Pertanika J. Sci. & Technol. 34 (1): 283 - 315 (2026)

PERTANIKA JOURNALS

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Evaluating Hyperparameter Tuning of Random Forest and CatBoost on Complex and Imbalanced Real-world Datasets

Andri[1,4], Darwin[1,4], Ng Poi Wong[1,4], Sutarman[2]*, and Erna Budhiarti Nababan[3]

[1]Graduate Programme of Computer Science, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Jalan Universitas Road no. 9, 20155 Medan, North Sumatra, Indonesia
[2]Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Sumatera Utara, Jalan Bioteknologi no. 1, 20155 Medan, North Sumatra, Indonesia
[3]Master's Study Programme in Data Science and Artificial Intelligence, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Jalan Universitas Road no. 9, 20155 Medan, North Sumatra, Indonesia
[4]Undergraduate Programme of Computer Science, Faculty of Informatics, Universitas Mikroskil, Jalan Thamrin no. 112, 124, 140, 20212 Medan, North Sumatra, Indonesia

ABSTRACT

Research on hyperparameter tuning of Random Forest (RF) and CatBoost on imbalanced datasets often focusses on these algorithms separately, with limited evaluation of both comprehensively across a wide range of dataset complexities. The effects of hyperparameter tuning in handling pattern complexity and class distribution in real-world imbalanced datasets remain unexplored. This gap hinders the understanding of how hyperparameter optimisation can improve performance for such data, leading to potential model unoptimality and challenges in selecting the most effective algorithm. This research evaluates the impact of RF and CatBoost hyperparameter tuning on complex and imbalanced real-world datasets, with XGBoost added as a comparative baseline. The datasets used include binary and multi-class categories with varying degrees of class imbalance and feature complexity, as measured using Shannon Entropy and Coefficient of Variation (CV). Hyperparameter tuning uses Bayesian Optimisation (BO-TPE), Hyperband (HB), and Random Search (RS). Results show that datasets with high CV result in significant differences between accuracy and F1-score values. Hyperparameter tuning on RF improved the average accuracy and F1 score on binary-class datasets, but did not have a significant impact on AUC. In contrast, tuning on RF for multi-class datasets provided more consistent improvements across all three evaluation metrics. On the other hand, CatBoost and XGBoost tuning provided consistent average improvements on all three

ARTICLE INFO

Article history:
Received: 20 December 2024
Accepted: 25 August 2025
Published: 06 February 2026

DOI: https://doi.org/10.47836/pjst.34.1.13

E-mail addresses:
andri@students.usu.ac.id, andri@mikroskil.ac.id (Andri)
darwin@students.usu.ac.id, darwin@mikroskil.ac.id (Darwin)
ngpoi@students.usu.ac.id, poiwong@mikroskil.ac.id (Ng Poi Wong)
sutarman@usu.ac.id (Sutarman)
ernabrn@usu.ac.id (Erna Budhiarti Nababan)
* Corresponding author

e-ISSN: 2231-8526

metrics for both binary and multi-class datasets. CatBoost generally shows the best efficiency on large datasets, followed by XGBoost and RF. In contrast, on small datasets, XGBoost is the most efficient, followed by RF and CatBoost.

Keywords: CatBoost, complex dataset, hyperparameter tuning, imbalanced dataset, random forest

INTRODUCTION

Imbalanced data distribution is one of the challenges in various classification problems and requires effective handling by learning from the imbalanced data (Patel & Bhavsar, 2024). Machine learning has become an important pillar in solving various real-world problems in the modern era, including analysing imbalanced data. Imbalanced datasets can be found in various fields, such as fraud detection (Ayyannan, 2024), medical diagnosis (Huang et al., 2024; Mustapha & Ozsahin, 2024; Y. Yang et al., 2024), credit risk (Matar et al., 2024), and education (Andri et al., 2023). In imbalanced datasets, there are conditions where the amount of data from the minority class is much less than the majority class, which can cause machine learning algorithms to focuss on learning and predicting the majority class more accurately than the minority class (Gu et al., 2022; Hazarika et al., 2023; Hazarika & Gupta, 2022, 2023).

In the context of handling imbalanced data, Random Forest and CatBoost were chosen because they have been proven effective and widely used in various real case studies, such as anaemia detection (Praptiwi et al., 2024), lung cancer classification (Zamzam et al., 2024), customer churn prediction (Imani & Arabnia, 2023), and intrusion detection systems (Alshamy et al., 2021). Their effectiveness in handling imbalanced class distributions has been reported in many studies, making them representative candidates in classification performance evaluation studies on real data. Random Forest is a popular ensemble technique that builds each tree independently, reducing bias and improving overall model performance (Praptiwi et al., 2024), while CatBoost has advantages in handling categorical features and training efficiency on large data (Hancock & Khoshgoftaar, 2020). However, the performance of both algorithms can be improved through optimal hyperparameter tuning (L. Yang & Shami, 2020). Hyperparameter tuning plays an important role in improving the accuracy and efficiency of the model by systematically exploring the hyperparameter space, which can significantly affect the predictive performance of machine learning models (Abdellatif et al., 2022; L. Yang & Shami, 2020; Mendes et al., 2023; Suryadi et al., 2024; Wen et al., 2022).

Although there are researches that examine the effects of hyperparameter tuning on RF or CatBoost separately in imbalanced datasets, there is limited research that combines the evaluation of both comprehensively on different complexity levels of imbalanced datasets. In addition, there is a lack of research on the effect of hyperparameter tuning on the performance of both algorithms in handling pattern complexity and class distribution

on real-world imbalanced datasets. This creates a gap in the understanding of how hyperparameter optimisation in RF and CatBoost can contribute to improved performance on data that has high-class imbalance and complexity. The lack of understanding of the effect of hyperparameter tuning on the performance of RF and CatBoost on complex imbalanced datasets, as well as the limitations of previous studies that rarely compare the performance difference of these algorithms after hyperparameter tuning, has led to potential model non-optimality in handling this type of data and hindered the selection of the most effective algorithm. This challenge becomes even more relevant given the importance of data-driven decision-making in real-world scenarios involving datasets with class imbalance and complex distribution patterns (Chen et al., 2024).

This research aims to evaluate the impact of hyperparameter tuning on the performance of RF and CatBoost on imbalanced datasets with varying degrees of complexity. This research is expected to make a significant contribution to understanding and overcoming the challenges of working with imbalanced datasets through an exploration of how hyperparameter optimisation can improve the capabilities of these algorithms, and is expected to serve as a reference for the development of more effective machine learning models in real-world applications.

LITERATURE REVIEW

The following are some literature reviews of research relevant to this research:

1. Barella et al. (2021) conducted two experiments to evaluate data complexity in imbalanced datasets. The first adapted complexity measures to assess class difficulty using 102 real-world datasets with stratified 5-Fold CV, showing improved evaluation, particularly for minority classes. The second experiment tested 101 datasets with imbalance treatment techniques (SMOTE, ADASYN, and Random Oversampling), revealing that reducing imbalance lowers data complexity and improves prediction performance, with a strong correlation between complexity reduction and model accuracy.
2. Borah and Gupta (2021) developed Robust Twin Bounded Support Vector Machine (RTBSVM) to handle the problem of binary class imbalance and outliers that often interfere with the performance of machine learning models. The test results show that RTBSVM is able to handle outliers and class imbalance simultaneously, where on the Yeast-0-5-6-7-9_vs_4 dataset, RTBSVM achieved an Area under the ROC curve (AUC) of 0.9117 with a computation time of 0.01926 seconds, and on the Ecoli3 dataset, RTBSVM achieved a G-mean of 0.7954 with a computation time of 0.0217 seconds.
3. Gu et al. (2022) optimised RF in the classification of imbalanced data by introducing an equilibrium ensemble method that is a combination of an over-sampling method, Data Centre Interpolation (DCI) and a hyperparameter optimisation algorithm of RF,

Improved Sparrow Search Algorithm (ISSA). Test results show that the F1-score and G-mean of DCI-ISSA outperform other methods such as SVM, Naive Bayes, and various ensemble variations of RF.

4. Wen et al. (2022) introduced a multi-objective optimisation method for linear motor design by optimising the hyperparameters of the RF with Bayesian Optimisation (BO) and Hyperband (HB). The multi-objective design optimisation uses Non-Dominated Sorting Genetic Algorithm II (NSGA-II). The test results show that RF optimised with Bayesian Optimisation and Hyperband (BOHB) has better performance than BO and HB, in which the R^2 of BOHB is higher and the RMSE is low.
5. Abdellatif et al. (2022) developed the SMOTE-Extra Trees (ET) model with hyperparameter optimisation using Hyperband (HB). The class imbalance problem was addressed using SMOTE. The model was applied to detect the presence of heart disease, and classify the severity of heart disease. The test results in detecting the presence of heart disease showed that the SMOTE-ET-HB model gave an accuracy of 99.2%, while in classifying the severity of heart disease, the accuracy reached 95.73%.
6. Ren et al. (2023) developed an Adaptive Laplacian Weight Random Forest (ALWRF) to improve the prediction of mixed-type and imbalanced data. Imbalanced data is handled using SMOTE-NC, and data imputation of missing values uses a combination of ALWRF with SMOTE-NC in the SncALWRFI method. Hyperparameter optimisation uses Bayesian Optimisation and Cross Validation. The test results show ALWRF has higher classification and regression accuracy than regular Random Forest and RF with Bayesian Optimisation, SncALWRFI provides the best imputation accuracy on datasets that have missing values with imbalanced classes, and SMOTE-NC improves the distribution of minority data without causing overfitting.
7. Albahli (2023) focusses on the use of RF and Decision Trees for the classification of student academic performance prediction. Hyperparameter tuning uses Bayesian Optimisation, Grid Search, and Random Search, and imbalanced datasets are handled using SMOTE. The test results show RF with Bayesian Optimisation and SMOTE gives an accuracy of 0.9 and F1-score of 0.89, and the Decision Tree with Bayesian Optimisation without SMOTE gives an accuracy of 0.95 and F1-score of 0.89. In terms of comparison of hyperparameter tuning methods, Bayesian Optimisation is faster and more accurate than Grid Search and Random Search, and SMOTE can improve model performance on imbalanced data.
8. Imani and Arabnia (2023) compared various machine learning algorithms: ANN, Decision Tree, RF, Logistic Regression, SVM, and Gradient Boosting (XGBoost, LightGBM, and CatBoost) in predicting customer churn in the telecommunications industry. Imbalanced datasets are handled using SMOTE, SMOTE+Tomek Links,

SMOTE-ENN, and hyperparameter optimisation using Optuna. The test results show that CatBoost with Optuna gives the best performance with an F1-score of 93%, and CatBoost with SMOTE-ENN gives an AUC of 91%.

