we have considered two datasets like "car-lease" and "MIS System with priority," which are available on Kaggle.com [16]. The first dataset “car lease” comprises 2000 instances and 6 numbers of features. Similarly, “MIS System with priority” consists of 1314 instances and 8 number of features.

B. Feature Preprocessing

In this phase, some of the data points in the dataset were missing so, mean of the variables were computed for replacing the missing value and then the min-max normalization procedure is adopted to normalize data values and make ready for evaluating a ML model for RTCP.

C. Parameter tuning with ML classifiers

The supervised machine learning classifiers use a labelled training set to identify new observation’s category amongst bunch of other categories [20-22]. In previous studies, even well- known classification algorithms failed to achieve a minimum accuracy to be eligible for scheduling TCP. To overcome these issues the parameters of the classifier are tuned to obtain a high performance in prioritizing the test cases as High, Medium and Low.

VI. EXPERIMENTAL RESULT

This section finds out the classification results in terms of ranked test cases such as low ranked case, medium ranked test case and high ranked test case. With the help of python packages like matplotlib, pandas and NumPy the proposed approach is implemented which allow for the creation of visually appealing and understandable presentations of the code flow. The efficiency of results of different classifiers before and after HPT are represented in Table IV.

TABLE II. PERFORMANCE OF CLASSIFIERS WITH OR WITHOUT HYPER-PARAMETER TUNING

Sl. No	Classifiers	Accuracy before HPT (Hyper-Parameter Tuning) (%)	Accuracy after Hyper-Parameter Tuning (%)
1	KNN	74%	77.70%
2	Decision Tree	61%	77.26%
3	Random Forest	63%	78.13%
4	AdaBoost	75%	78.02%
5	Bagging	66%	71.71%

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [4, 5, 6, 7, 8],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [200, 500]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
CV_rfc.best_estimator_
```