9. Yuliana et al. (2024) focussed on hyperparameter optimisation of RF using Grid Search in predicting 5G coverage. The test results show an increase in the accuracy level of hyperparameter tuning, which has decreased in the RMSE value from 1.14 to 0.66, MSE from 26.23 to 0.43, and MAE from 0.12 to 0.05, as well as an increase in R^2 from 0.979 to 0.9931.
10. Suryadi et al. (2024) focussed on improving the performance of RF classification by tuning its hyperparameters in predicting software defects. Hyperparameter tuning uses Grid Search, Random Search, Optuna, Bayesian (Hyperopt), Hyperband, TPE, and Nevergrad techniques. Imbalanced data is handled using SMOTE, and feature selection is used to improve classification efficiency using a Genetic Algorithm. The test results show that the Nevergrad method provides the highest accuracy with an average of 0.039, and the Hyperband method provides the highest AUC with an average of 0.0351.
11. Praptiwi et al. (2024) overcame the imbalanced class problem by using RF and CatBoost ensemble models in detecting anaemia risk factors in children aged between 5-12 years. The methods used to solve the imbalanced class problem consist of Oversampling methods: Random Over Sampling (ROS), SMOTE, and G-SMOTE, Undersampling methods: Random Under Sampling (RUS), Instance Hardness Threshold (IHT), and hybrid methods: SMOTE-ENN. Hyperparameter optimisation uses Bayesian Search with 20 iterations. The test results show CatBoost with G-SMOTE outperforms other methods, which obtained a sensitivity of 0.7104, specificity of 0.7043, G-mean of 0.7067, and AUC of 0.7844, G-SMOTE improves sensitivity in RF and CatBoost ensemble models, and SMOTE-ENN improves G-mean for RF models.
12. Zamzam et al. (2024) compared the performance of CatBoost and RF methods with hyperparameter tuning using Bayesian Optimisation to classify lung cancer. The test results show RF with Bayesian Optimisation provides the best performance with an accuracy value of 97.1% and an AUC of 99.86%.
13. Zheng et al. (2024) developed an ensemble model based on XGBoost, LightGBM, and Neural Networks as a meta-classifier, using SMOTE to address imbalanced datasets in payment transactions. The model is compared with other individual models: Logistic Regression, CatBoost, and TabNet. The test results without using SMOTE show that the ensemble model provides a performance of Precision 0.97, AUC 0.97, Recall 0.86, and F1-score 0.91, while the test results using SMOTE show that the ensemble model provides a performance of Precision 0.98, AUC 0.99, Recall 0.9, and F1-score 0.94.

14. Manda et al. (2024) developed a model to detect fraud in credit card transactions with RF, AdaBoost, CatBoost, XGBoost, and LightGBM. Imbalanced data was handled using SMOTE, and hyperparameter optimisation using Grid Search and Cross-Validation. The test results show that XGBoost gives the best performance with AUC 0.974, LightGBM with AUC 0.946, and CatBoost with AUC 0.86.

METHODOLOGY

The research design that shows the detailed process flow in this study can be seen in Figure 1 and modular architecture can be seen in Figure 2, including inputting the dataset and performing data preprocessing which includes: (1) checking for empty values to ensure

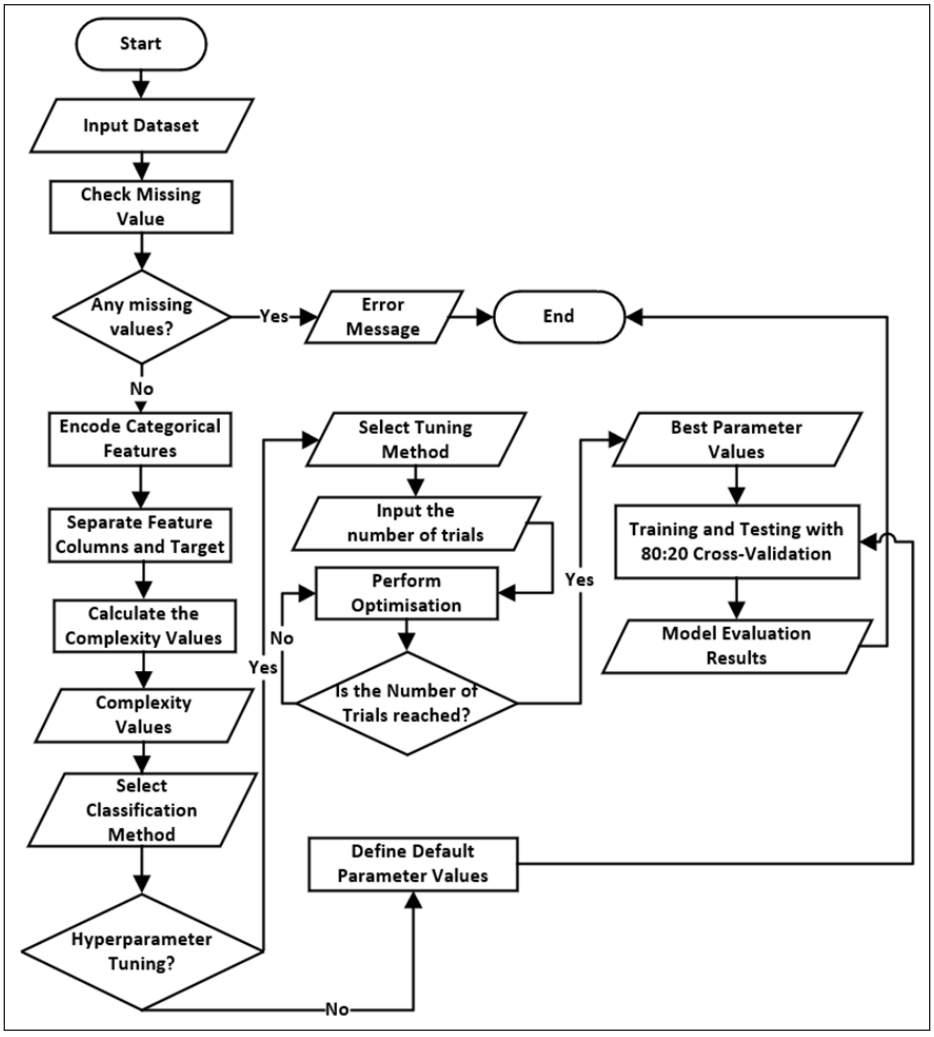

Figure 1. Detailed process flow of research design

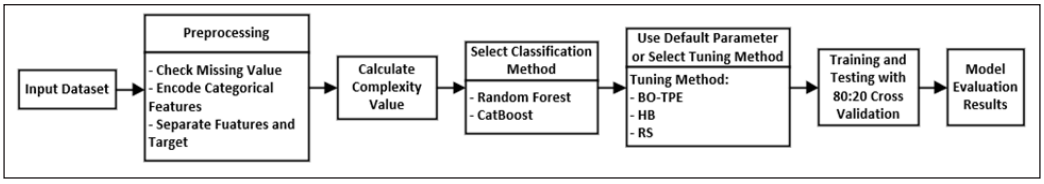

Figure 2. Modular architecture of research design

there is no missing data in either features or targets, (2) encoding of categorical features to convert categorical values to numerical in order to be processed by machine learning algorithms, (3) separation between feature and target columns. After that, calculate the complexity value of the dataset, and select the classification method. If no hyperparameter tuning is performed, then specify the default parameters, otherwise if hyperparameter tuning is performed, then specify the number of trials (this study uses a number of trials of 100), then perform hyperparameter optimisation until all trials are complete, and proceed to the process of training and testing the model using cross-validation with a data division of 80:20. The results of the evaluation model are then analysed based on the default parameters and the best tuning parameters. For comparison, the Random Forest and CatBoost tuning results were also compared with XGBoost.

Dataset

The datasets used are datasets taken from several open datasets database sources, such as UCI Machine Learning Repository, Kaggle, and OpenML. These datasets are real-world data that have an uneven distribution of classes and have no missing values. Selection of binary class datasets by considering the percentage of majority classes that have a proportion of at least 60% of the total dataset samples. Meanwhile, data selection on multi-class datasets is done by considering the majority class with a minimum number of n, where n is expressed as 100% / number of classes + 10% of the total number of dataset samples. Through this approach, the selected dataset will reflect the imbalance of data in the real world, but still maintain an adequate representation of the majority class. The size of the dataset ranges from 351 to 100,000, with the number of features ranging from 22 to 38. The level of complexity of the dataset was also measured using Shannon entropy and coefficient of variation metrics. The basic information and imbalanced class of the datasets used can be seen in Table 1, with a brief description as follows:

1. Ionosphere (binary-class) is a dataset collected from a radar signalling system at Goose Bay, Labrador that classifies radar signals based on ionospheric features, which has a more moderate imbalanced class and relatively simple features.
2. KC2 (binary-class) is a dataset sourced from the NASA Metrics Data Programmeme (MDP), used to predict the presence of defects in software modules used to process scientific data, which is a simpler dataset with two imbalanced classes.

3. PC4 (binary-class) is dataset sourced from NASA MDP which is the output of software running on artificial satellites orbiting the earth, which has lower imbalanced classes compared to other datasets.
4. Forest Type Mapping (multi-class) is a four-class dataset used to map forest types using satellite imagery data, which is relatively balanced, although there are challenges in feature complexity.
5. Predict Students' Dropout and Academic Success (multi-class) is a dataset that predicts the dropout status of university students, which has a significant class imbalance with far fewer students dropping out than not dropping out.
6. Credit Score Classification (multi-class) is a dataset on credit scoring to predict the default risk of customers, which has a significant class imbalance with the majority of data coming from the low-risk class.

Table 1
Basic information and imbalanced class of the datasets

Dataset	Domain (Source)	Class Category	
		Target	Value
Ionosphere n=351 features=34	Remote Sensing (https://archive.ics.uci.edu/dataset/52/ionosphere)	"target"	"g" (Good) n=225 ratio=1.79 percentage=64.1%
			"b" (Bad) n=126 ratio=1 percentage=35.9%
KC2 n=522 features=22	Software Engineering (https://openml.org/search?type=data&status=active&sort=runs&id=1063)	"problems"	"yes" n=107 ratio=1 percentage=20.5%
			"no" n=415 ratio=3.88 percentage=79.5%
PC4 n=1458 features=38	Software Engineering (https://openml.org/search?type=data&status=active&sort=runs&id=1049)	"c"	"TRUE" n=178 ratio=1 percentage=12.2%
			"FALSE" n=1280 ratio=7.19 percentage=87.8%

Table 1 *(continued)*

Dataset	Domain (Source)	Class Category	
		Target	Value
Forest Type Mapping n=523 features=27	Remote Sensing (https://archive.ics.uci.edu/dataset/333/forest+type+mapping)	"class"	"s" (Sugi) n=195 ratio=2.35 percentage=37.3%
			"h" (Hinoki) n=86 ratio=1.04 percentage=16.4%
			"d" (Mixed Deciduous) n=159 ratio=1.92 percentage=30.4%
			"o" (Other) n=83 ratio=1 percentage=15.9%
Predict Students' Dropout and Academic Success n=4424 features=36	Education (https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success)	"Target"	"Enrolled" n=794 ratio=1 percentage=18%
			"Graduate" n=2209 ratio=2.78 percentage=49.9%
			"Dropout" n=1421 ratio=1.79 percentage=32.1%
Credit Score Classification n=100000 features=28	Finance (https://www.kaggle.com/datasets/iremnurtokuroglu/credit-score-classification-cleaned-dataset/data)	"credit_score"	"0" n=28998 ratio=1.63 percentage=29%
			"1" n=53174 ratio=2.98 percentage=53.2%
			"2" n=17828 ratio=1 percentage=17.8%

Note. n = number of instances, features = number of features, source = open dataset database source name, ratio = ratio of the number of target value instances to the smallest number of target value instances, percentage = the percentage of the number of target value instances to the total number of instances

Random Forest

The tree-based model is the basis of the Random Forest (RF) algorithm. It recursively splits a dataset based on certain criteria into two parts until it meets a predefined stopping condition. The result of the division is the leaf nodes at the bottom of the decision tree. The recursive division of the two-dimensional input space with axis-aligned constraints can be seen as in Figure 3, where the first split occurs at $x_2 \geq a_2$, followed by next subspace splits at $x_1 \geq a_4$ on the left branch. On the right branch, it split at $x_1 \geq a_1$ and then splits again at $x_2 \geq a_3$ on one of its subbranches. The representation of the partitioned subspace in the decision tree can be seen in Figure 4 (Schonlau & Zou, 2020).

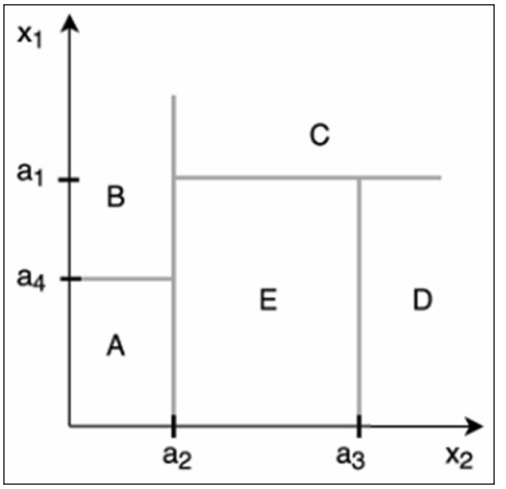

Figure 3. Recursive division of two-dimensional input space (Schonlau & Zou, 2020)

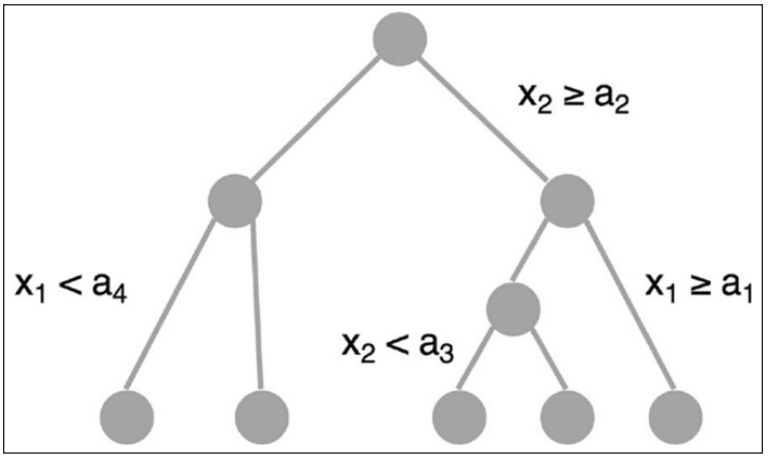

Figure 4. Representation of a decision tree (Schonlau & Zou, 2020)

```
for I ß 1 to B do
      Draw a bootstrap sample of size N from the training data;
      while node size != minimum node size do
            randomly select a subset of m predictor variables from total p;
            for j ß 1 to m do
                  if jth predictor optimizes splitting criterion then
                        split internal node into two child nodes;
                        break;
                  end
            end
      end
end
return the ensemble tree of all B subtrees generated in the outer for loop;
```

Figure 5. The Random Forest algorithm (Schonlau & Zou, 2020)

Decision trees can be used in classification tasks with categorical outputs or regression with continuous outputs. The output of the decision tree depends on the division criteria and the specified stopping conditions. In classification, a commonly used criterion is entropy, which is used to maximise the information obtained at each division, while in regression, a frequently used criterion is the mean squared error (Schonlau & Zou, 2020).

Decision trees are prone to overfitting, which is when the model is trained and predicted using a training dataset that is too close and thus performs poorly against new data. Bootstrap aggregating (bagging) methods can be used to improve generalisability, where many individual decision trees are built using bootstrap samples from the original dataset. RF is an ensemble-based algorithm that combines predictions from many decision trees to reduce overfitting and improve accuracy. The RF algorithm is illustrated in Figure 5.

CatBoost

CatBoost is an open-source implementation of Gradient Boosted Decision Tree (GBDT) for supervised machine learning with two key innovations: Ordered Target Statistics and Ordered Boosting. GBDT is highly recommended for heterogeneous datasets, which have features with different data types, such as those found in web search engines, recommendation systems, and weather forecasts. As for homogeneous data consisting of features with similar data types, it is better to use other algorithms such as neural networks. Research shows that CatBoost is more suitable for heterogeneous data, but not the best choice for homogeneous data. Therefore, the homogeneous or heterogeneous nature of the data should be one of the considerations in the implementation of machine learning (Hancock & Khoshgoftaar, 2020).

Research comparing the accuracy and training time of the GBDT algorithms: CatBoost, XGBoost, and LightGBM against benchmarking tasks with large datasets, shows that CatBoost is sensitive to hyperparameter settings which can affect its training time and performance compared to other algorithms. CatBoost's training time varies greatly

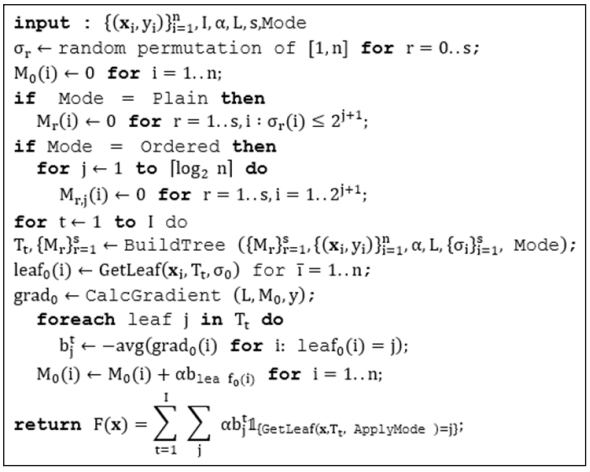

Figure 6. The CatBoost algorithm (Prokhorenkova et al., 2017)

depending on the hyperparameter configuration, with some configurations requiring training time between 10 to 100 minutes, while with other algorithms, it almost reaches 1000 minutes. In addition, XGBoost could not be run on the Epsilon benchmark as it consumed more memory than CatBoost for the same task. This shows that it is important to adjust the hyperparameters to get the best training results on CatBoost (Hancock & Khoshgoftaar, 2020).

CatBoost is a boosting implementation that outperforms other boosting methods in terms of performance. Its advantage lies in the innovative idea of the algorithm in improving model performance, where one of the advantages is that it is able to handle categorical features. The CatBoost algorithm is illustrated in Figure 6.

Hyperparameter Tuning

Hyperparameters are parameters that will affect the training process and the structure of the machine learning model. Hyperparameters are determined before training begins and require careful selection. Hyperparameters play an important role in model performance. Examples of hyperparameters include learning rate, number of trees, depth of decision tree, penalty term, momentum, learning rate decay, and regularisation constant (Ilemobayo et al., 2024).

Hyperparameter tuning is to find the optimal hyperparameter combination that provides the best performance, reduced bias and variance, and resource efficiency in machine learning models. Hyperparameters control the learning process and model structure, thus greatly affecting performance (Ilemobayo et al., 2024).