```
RandomForestClassifier(max_depth=4, max_features='auto', n_estimators=200,
                       random_state=42)
```

Fig.2 Best Parameter of Random Forest Classifiers

Table IV displays the accuracy performance of the various classifiers both before and after HPT.The classifier KNN, DT, RF, AdaBoost, and Bagging before HPT showed accuracy of 74%, 61%, 63%, 75% and 66% respectively whereas after HPT the accuracy of classifiers are intensely changed as 77.70%, 77.26%, 78.13%, 78.02% and 71.71%. The RF classifier with HPT has accuracy of 78.13%, performing the best among the other classifiers. The hyper parameter settings are shown in the Fig.2. To overcome the issues of overfitting, cross validation (CV) a hyperparameter optimization method is used whose value is selected as 5. Different iterations of 5- fold CV has been performed for achieving the hyperparameter tuning and in each iteration different model setting is also used. Then from the different model, the best model is selected. The second criteria Gini index is used for pure splitting of the decision nodes. The justification of using Gini index is, it is faster and computationally less expensive [23-24]. The parameter estimator represents the number of trees, we have set the value as 200 because more the number of trees more will be the performance and the model will be stronger.

A. Comparison between with or without weight factor HPT

In this section, we compared the “Car leasing “and “MIS” data sets. The precision is greatly enhanced by the addition of a weight component. On these two datasets, we applied the five classifiers KNN, RF, DT, Bagging and Boosting with HPT. As can be seen in Table III, the results showed that accuracy in the car-lease dataset without weight factor is 36%. Similar to this, we found that accuracy increases to 75% when the weight component is included. This leads us to the conclusion

that selecting the appropriate characteristic may transform accuracy from being low to being high.

TABLE III. COMPARISON BETWEEN TWO DATASETS WITH OR WITHOUT WEIGHT FACTOR USING HYPER-PARAMETER TUNING

Data Sets	Classifiers	Accuracy with Hyper-Parameter Tuning (%)
Without weight Factor	KNN	36%
	RF	35%
	DT	34%
	Bagging	33%
	Boosting	36%
With Weight Factor	KNN	77.70%
	RF	78.13%
	DT	77.26%
	Bagging	71.71%
	Boosting	78.02%

B. Comparison of proposed model with existing technique

The accuracy is presented to further demonstrate the effectiveness of TCP using Hyper-Parameter with existing technique as shown in Table IV and Fig. 3

The proposed technique and its accuracy are evaluated and compared with other existing unsupervised (clustering) techniques as K-means, NDBC-FFNN and AHC. The accuracy of applying K-means and NDBC-FFNN is 66%, that of using AHC technique accuracy is 39%. It is unquestionable that our proposed model having accuracy 78.13% is outperformed among the existing classifiers.

TABLE IV. COMPARISION OF PROPOSED MODEL WITH EXISTING TECHNIQUE

Sl.no	Techniques used	Accuracy (%)
1	K-Mean	66%
2	(NDBC-FFNN) Novel DBSCAN Clustering based Feed Forward Neural Network	66%
3	(AHC) Agglomerative Hierarchical clustering	39%
4	Proposed Technique	78.13%

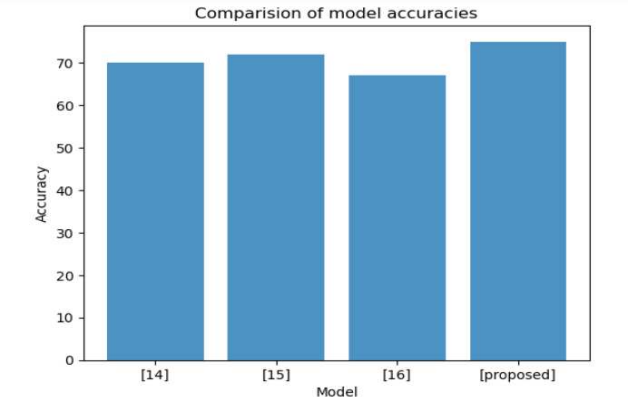

Fig.3 Comparison of proposed model with existing approach

VII. CONCLUSION AND FUTURE WORK

During our careful analysis we were confronted with various Machine Learning methods used for ranking the test cases. We discovered that always source code is not available with respect to rank the most important test cases. Therefore, a new method known as requirement-dependent test case prioritization is presented. This approach prioritized the test cases dependent on requirement factors. In this study, we have adopted a multi-class classification method of machine learning technique with HPT for prioritizing test cases based on requirements. The suggested method was verified with the 'MIS with Priority' dataset. To assess the performance in terms of accuracy, we have employed four machine learning classifiers including bagging, RF, DT, KNN and Boosting. RF estimators outperformed all having 78.13% accuracy. In future, we plan to consider a dataset having a greater number of features for achieving the best performance of the ML model and use more than two datasets for optimizing the model to get prioritized test cases for early fault detection.

REFERENCES

[1] S. Mohanty, A. A. Acharya and D. P. Mohapatra, "A model-based prioritization technique for component based using UML state chart diagram," 3rd Int. Conf. Ele. Comp. Techno, IEEE, 2011.

[2] S. Mohanty, A. A. Acharya and D. P Mohapatra, "A survey on model-basedtest case prioritization," Int. Jour. Comp. Sci. Info. Techno, vol. 3, pp. 1042-1047, 2011.