Various techniques have been developed to automate and optimise the hyperparameter tuning process, as follows:

a. Bayesian Optimisation is a probabilistic approach to global optimisation that is often used in machine learning and optimisation. The technique aims to find the optimal combination of hyperparameters to maximise a specific objective function, such as accuracy in machine learning and deep learning models, or efficiency in manufacturing processes. Bayesian Optimisation can use various models as surrogate functions, such as Gaussian Process (GP), Random Forest (RF), and Tree-structured Parzen Estimators (TPE). These surrogate functions simulate the distribution of the objective function to approximate the unknown objective function. This process is performed iteratively by updating the surrogate model using new observations of the objective function, so that the prediction of the optimal hyperparameter combination becomes increasingly accurate (Vo et al., 2023).

b. Hyperband is an adaptive strategy for resource allocation and early stopping in hyperparameter optimisation. This technique evaluates hyperparameter configurations and assigns more resources to promising configurations, balancing exploration and exploitation (Ilemobayo et al., 2024).

c. Random Search is a simple and widely used hyperparameter optimisation technique in machine learning. It randomly samples hyperparameters from a predefined search space, and then evaluates model performance using a validation set for each hyperparameter combination. One of its advantages is its simplicity, which allows implementation with little need for customisation. In addition, this technique is computationally efficient as it can be parallelised by optimally utilising resources. Random Search also often provides good performance on low-dimensional hyperparameter spaces, outperforming more complex optimisation techniques (Vo et al., 2023).

Complexity Measurement

Complexity is a phenomenon that has many dimensions, involving various aspects such as disorder, non-linearity, and the ability to self-organise. Since scientists began studying complex systems, many measures of complexity have been proposed, and the number continues to grow. Complexity has a variety of characteristics, but not all of these characteristics are found in every complex system (Wiesner & Ladyman, 2019).

Complexity features are divided into two categories: complexity conditions and complexity conditions. Complexity conditions include number of elements, number of interactions, disorders, imbalance, and feedback. Complexity products include nonlinearity, robustness of order, robustness of function, self-organisation, nestedness, adaptive behaviour, modularity, and memory. Not all products appear in all complex systems, especially those found only in functional or live systems, such as adaptive behaviour, robustness of function, modularity, and memory (Wiesner & Ladyman, 2019).

Shannon Entropy is one of the tools to measure complexity in the form of the level of uncertainty in a probability distribution P. Under conditions where all probabilities are equal, the distribution is called a uniform distribution. Shannon Entropy measures species diversity by considering the frequency of each species in a habitat. Shannon Entropy reaches a maximum value when all species have the same probability, indicating a uniform distribution and high uncertainty, using Equation 1 as follows (Wiesner & Ladyman, 2019).

$$H(X) := -\sum_{x \in X} P(x) \log P(x) \quad [1]$$

where x is an element of the set of values X and P(x) is the probability of event X.

Coefficient of Variation (CV) is also another metric to measure complexity, which is the square root of the variance divided by the mean, as shown in Equation 2. CV compares distributions with the same variance but with different means (Wiesner & Ladyman, 2019). For example, a distribution with a variance of 15 and a mean of 25 is considered more diverse than a distribution with a variance of 15 and a mean of 1500. CV reflects this difference and allows a fairer comparison between different distributions.

$$\text{cv} := \frac{\sqrt{\text{Var}X}}{\langle X \rangle} \quad [2]$$

where VarX is the variance value of X and (X) is the average or mean of the variable X.

Experimental Setup

Hyperparameter tuning is performed to optimise several important parameters that affect the performance of the RF and CatBoost. The optimised hyperparameter tuning on RF can be seen in Table 2, while the optimised hyperparameter tuning on CatBoost can be seen in Table 3. The search space for RF hyperparameter tuning includes: n_estimators (50 to 200 in increments of 50), max_depth (5 to 50), min_samples_split (2 to 10), min_samples_leaf (1 to 10), and max_features (options: "sqrt", "log2", None). A wider search space is avoided to maintain computational complexity. The search space for CatBoost hyperparameter tuning includes: iterations (set at 100 to control computation time), learning_rate (0.01 to 0.1), depth (3 to 10), l2_leaf_reg (0.001 to 10), and border_count (32 to 256). All experiments were run using the Jupyter Notebook environment on a server with the following specifications: 20 physical cores (28 logical cores) with a CPU frequency of 2.04 GHz, 62.63 GB of RAM, and one active GPU unit. These specifications were chosen to ensure the stability and efficiency of the tuning process, especially when applying optimisation algorithms such as BO-TPE, Hyperband, and Random Search that require considerable computing resources.

Table 2
Hyperparameter tuning on RF

Parameter	Symbol	Description	Search Space
n_estimators	nDT	The number of trees. The default is 100.	[50, 100, 150, 200]
max_depth	maxD	The maximum depth of the tree. The default is None.	5 to 50
min_samples_split	minS	The minimum number of samples required to split an internal node. The default is 2.	2 to 10
min_samples_leaf	minL	The minimum number of samples required to be classified as a leaf. The default is 1.	1 to 10
max_feature	nAtt	The number of attributes to consider when looking for the best split. The default is to use 'sqrt'.	'sqrt', 'log2', None

Note. nDT = tree depth, maxD = maximum depth, minS = minimum samples to split a node, minL = minimum samples per leaf, nAtt = number of attributes

Table 3
Hyperparameter tuning on CatBoost

Parameter	Symbol	Description	Search Space
iterations	nTree	The maximum number of trees. The default is 100.	100 (fixed)
learning_rate	lr	The learning rate. The default is 0.03.	0.01 to 0.1
depth	nDT	Depth of the trees. The default is 6.	3 to 10
l2_leaf_reg	l2leaf	Coefficient at the L2 regularisation term of the cost function. Any positive value is allowed. The default is 3.	0.001 to 10
border_count	nBC	The number of splits for numerical features. Allowed values are integers from 1 to 65535 inclusively. Recommended values are up to 255. Larger values slow down the training. The default is 254.	32 to 256, step 32

Note. nTree = number of trees, lr = learning rate, nDT = tree depth, l2leaf = L2 leaf regularisation, nBC = border count

The hyperparameter tuning process in RF and Catboost is done by finding the optimal parameter combination by utilising BO-TPE, Hyperband, and Random Search to maximise accuracy, precision, recall, f1-score, and AUC. BO-TPE is more efficient than traditional techniques such as Random Search and Grid Search because it is able to detect optimal hyperparameter combinations by analysing previously tested values (L. Yang & Shami, 2020). Hyperband and Random Search are among the two techniques that provide better results compared to other hyperparameter tuning techniques when measured using AUC (Suryadi et al., 2024).

Dataset complexity was measured to gain insight into how the model adapts to imbalanced class distribution patterns and high feature complexity. Dataset complexity is

measured using Shannon Entropy (H(x)) and Coefficient of Variation (CV). H(x) measures the diversity of information in the dataset, while CV indicates the amount of variation relative to the mean in the feature distribution (Wiesner & Ladyman, 2019). Each dataset was divided into 80:20, where 80% for training and 20% for testing.

The dataset was tested by cross-validation using scikit-learn, which is a widely used machine learning library. Such testing ensures robust evaluation and reduces the risk of overfitting. The model implementation was done using the Python programming language.

RESULTS AND DISCUSSIONS

The test results show that multi-class datasets tend to provide higher complexity than binary-class datasets, as seen from the combination of H(x) and CV values of each dataset in Table 4.

The impact of hyperparameter tuning results on RF and CatBoost on binary-class datasets with various levels of complexity and class imbalance, as can be seen in Table 4, and can be explained as follows:

The Ionosphere dataset shows a high degree of irregularity (H(x) = 0.7795) and high variation in data (CV = 0.7522). Hyperparameter tuning on RF with all three tuning techniques did not improve accuracy, F1-score, and AUC. Tuning on CatBoost with BO-TPE gave the highest accuracy of 0.9718, and the highest F1-score of 0.9783, but no improvement in AUC, while with HB gave the highest AUC of 0.9861, but provided no improvement on accuracy and F1-score. RS gave the highest accuracy of 0.9718, the highest F1-score of 0.9783, and the highest AUC of 0.9852.

The KC2 dataset shows a moderate degree of irregularity (H(x) = 0.5241) and high data variation (CV = 0.9853). Hyperparameter tuning on RF with all three tuning techniques did not improve accuracy, F1-score, and AUC. Tuning on CatBoost with BO-TPE gave the highest accuracy of 0.8286, and the highest F1-score of 0.5, but no improvement in AUC, while HB provided no improvement in all three metrics. RS also provided improvements in accuracy and F1-score, but still smaller than BO-TPE and no improvement in AUC.

Dataset PC4 shows a fairly low level of data irregularity (H(x) = 0.451) and high data variation (CV = 0.9596). Hyperparameter tuning on RF with BO-TPE gave the highest accuracy of 0.8973 and F1-score of 0.4444, while HB and RS gave no improvement in accuracy, F1-score, and AUC. Tuning on CatBoost RS gave the highest accuracy of 0.9075, the highest F1-score of 0.5263 and the highest AUC of 0.9539. BO-TPE and HB also provide improvements in F1-score and AUC, but both are still lower than RS and no improvement in accuracy.

The impact of hyperparameter tuning results on RF and CatBoost on multi-class datasets with various levels of complexity and class imbalance, as can be seen in Table 4, and can be explained as follows:

1. The Forest Type Mapping dataset shows a high degree of data irregularity (H(x) = 0.814) and low data variation (CV = 0.1723). Hyperparameter tuning on RF with BO-TPE gave the highest accuracy of 0.8762 and F1-score of 0.8737. HB gave the highest AUC improvement of 0.9835 and also improved accuracy and F1-score, though still smaller than BO-TPE. In comparison, RS only gave the highest accuracy of 0.8762, F1-score of 0.8737, and AUC of 0.9831. Tuning on CatBoost with BO-TPE provided the highest AUC of 0.9847, but no improvement in accuracy and F1-score, while HB provided improvement in AUC of 0.9845, but no improvement in accuracy and F1-score. RS provided no improvement in all three metrics.
2. The Predict Students' Dropout and Academic Success dataset shows a low level of data irregularity (H(x) = 0.2034) and a fairly high level of data variation (CV = 0.7047). Hyperparameter tuning on RF with HB gave the highest AUC of 0.8899, but no improvement in accuracy and F1-score, while tuning the RF with BO-TPE gave an increase in AUC from 0.8858 to 0.8888 but no improvement in accuracy and F1-score, while tuning on RF with RS gave an increase in AUC from 0.8858 to 0.8869 but no improvement in accuracy and F1-score. Tuning on CatBoost with HB gave the highest accuracy of 0.7808, the highest F1-score of 0.7664, and the highest AUC of 0.8953, while tuning with BO-TPE gave an improvement in accuracy, F1-score and AUC—though still lower than HB. Meanwhile, tuning with RS gave an improvement in F1-score and AUC, but no improvement in accuracy.
3. The Credit Score Classification dataset shows a moderate level of irregularity (H(x) = 0.5253) and moderate data variation (CV = 0.6141). Hyperparameter tuning on RF with HB provides the highest accuracy of 0.824, the highest F1-score of 0.8239, and the highest AUC of 0.9346, while with BO-TPE and RS also provide improvements in accuracy, F1-score and AUC, but both are still smaller than HB. Tuning on CatBoost with BO-TPE provides the highest accuracy of 0.7739, F1-score of 0.7742, and the highest AUC of 0.9063, while with HB and RS also provide improvements in accuracy, F1-score and AUC, but both are still smaller than BO-TPE.

To complete the comparative analysis, the results of hyperparameter tuning tests on XGBoost are shown in Table 5.

The computation time test results from Table 4 and Table 5 show that hyperparameter tuning with BO-TPE, Hyperband, and Random Search, provides a consistent significant improvement to the execution time compared to using the default parameters. For example, it can be seen that on the relatively small Ionosphere dataset (n=351), the computation time with RF increases from 0.92 seconds (without tuning) to 92.46 seconds with BO-TPE, while CatBoost records 195.19 seconds, and XGBoost shows a more efficient time of 37.3 seconds. On large datasets such as Credit Score Classification (n=100000), tuning with RF results in very high computation times of 32838 seconds (BO-TPE), 44197 seconds (Hyperband), and about 35002 seconds (Random Search). In contrast, tuning with CatBoost showed much more efficient times of 2290 seconds (BO-TPE), 3011 seconds (Hyperband), and 1668 seconds (Random Search). Tuning on XGBoost also shows relatively good efficiency compared to RF, with tuning times of 4535 seconds (BO-TPE), 4951 seconds (Hyperband), and 2245 seconds (Random Search). Thus, although all three algorithms experienced an increase in computation time during tuning, CatBoost was most efficient on large datasets, followed by XGBoost, while RF showed the highest computation time. This efficiency is not always consistent across all datasets, as on small to medium datasets such as KC2 and Forest Type Mapping, XGBoost is often faster than CatBoost and RF. The time variation may also be affected by the characteristics of the dataset and the tuning strategy used, although the search space and tuning configuration are kept consistent.

Based on the test results of hyperparameter tuning on RF and CatBoost from Table 4, we recapitulate the hyperparameter tuning results on binary-class and multi-class datasets by comparing the best metric results, as can be seen in Table 6 and Table 7. The metrics compared include accuracy, F1-score, and AUC between no tuning (default) and after tuning. The after-tuning value taken is the highest value of the three tuning techniques. The recapitulation results can also be seen in graphical form in Figure 7, and for comparison, the recapitulation results of the hyperparameter tuning test on XGBoost can be seen in Table 8.

Based on the results from Table 6, hyperparameter tuning on RF for binary-class datasets with varying degrees of complexity and class imbalance provides an improvement in average accuracy and F1-score, but does not improve average AUC. In contrast, tuning on CatBoost showed an improvement on all three evaluation metrics, namely average accuracy, F1-score, and AUC. The tuning results on XGBoost as presented in Table 8 also show an improvement on all three metrics, while obtaining the best results of all three metrics compared to RF and CatBoost with an average accuracy of 90.61%, F1-score of 69.8%, and AUC of 90.22%. Meanwhile, on multi-class datasets with varying degrees

of complexity and class imbalance as shown in Table 7, both RF and CatBoost showed an average improvement on all metrics after tuning. However, RF's performance was superior to CatBoost on all three metrics. XGBoost also showed consistent performance improvement on all metrics and achieved the best results on all three metrics compared to RF and CatBoost, with an average accuracy of 83.02%, average F1-score of 82.68%, and average AUC of 93.88%.

While this study provides valuable insights into the effect of hyperparameter tuning on model performance on several imbalanced real-world datasets, there are several limitations. Firstly, the level of class imbalance in the datasets used is moderate and does not cover extreme scenarios (e.g. ratio > 95:5). Therefore, the effectiveness of the tuning strategy used in this study may not be fully applicable to cases with extreme class imbalance, so additional techniques are needed for further study. Secondly, the feature complexity on datasets ranging from 22 to 38 features may not be fully generalisable to datasets that have a much larger number of features and high complexity of relationships between features. Thirdly, the search space of the hyperparameters is limited to maintain computational efficiency. While this search space includes commonly used parameters, it is important to realise that a wider search space can potentially result in a more optimal model configuration, but also demands significantly more computational resources. Therefore, the hyperparameter tuning results in this study are a compromise between parameter exploration and resource constraints.

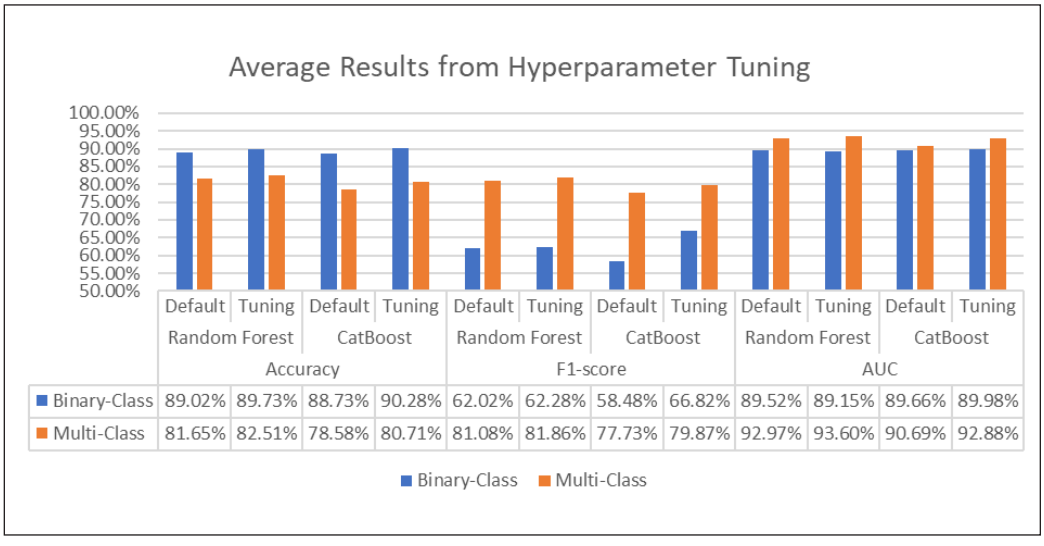

Figure 7. Comparison graph of the average hyperparameter tuning results of RF and CatBoost

Table 4
Hyperparameter tuning test results on RF and CatBoost

Dataset	Tuning	Random Forest						CatBoost					
		Accuracy	Precision	Recall	F1-score	AUC	Best Param/ CT	Accuracy	Precision	Recall	F1-score	AUC	Best Param/ CT
Ionosphere (Binary) Mc=64.1% H(x)=0.7795 CV=0.7522	No Tuning	0.9577	0.9778	0.9565	0.967	0.9726	{nDT=100, maxD=None, minS=2, minL=1, nAtt='sqrt'}/ 0.92s	0.9577	0.9575	0.9783	0.9677	0.9809	{lr=0.03, nDT=6, l2leaf=3, nBC=254}/ 1.59s
	BO-TPE	0.9577	0.9778	0.9565	0.967	0.9726	{nDT=100, maxD=35, minS=2, minL=1, nAtt='log2'}/ 92.46s	0.9718	0.9783	1	0.9783	0.9791	{lr=0.034, nDT=6, l2leaf=0.002, nBC=224}/ 195.19s
	HB	0.9577	0.9778	0.9565	0.967	0.9704	{nDT=200, maxD=18, minS=2, minL=2, nAtt='log2'}/ 74.69s	0.9577	0.9574	0.9783	0.9677	0.9861	{lr=0.04, nDT=7, l2leaf=0.007, nBC=192}/ 181.78s
	RS	0.9577	0.9778	0.9565	0.967	0.9674	{nDT=100, maxD=50, minS=3, minL=1, nAtt='sqrt'}/ 124.84s	0.9718	0.9783	0.9783	0.9783	0.9852	{lr=0.035, nDT=5, l2leaf=0.004, nBC=}/ 184.38s

Table 4 *(continued)*

Dataset	Tuning	Random Forest						CatBoost					
		Accuracy	Precision	Recall	F1-score	AUC	Best Param/ CT	Accuracy	Precision	Recall	F1-score	AUC	Best Param/ CT
KC2 (Binary) Mc=79.5% H(x)=0.5241 CV=0.9853	No Tuning	0.819	0.6154	0.3636	0.4571	0.7623	{nDT=100, maxD=None, minS=2, minL=1, nAtt='sqrt'}/ 0.77s	0.8	0.5385	0.3182	0.4	0.7653	{lr=0.03, nDT=6, l2leaf=3, nBC=254}/ 0.84s
	BO-TPE	0.8	0.5455	0.2727	0.3636	0.7464	{nDT=200, maxD=25, minS=4, minL=10, nAtt=None}/ 141s	0.8286	0.6429	0.4091	0.5	0.7588	{lr=0.017, nDT=6, l2leaf=7.802, nBC=32}/ 78.88s
	HB	0.819	0.6154	0.3636	0.4571	0.7508	{nDT=200, maxD=6, minS=10, minL=10, nAtt='sqrt'}/ 91.71s	0.8	0.5385	0.3182	0.4	0.7593	{lr=0.013, nDT=9, l2leaf=4.383, nBC=64}/ 125.41s
	RS	0.8	0.5385	0.3182	0.4	0.7525	{nDT=50, maxD=20, minS=6, minL=10, nAtt='log2'}/ 83.79s	0.819	0.6154	0.3636	0.4571	0.7593	{lr=0.045, nDT=3, l2leaf=3.314, nBC=128}/ 106.83s