[3] R. Pan, M. Bagherzadeh, T. A Ghaleb and L. Briand, "Test case selection and prioritization using machine learning: a systematic literature review," Emp. Soft. Eng, vol. 2, p. 29, 2022.

[4] A. Sharif, D. Marijan and M. Liaaen, "Deep order: deep learning for test case prioritization in continuous integration testing," Intl. Conf. Soft. Main. Evo, pp. 525-534. IEEE, 2021.

[5] H. Spieker, A. Gotlieb, D. Marijan and M. Mossige, "Reinforcement learning for automatic test case prioritization and selection in continuous integration." Procs. SIG. Intl. Symp. Soft. Test, pp. 12-22, ACM, 2017.

[6] R. Lachmann, S. Schulze, M. Nieke, C. Seidl and I. Schaefer, "System-level test case prioritization using machine learning," Intl. Conf. Mach. Lear. Apps, pp. 361-368. IEEE, 2016.

[7] R. Lachmann, "Machine learning-driven test case prioritization approaches for black-box software testing," Euro. Eley. Conf. Germany, 2018.

[8] M. Arafeen and H. Do, "Test case prioritization using requirements-basedclustering," intl. conf. soft. Val, pp. 312-321, IEEE, 2013.

[9] H. Srikanth, C. Hettiarachchi and H. Do, "Requirements based test prioritization using risk factors: An industrial study," Info. Soft. Techno, vol. 69, pp 71-83, ELSEVIER, 2016.

[10] M. Luqman, W. M. N. Kadir, M. Khatibsyarbini and M. A. Isa, "Model-based test case prioritization using selective and even-spread count-based methods with scrutinized ordering criterion," vol. 2, p. e0229312, Plos one, 2020.

[11] R. Mukherjee and K. Patnaik, "A survey on different approaches for software test case prioritization, "Jour. Saud. Uni. Comp. Info. Sci. vol. 33, pp. 1041-1054, ELSEVIER, 2021.

[12] L. Zahedi, F. G. Mohammadi, S. Rezapour, M. W. Ohland and M. H. Amini, "Search algorithms for automated hyper-parameter tuning," vol. 14677, arXiv preprint, 2021.

[13] S. Chaudhary and A. Jatain, "Performance evaluation of clustering techniques in test case prioritization," Intl. Conf.

Comnl. Perfo. Eva, pp. 699-703,IEEE, 2020.

[14] A. Kumar and A. K Misra, "Clustering Based Prioritization of Test Suites in Software Testing," Intl. Jour. Sci. Eng, vol. 8, pp. 1-5, 2012.

[15] Y. Shin, M. Harman, P. Tonella and A. Susi, "Clustering test cases to achieve effective and scalable prioritization incorporating expert knowledge," Proc. Intl. sym. Soft. analysis, pp. 201-212. 2009.

[16] https://www.kaggle.com/datasets/charvijain27/training-data-2-csv-utfcsv.

[17] Pranjal, P., Mallick, S., Madan, M., Mishra, S., Alkhayyat, A., & Bhaktisudha, S. (2023, February). A Smart Data-Driven Prototype for Depression and Stress Tracking in Patients. In International Conference On Innovative Computing And Communication (pp. 423-434). Singapore: Springer Nature Singapore.

[18] Verma, S., Sinha, S., Chaudhury, P., Mishra, S., Alkhayyat, A. (2023). Crop Yield Forecasting with Precise Machine Learning. In: Hassanien, A.E., Castillo, O., Anand, S., Jaiswal, A. (eds) International Conference on Innovative Computing and Communications. ICICC 2023. Lecture Notes in Networks and Systems, vol 537. Springer, Singapore. https://doi.org/10.1007/978-981-99-3010-4_38

[19] Dutta, S., Choudhury, S., Chakraborty, A., Mishra, S., & Chaudhary, V. (2023, February). Parkinson Risks Determination Using SVM Coupled Stacking. In International Conference On Innovative Computing And Communication (pp. 283-291). Singapore: Springer Nature Singapore.

[20] Patra, P., Ved, V., Chakraborty, S., Mishra, S., & Chaudhary, V. (2023, February). ECG-Based Cardiac Abnormalities Analysis Using Adaptive Artificial Neural Network. In International Conference On Innovative Computing And Communication (pp. 245-251). Singapore: Springer Nature Singapore.

[21] Mishra, N., Mishra, S., & Tripathy, H. K. (2022, December). Rice Yield Estimation Using Deep Learning. In International Conference on Innovations in Intelligent Computing and Communications (pp. 379-388). Cham: Springer International Publishing.

[22] Mishra, S., Mishra, B. K., & Tripathy, H. K. (2015, December). A neuro-genetic model to predict hepatitis disease risk. In 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC) (pp. 1-3). IEEE.

[23] Abhishek, Tripathy, H. K., & Mishra, S. (2022). A Succinct Analytical Study of the Usability of Encryption Methods in Healthcare Data Security. In Next Generation Healthcare Informatics (pp. 105-120). Singapore: Springer Nature Singapore

[24] Suman, S., Mishra, S., & Tripathy, H. K. (2021). A Support Vector Machine Approach for Effective Bicycle Sharing in Urban Zones. In Cognitive Informatics and Soft Computing: Proceeding of CISC 2020 (pp. 73-83). Springer Singapore.