Table 4 *(continued)*

Dataset	Tuning	Random Forest						CatBoost					
		Accuracy	Precision	Recall	F1-score	AUC	Best Param/ CT	Accuracy	Precision	Recall	F1-score	AUC	Best Param/ CT
PC4 (Binary) Mc=87.8% H(x)=0.451 CV=0.9596	No Tuning	0.8938	0.6316	0.3333	0.4364	0.9507	{nDT=100, maxD=None, minS=2, minL=1, nAtt='sqrt'}/ 1.49s	0.9041	0.8333	0.2778	0.4167	0.9435	{lr=0.03, nDT=6, l2leaf=3, nBC=254}/ 0.42s
	BO-TPE	0.8973	0.6667	0.3333	0.4444	0.9456	{nDT=150, maxD=21, minS=4, minL=1, nAtt='sqrt'}/ 180.07s	0.8973	0.65	0.3611	0.4643	0.9452	{lr=0.088, nDT=5, l2leaf=0.117, nBC=128}/ 168.21s
	HB	0.8938	0.6667	0.2778	0.3922	0.9429	{nDT=150, maxD=24, minS=10, minL=1, nAtt='sqrt'}/ 195.6s	0.9007	0.6842	0.3611	0.4727	0.9529	lr=0.034, nDT=8, l2leaf=0.081, nBC=160}/ 153.83s
	RS	0.8904	0.6429	0.25	0.36	0.9494	{nDT=200, maxD= 14, minS=4, minL=2, nAtt='sqrt'}/ 232.72s	0.9075	0.7143	0.4167	0.5263	0.9539	{lr=0.108, nDT=7, l2leaf=5.762, nBC=189}/ 176.21s

Table 4 *(continued)*

Dataset	Tuning	Random Forest						CatBoost					
		Accuracy	Precision	Recall	F1-score	AUC	Best Param/ CT	Accuracy	Precision	Recall	F1-score	AUC	Best Param/ CT
Forest Type Mapping (Multi) Mc=37.3% H(x)=0.814 CV=0.1723	No Tuning	0.8571	0.8644	0.8571	0.8549	0.9732	{nDT=100, maxD=None, minS=2, minL=1, nAtt='sqrt'}/ 1.12s	0.8762	0.8829	0.8762	0.8737	0.9784	lr=0.03, nDT=6, l2leaf=3, nBC=254}/ 2.85s
	BO-TPE	0.8762	0.8829	0.8762	0.8737	0.98	{nDT=100, maxD=43, minS=9, minL=9, nAtt='sqrt'}/ 99.516s	0.8571	0.8688	0.8571	0.8553	0.9847	{lr=0.092, nDT=4, l2leaf=0.67, nBC=256}/ 212.83s
	HB	0.8667	0.8719	0.8667	0.864	0.9835	{nDT=200, maxD=26, minS=9, minL=3, nAtt='log2'}/ 160.01s	0.8571	0.8743	0.8571	0.8554	0.9845	{lr=0.087, nDT=4, l2leaf=1.43, nBC=64}/ 174.4s
	RS	0.8762	0.8829	0.8762	0.8737	0.9831	{nDT=100, maxD=16, minS=7, minL=4, nAtt='log2'}/ 132.03s	0.8667	0.8637	0.8303	0.8443	0.9584	{lr=0.031, nDT=5, l2leaf=1.003, nBC=128}/ 396.04s

Table 4 *(continued)*

Dataset	Tuning	Random Forest						CatBoost					
		Accuracy	Precision	Recall	F1-score	AUC	Best Param/ CT	Accuracy	Precision	Recall	F1-score	AUC	Best Param/ CT
Predict Students' Dropout and Academic Success (Multi) Mc=49.9% H(x)=0.2034 CV=0.7047	No Tuning	0.7751	0.7607	0.7751	0.7603	0.8858	{nDT=100, maxD=None, minS=2, minL=1, nAtt='sqrt'}/ 3.7s	0.7672	0.7467	0.7672	0.7426	0.8826	{lr=0.03, nDT=6, l2leaf=3, nBC=254}/ 5.77s
	BO-TPE	0.7751	0.7606	0.7751	0.7582	0.8888	{nDT=200, maxD=20, minS=6, minL=3, nAtt='log2'}/ 326.4s	0.7695	0.7544	0.7695	0.7557	0.8937	{lr=0.089, nDT=7, l2leaf=0.083, nBC=256}/ 296.84s
	HB	0.7672	0.7527	0.7672	0.7513	0.8899	{nDT=200, maxD=38, minS=8, minL=2, nAtt='sqrt'}/ 654.68s	0.7808	0.7684	0.7808	0.7664	0.8953	{lr=0.082, nDT=7, l2leaf=0.564, nBC=256}/ 254.57s
	RS	0.7661	0.7498	0.7661	0.7489	0.8869	{nDT=150, maxD=29, minS=7, minL=1, nAtt='sqrt'}/ 519.8s	0.765	0.746	0.765	0.749	0.894	{lr=0.09, nDT=6, l2leaf=0.002, nBC=96}/ 404.95s

Table 4 *(continued)*

Dataset	Tuning	Random Forest						CatBoost					
		Accuracy	Precision	Recall	F1-score	AUC	Best Param/ CT	Accuracy	Precision	Recall	F1-score	AUC	Best Param/ CT
Credit Score Classification (Multi) Mc=53.2% H(x)=0.5253 CV=0.6141	No Tuning	0.8173	0.8175	0.8173	0.8172	0.9301	{nDT=100, maxD=None, minS=2, minL=1, nAtt='sqrt'}/ 215.73s	0.714	0.7209	0.714	0.7155	0.8596	{lr=0.03, nDT=6, l2leaf=3, nBC=254}/ 13.16s
	BO-TPE	0.8228	0.8232	0.8228	0.8227	0.9342	{nDT=150, maxD=45, minS=3 minL=1, nAtt='sqrt'}/ 32838.06s	0.7739	0.7753	0.7739	0.7742	0.9063	lr=0.099, nDT=10, l2leaf=0.002, nBC=32}/ 2290.56s
	HB	0.8240	0.8243	0.8240	0.8239	0.9346	{nDT=200, maxD=37, minS=2, minL=1, nAtt='log2'}/ 44197.04s	0.7722	0.7736	0.7722	0.7725	0.9038	{lr=0.099, nDT=10, l2leaf=0.01, nBC=128}/ 3011.68s
	RS	0.8225	0.8228	0.8225	0.8224	0.934	{nDT=200, maxD=45, minS=3, minL=1, nAtt='log2'}/ 35002.73s	0.7546	0.7577	0.7546	0.7552	0.8946	{lr=0.076, nDT=10, l2leaf=0.702, nBC=160}/ 1668.35s

Note. Mc = Majority class, CT = Computational Time

Table 5
Hyperparameter tuning test results on XGBoost

Dataset	Tuning	XGBoost					
		Accuracy	Precision	Recall	F1-score	AUC	Best Param/CT
Ionosphere (Binary) Mc=64.1% H(x)=0.7795 CV=0.7522	No Tuning	0.9437	0.9565	0.9565	0.9565	0.973	{nEst=100, lr=0.3, maxD=6, minCW=1, subS=1.0, colS=1.0, gm=0}/1.89s
	BO-TPE	0.9437	0.9375	0.9783	0.9574	0.9809	{nEst=300, lr=0.2756, maxD=7, minCW=1, subS=0.5515, colS=0.6289, gm=3.4934}/37.3s
	HB	0.9578	0.9575	0.9783	0.9677	0.973	{nEst=500, lr=0.1132, maxD=4, minCW=1, subS=0.548, colS=0.723, gm=0.057}/54.58s
	RS	0.9718	0.9583	1	0.9787	0.9748	{nEst=200, lr=0.0937, maxD=12, minCW=3, subS=0.6879, colS=0.6907, gm=0.6009}/35.61s
KC2 (Binary) Mc=79.5% H(x)=0.5241 CV=0.9853	No Tuning	0.8	0.5294	0.4091	0.4615	0.7218	{nEst=100, lr=0.3, maxD=6, minCW=1, subS=1.0, colS=1.0, gm=0}/0.47s
	BO-TPE	0.819	0.6154	0.3636	0.4571	0.75	{nEst=200, lr=0.101, maxD=4, minCW=9, subS=0.9865, colS=0.8928, gm=4.5787}/20.74s
	HB	0.8286	0.6429	0.4091	0.5	0.7459	{nEst=200, lr=0.1574, maxD=5, minCW=9, subS=0.9226, colS=0.7894, gm=4.8749}/23.29s
	RS	0.819	0.6	0.4091	0.4865	0.776	{nEst=200, lr=0.0685, maxD=5, minCW=2, subS=0.5355, colS=0.9736, gm=4.9697}/26.67s
PC4 (Binary) Mc=87.8% H(x)=0.451 CV=0.9596	No Tuning	0.91	0.6923	0.5	0.5806	0.9481	{nEst=100, lr=0.3, maxD=6, minCW=1, subS=1.0, colS=1.0, gm=0}/0.6s
	BO-TPE	0.9144	0.6897	0.5556	0.6154	0.9422	{nEst=400, lr=0.1024, maxD=5, minCW=1, subS=0.5402, colS=0.6193, gm=0.7046}/76.04s
	HB	0.9144	0.7037	0.5278	0.6032	0.9498	{nEst=300, lr=0.1349, maxD=6, minCW=1, subS=0.7472, colS=0.7754, gm=0.1591}/59.7s
	RS	0.9178	0.7308	0.5278	0.6129	0.9428	{nEst=400, lr=0.07, maxD=8, minCW=2, subS=0.5426, colS=0.8002, gm=0.4593}/50.59s

Table 5 *(continued)*

Dataset	Tuning	XGBoost					
		Accuracy	Precision	Recall	F1-score	AUC	Best Param/CT
Forest Type Mapping (Multi) Mc=37.3% H(x)=0.814 CV=0.1723	No Tuning	0.8762	0.8937	0.8762	0.8745	0.9743	{nEst=100, lr=0.3, maxD=6, minCW=1, subS=1.0, colS=1.0, gm=0}/3.85s
	BO-TPE	0.8857	0.8997	0.8857	0.8837	0.9828	{nEst=100, lr=0.055, maxD=9, minCW=10, subS=0.9621, colS=0.5985, gm=0.723}/72.35s
	HB	0.8857	0.8997	0.8857	0.8837	0.9814	{nEst=400, lr=0.1183, maxD=6, minCW=10, subS=0.9712, colS=0.5378, gm=1.6244}/92.55s
	RS	0.8857	0.8997	0.8857	0.8837	0.982	{nEst=200, lr=0.0283, maxD=5, minCW=7, subS=0.9534, colS=0.6249, gm=1.5405}/102.19s
Predict Students' Dropout and Academic Success (Multi) Mc=49.9% H(x)=0.2034 CV=0.7047	No Tuning	0.7632	0.7603	0.7672	0.7622	0.8887	{nEst=100, lr=0.3, maxD=6, minCW=1, subS=1.0, colS=1.0, gm=0}/1.23s
	BO-TPE	0.7774	0.7694	0.7774	0.7693	0.8964	{nEst=500, lr=0.022, maxD=10, minCW=4, subS=0.849, colS=0.5071, gm=0.8835}/385.34s
	HB	0.7661	0.7562	0.7661	0.7578	0.8937	{nEst=500, lr=0.0294, maxD=5, minCW=7, subS=0.8022, colS=0.6161, gm=0.5224}/283.81s
	RS	0.774	0.765	0.774	0.7661	0.895	{nEst=300, lr=0.0283, maxD=8, minCW=1, subS=0.7943, colS=0.7567 gm=1.0436}/192.44s
Credit Score Classification (Multi) Mc=53.2% H(x)=0.5253 CV=0.6141	No Tuning	0.7952	0.7956	0.7952	0.7953	0.9181	{nEst=100, lr=0.3, maxD=6, minCW=1, subS=1.0, colS=1.0, gm=0}/8.91s
	BO-TPE	0.8249	0.8254	0.8249	0.8249	0.9371	{nEst=500, lr=0.0681, maxD=12, minCW=2, subS=0.833, colS=0.7225, gm=0.08}/4535.25s
	HB	0.8274	0.8278	0.8274	0.8274	0.9372	{nEst=500, lr=0.0802, maxD=10, minCW=1, subS=0.8024, colS=0.5474, gm=0.0587}/4951s
	RS	0.8229	0.8233	0.8229	0.823	0.9344	{nEst=500, lr=0.0817, maxD=8, minCW=4, subS=0.7341, colS=0.5118, gm=0.098}/2245.06s

Note. Mc = Majority class, CT = Computational Time, nEst = n_estimators, lr = learning rate, maxD = maximum depth, minCW = minimum child weight, subS = subsample, colS = colsample_bytree, gm = gamma

Table 6
Recapitulation of hyperparameter tuning results on binary-class dataset using RF and CatBoost Models

Dataset	Accuracy				F1-score				AUC			
	Random Forest		CatBoost		Random Forest		CatBoost		Random Forest		CatBoost	
	Default	Tuning	Default	Tuning	Default	Tuning	Default	Tuning	Default	Tuning	Default	Tuning
Ionosphere H(x)=0.7795 CV=0.7522	95.77%	95.77% (BO, HB, RS)	95.77%	97.18% (BO-TPE, RS)	96.7%	96.7% (BO, HB, RS)	96.77%	97.83% (BO-TPE, RS)	97.26%	97.26% (BO-TPE)	98.09%	98.61% (HB)
KC2 H(x)=0.5241 CV=0.9853	81.9%	81.9% (HB)	80%	82.9% (BO-TPE)	45.71%	45.71% (HB)	40%	50% (BO-TPE)	76.23%	75.25% (RS)	76.53%	75.93% (HB, RS)
PC4 H(x)=0.451 CV=0.9596	89.38%	89.73% (BO-TPE)	90.41%	90.75% (RS)	43.64%	44.44% (BO-TPE)	41.67%	52.63% (RS)	95.07%	94.94% (RS)	94.35%	95.39% (RS)
Average	89.02%	89.73%	88.73%	90.28%	62.02%	62.28%	59.48%	66.82%	89.52%	89.15%	89.66%	89.98%

Table 7
Recapitulation of hyperparameter tuning results on multi-class dataset using random forest (RF) and CatBoost Models

Dataset	Accuracy				F1-score				AUC			
	Random Forest		CatBoost		Random Forest		CatBoost		Random Forest		CatBoost	
	Default	Tuning	Default	Tuning	Default	Tuning	Default	Tuning	Default	Tuning	Default	Tuning
Forest Type Mapping H(x)=0.814 CV=0.1723	85.71%	87.62% (BO-TPE, RS)	87.62%	86.67% (RS)	85.49%	87.37% (BO-TPE, RS)	87.37%	85.54% (HB)	97.32%	98.35% (HB)	97.84%	98.47% (BO-TPE)
Predict Students Dropout H(x)=0.2034 CV=0.7047	77.51%	77.51% (BO-TPE)	76.72%	78.08% (HB)	76.03%	75.82% (BO)	74.26%	76.64% (HB)	88.58%	88.99% (HB)	88.26%	89.53% (HB)
Credit Score H(x)=0.5253 CV=0.6141	81.73%	82.4% (HB)	71.4%	77.39% (BO-TPE)	81.72%	82.39% (HB)	71.55%	77.42% (BO-TPE)	93.01%	93.46% (HB)	85.96%	90.63% (BO-TPE)
Average	81.65%	82.51%	78.58%	80.71%	81.08%	81.86%	77.73%	79.87%	92.97%	93.6%	90.69%	92.88%

Table 8
Recapitulation of hyperparameter tuning results on binary-class and multi-class dataset using XGBoost model

Dataset	Accuracy		F1-score		AUC	
	Default	Tuning	Default	Tuning	Default	Tuning
Binary-Class						
Ionosphere H(x)=0.7795 CV=0.7522	94.37%	97.18 (RS)	95.65%	97.87% (RS)	97.3%	98.08% (BO-TPE)
KC2 H(x)=0.5241 CV=0.9853	80%	82.86% (HB)	46.15%	50% (HB)	72.18%	77.6% (RS)
PC4 H(x)=0.451 CV=0.9596	91%	91.78% (RS)	58.06%	61.54% (BO-TPE)	94.81%	94.98% (HB)
Average	88.46%	90.61%	66.62%	69.8%	88.1%	90.22%
Multi-Class						
Forest Type Mapping H(x)=0.814 CV=0.1723	87.62%	88.57% (BO-TPE, HB, RS)	87.45%	88.37% (BO-TPE, HB, RS)	97.43%	98.28% (BO-TPE)
Predict Students Dropout H(x)=0.2034 CV=0.7047	76.32%	77.74% (BO-TPE)	76.22%	76.93% (BO-TPE)	88.87%	89.64% (BO-TPE)
Credit Score H(x)=0.5253 CV=0.6141	79.52%	82.74% (HB)	79.53%	82.74% (HB)	91.81%	93.72% (HB)
Average	81.15%	83.02%	81.07%	82.68%	92.7%	93.88%

CONCLUSIONS

Datasets with a high degree of variation with CV values close to 1 (KC2 and PC4 datasets) tend to produce quite low F1-score compared to accuracy values. Hyperparameter tuning on RF for binary-class datasets with varying degrees of complexity and class imbalance improved the average accuracy and F1-score, but did not improve the average AUC value. However, hyperparameter tuning on RF showed uniform improvement for all three metrics on multi-class datasets. In contrast, the tuning of CatBoost and XGBoost showed an average improvement on all three evaluation metrics on both binary-class and multi-class datasets. XGBoost achieved the best results on all three metrics on both binary and multi-class datasets compared to RF and CatBoost. In terms of computational efficiency, RF, CatBoost, and XGBoost experienced a significant increase in computation time during the tuning process, with the time variation depending on the complexity of the dataset and the tuning technique used. Although all three algorithms experience an increase in computation time during tuning, CatBoost is generally the most efficient on large datasets, followed by

XGBoost, while RF shows the highest computation time. Meanwhile, XGBoost was most efficient on small datasets, followed by RF and CatBoost.

As a follow-up to the limitations identified in this study, future research could be directed towards the following. Firstly, the use of datasets with more extreme levels of class imbalance to test the extent to which the effectiveness of hyperparameter tuning techniques can be sustained or even degraded under more challenging conditions. Secondly, feature complexity can also be increased by using high-dimensional datasets to assess the scalability and performance of tuning in a more statistically complex environment. Thirdly, a wider and more aggressive exploration of the hyperparameter search space can be conducted, especially with the support of larger computational resources, to compare whether the trade-off between efficiency and accuracy chosen in this study remains relevant. Fourth, the scope of the algorithms used can be expanded to include other models, even deep learning approaches, so that the tuning results can be compared more thoroughly. Fifth, the interpretability aspect of the model could also be an interesting focuss, particularly to explain how the optimal combination of parameters can influence the model's decision-making, as well as how this impacts the context of real applications.

ACKNOWLEDGEMENTS

We would like to thank Dr. Sutarman, M.Sc. and Dr. Erna Budhiarti Nababan, M.IT. from the Universitas Sumatera Utara, Medan, Indonesia for the support, guidance, and direction to make this article could be produced and published.

ABBREVIATION

AUC	:	Area Under the Receiver Operating Characteristic (ROC) Curve
BO-TPE	:	Bayesian Optimisation with Tree-structured Parzen Estimator
colS	:	Colsample_bytree
CT	:	Computational Time
CV	:	Coefficient of Variation
GBDT	:	Gradient Boosted Decision Trees
gm	:	Gamma
GP	:	Gaussian Process
H(x)	:	Shannon Entropy
HB	:	Hyperband
l2leaf	:	L2 leaf regularisation
lr	:	Learning rate
maxD	:	Maximum depth
Mc	:	Majority class

minCW : Minimum child weight
minL : Minimum samples per leaf
minS : Minimum samples to split a node
nAtt : Number of attributes
nBC : Border count
nDT : Tree depth
nEst : Number of estimators
nTree : Number of trees
RF : Random Forest
RS : Random Search
subS : Subsample
TPE : Tree-structured Parzen Estimator

REFERENCES

Abdellatif, A., Abdellatef, H., Kanesan, J., Chow, C.-O., Chuah, J. H., & Gheni, H. M. (2022). An effective heart disease detection and severity level classification model using machine learning and hyperparameter optimisation methods. *IEEE Access*, *10*, 79974-79985. https://doi.org/10.1109/ACCESS.2022.3191669

Albahli, S. (2023). Efficient hyperparameter tuning for predicting student performance with bayesian optimisation. *Multimedia Tools and Applications*, *83*(17), 52711-52735. https://doi.org/10.1007/s11042-023-17525-w

Alshamy, R., Ghurab, M., Othman, S., & Alshami, F. (2021). Intrusion detection model for imbalanced dataset using SMOTE and random forest algorithm. In *Communications in Computer and Information Science* (Vol. 1487, pp. 361-378). https://doi.org/10.1007/978-981-16-8059-5_22

Andri, Yunis, R., & Tanti. (2023). Optimising random forest classification using chi-square and SMOTE-ENN on student drop-out data. In *2023 Eighth International Conference on Informatics and Computing (ICIC)* (pp. 1-5). https://doi.org/10.1109/ICIC60109.2023.10382055

Ayyannan, M. (2024). Accuracy enhancement of machine learning model by handling imbalanced data. In *2024 International Conference on Expert Clouds and Applications (ICOECA)* (pp. 593-599). https://doi.org/10.1109/ICOECA62351.2024.00109

Barella, V. H., Garcia, L. P. F., de Souto, M. C. P., Lorena, A. C., & de Carvalho, A. C. P. L. F. (2021). Assessing the data complexity of imbalanced datasets. *Information Sciences*, *553*, 83-109. https://doi.org/10.1016/j.ins.2020.12.006

Borah, P., & Gupta, D. (2021). Robust twin bounded support vector machines for outliers and imbalanced data. *Applied Intelligence*, *51*(8), 5314-5343. https://doi.org/10.1007/s10489-020-01847-5

Chen, W., Yang, K., Yu, Z., Shi, Y., & Chen, C. L. P. (2024). A survey on imbalanced learning: Latest research, applications and future directions. *Artificial Intelligence Review*, *57*(6), Article 137. https://doi.org/10.1007/s10462-024-10759-6

Gu, Q., Tian, J., Li, X., & Jiang, S. (2022). A novel random forest integrated model for imbalanced data classification problem. *Knowledge-Based Systems*, *250*, Article 109050. https://doi.org/10.1016/j.knosys.2022.109050

Hancock, J. T., & Khoshgoftaar, T. M. (2020). CatBoost for big data: An interdisciplinary review. *Journal of Big Data*, *7*(1), Article 94. https://doi.org/10.1186/s40537-020-00369-8

Hazarika, B. B., & Gupta, D. (2022). Density weighted twin support vector machines for binary class imbalance learning. *Neural Processing Letters*, *54*(2), 1091-1130. https://doi.org/10.1007/s11063-021-10671-y

Hazarika, B. B., & Gupta, D. (2023). Affinity-based fuzzy kernel ridge regression classifier for binary class imbalance learning. *Engineering Applications of Artificial Intelligence*, *117*, Article 105544. https://doi.org/10.1016/j.engappai.2022.105544

Hazarika, B. B., Gupta, D., & Borah, P. (2023). Fuzzy twin support vector machine based on affinity and class probability for class imbalance learning. *Knowledge and Information Systems*, *65*(12), 5259-5288. https://doi.org/10.1007/s10115-023-01904-8

Huang, C.-M., Lin, C.-H., Hung, C.-S., Zeng, W.-H., Zheng, Y.-C., & Tsai, C.-M. (2024). Utilising nearest-neighbor clustering for addressing imbalanced datasets in bioengineering. *Bioengineering*, *11*(4), Article 345. https://doi.org/10.3390/bioengineering11040345

Ilemobayo, J. A., Durodola, O., Alade, O., Awotunde, O. J., Olanrewaju, A. T., Falana, O., Ogungbire, A., Osinuga, A., Ogunbiyi, D., Ifeanyi, A., Odezuligbo, I. E., & Edu, O. E. (2024). Hyperparameter tuning in machine learning: A comprehensive review. *Journal of Engineering Research and Reports*, *26*(6), 388-395. https://doi.org/10.9734/jerr/2024/v26i61188

Imani, M., & Arabnia, H. R. (2023). Hyperparameter optimisation and combined data sampling techniques in machine learning for customer churn prediction: A comparative analysis. *Technologies*, *11*(6). https://doi.org/10.20944/preprints202308.1478.v2

Manda, V. T., Kondapalli, D., Malla, A. S., M, J. N., & Charan, Y. (2024). Imbalanced data challenges and their resolution to improve fraud detection in credit card transactions. *Research Square*. https://doi.org/10.21203/rs.3.rs-3962043/v1

Matar, N., Sowan, B., & Al-Jaber, A. (2024). Evaluating models performance for credit risk detection for imbalanced data. In *2024 2nd International Conference on Cyber Resilience (ICCR)* (pp. 1-6). https://doi.org/10.1109/ICCR61006.2024.10532912

Mendes, P., Casimiro, M., Romano, P., & Garlan, D. (2023). HyperJump: Accelerating HyperBand via risk modelling. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(8), 9143-9152. https://doi.org/10.1609/aaai.v37i8.26097

Mustapha, M. T., & Ozsahin, D. U. (2024). Class imbalance and its impact on predictive models for binary classification of disease: A comparative analysis. In W. A. Zgallai & D. U. Ozsahin (Eds.), *Artificial intelligence and image processing in medical imaging* (pp. 389-408). https://doi.org/10.1016/B978-0-323-95462-4.00014-5

Patel, V., & Bhavsar, H. (2024). Novel approach for addressing data imbalance. *Research Square*. https://doi.org/10.21203/rs.3.rs-4023156/v1

Praptiwi, D. Y., Kurnia, A., Fitrianto, A., & Ernawati, F. (2024). Random forest and CatBoost with handling imbalanced class for detection of risk factors anemia in children (5-12 years). *International Journal of*

Scientific Research in Science, Engineering and Technology, *11*(3), 302-312.https://doi.org/10.32628/IJSRSET24113134

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2017). CatBoost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, *30*, 6638-6648.http://arxiv.org/abs/1706.09516

Ren, L., Seklouli, A. S., Zhang, H., Wang, T., & Bouras, A. (2023). An adaptive Laplacian weight random forest imputation for imbalance and mixed-type data. *Information Systems*, *111*, Article 102122.https://doi.org/10.1016/j.is.2022.102122

Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal*, *20*(1), 3-29.https://doi.org/10.1177/1536867X20909688

Suryadi, M. K., Herteno, R., Saputro, S. W., Faisal, M. R., & Nugroho, R. A. (2024). Comparative study of various hyperparameter tuning on random forest classification with SMOTE and feature selection using genetic algorithm in software defect prediction. *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, *6*(2), 137-147. https://doi.org/10.1177/1536867X20909688

Vo, H.-T., Ngoc, H. T., & Quach, L.-D. (2023). An approach to hyperparameter tuning in transfer learning for driver drowsiness detection based on bayesian optimisation and random search. *International Journal of Advanced Computer Science and Applications*, *14*(4), 828-837. https://doi.org/10.14569/IJACSA.2023.0140492

Wen, C., Zhao, Q., Li, M., Liu, J., Li, M., & Zhao, X. (2022). Multi-objective optimisation based on hyperparameter random forest regression for linear motor design. *International Journal of Machine Learning and Cybernetics*, *13*(10), 2929-2942. https://doi.org/10.1007/s13042-022-01573-z

Wiesner, K., & Ladyman, J. (2019). Measuring complexity. In *Trying to measure globalisation* (pp. 19-34). https://link.springer.com/10.1007/978-94-007-2807-3_2

Yang, L., & Shami, A. (2020). On hyperparameter optimisation of machine learning algorithms: Theory and practice. *Neurocomputing*, *415*, 295-316. https://doi.org/10.1016/j.neucom.2020.07.061

Yang, Y., Khorshidi, H. A., & Aickelin, U. (2024). A review on over-sampling techniques in classification of multi-class imbalanced datasets: Insights for medical problems. *Frontiers in Digital Health*, *6*, Article 1430245. https://doi.org/10.3389/fdgth.2024.1430245

Yuliana, H., Iskandar, Hendrawan, Basuki, S., Hidayat, M. R., Charisma, A., & Vidyaningtyas, H. (2024). Hyperparameter optimisation of random forest for 5G coverage prediction. *Buletin Pos dan Telekomunikasi*, *22*(1), 75-90. https://doi.org/10.17933/bpostel.v22i1.390

Zamzam, Y. F., Saragih, T. H., Herteno, R., Muliadi, Nugrahadi, D. T., & Huynh, P.-H. (2024). Comparison of CatBoost and random forest methods for lung cancer classification using hyperparameter tuning Bayesian optimisation-based. *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, *6*(2), 125-136. https://doi.org/10.35882/jeeemi.v6i2.382

Zheng, Q., Yu, C., Cao, J., Xu, Y., Xing, Q., & Jin, Y. (2024). Advanced payment security system: XGBoost, LightGBM and SMOTE integrated. In *2024 IEEE International Conference on Metaverse Computing, Networking, and Applications (MetaCom)* (pp. 336-342). https://doi.org/10.1109/MetaCom62920.2024.00063