Journal of Information Systems Engineering and Management
2025, 10(28s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Enhancement of Random Forest Applied to Program-Recommendation for Waitlisted Applicants

John Vincent L. Guanco[1], Gian Karlo V. Ramos[2], Vivien A. Agustin[3], Erwin D. Marcelo[4]

[1] *Computer Science, College of Information Systems and Technology Management, University of the City of Manila, Manila, Philippines. jvlguanco@gmail.com*

[2] *Computer Science, College of Information Systems and Technology Management, University of the City of Manila, Manila, Philippines. karloramos0129@gmail.com*

[3] *Faculty in Computer Science, College of Information Systems and Technology Management, University of the City of Manila, Manila, Philippines. vaagustin@plm.edu.ph*

[4] *Admission Officer, College Admission Office, University of the City of Manila, Manila, Philippines. edmarcelo@plm.edu.ph*

ARTICLE INFO

Received: 30 Dec 2024

Revised: 12 Feb 2025

Accepted: 26 Feb 2025

ABSTRACT

This study introduces an Enhanced Random Forest model for a program recommendation system aimed at supporting waitlisted applicants. The study integrates feature engineering, feature selection, and a hybrid hyperparameter tuning approach to improve the model's classification accuracy, reliability, and interpretability. Feature engineering was used to capture interrelationship between program choices and associated college departments, allowing the model to recognize nuance relationships. Recursive Feature Elimination systematically eliminated the least significant features, ensuring that the model focuses on features that have high predictive attributes. To mitigate the limitations of traditional hyperparameter tuning methods, a hybrid approach of RandomizedSearchCV and GridSearchCV was utilized to effectively find optimal parameters while minimizing computation costs. A dataset of 1,505 applicants who underwent the Pamantasan ng Lungsod ng Maynila Admission Test for school year 2024 was utilized to train models. The Enhanced Random Forest was compared against Traditional Random Forest, AdaBoost, Gradient Boosting, and Extra Trees Classifier using an 8-fold cross-validation with different scoring metrics such as accuracy, F1-Score, and ROC-AUC. Results show a significant improvement, where the enhanced model achieved an accuracy of 75.10% with an F1-Score of 74.23% and an ROC-AUC score of 96.57%, outperforming other models across all metrics. The Enhanced Random Forest also achieved a lower standard deviation across all metrics resulting in a stable model compared to the other models. A web application using the enhanced model was developed to support the program recommendation process. The findings demonstrate the effectiveness of the proposed enhancements in addressing challenges in an admission-based recommendation system by offering a robust framework for academic decision-making.

Keywords: Feature Engineering, Feature Selection, GridSearchCV, Hyperparameters, Random Forest, RandomizedSearchCV

INTRODUCTION

Academic admission is the primary process that student applicants initiate with their academic endeavor [1]. Credential evaluation is an in-depth process involving strict assessment to identify the most fitting student prospects who would meet the criteria for getting accepted [2]. Being admitted to college is quite complex and usually involves precise verification and evaluation of the student's academic ability [3]. Hence, machine learning algorithms are adopted to predict and classify the most fitting program for an applicant.

Random Forest is a machine learning model for supervised learning that is typically used for regression and classification tasks. During training, it produces several decision trees that aim to attain a unified output [4]. This randomized nature of the model allows it to handle complex datasets effectively. It delivers crucial information on the importance of characteristics, which assists in choosing the vital features in the datasets [5]. Although random forests are widely used, there are still limitations within the algorithm that need to be modified for enhancement. It is susceptible to overfitting due to noisy dataset [6]. This is usually fixed by applying hyperparameter tuning, wherein hyperparameters, which refer to parameters that need to be set before training the model, are thoroughly chosen to find the best fit for the model. In this way, the model will have a better learning process, which leads to better performance [7].

Despite its evident advantages, the application of program recommendations for classification poses challenges as it relates to handling complex data, considering that this field of application needs to address the intricate relationships within the admission datasets. Specifically, recommendation systems have a common significant challenge with regards to capturing feature interactions, which is a crucial factor to achieve a desirable performance and accuracy of the machine learning model [8]. This problem underscores the need for methodologies such as feature engineering to derive those relationships and correlated features to achieve a better classification model. Additionally, the randomized nature of its feature selection process also poses drawbacks as it can overlook critical features that may be a vital factor, potentially negatively affecting the model performance.

This study proposes an enhancement of the Random Forest model by including a feature engineering method, feature selection, and a hybrid hyperparameter tuning. Feature engineering will be applied to the dataset by determining the similarities between the different classes [8]. In the context of the admission process, each class are grouped based on their corresponding college department. Then, the interactions between these college departments and program choices of the applicants will be considered. These interactions will aid the model to capture the underlying relationships between the features which can enhance the classification capability of the model [9]. Furthermore, the feature selection will use Recursive Feature Elimination (RFE) which removes certain features that had the lowest feature importance score. This method will rank features recursively and will continually remove features that are deemed least important until the desired number of features is achieved. Lastly, the hybrid approach to hyperparameter tuning utilizing the RandomizedSearchCV and GridSearchCV. It is utilized to find the optimized parameters of the Random Forest Model. RandomizedSearchCV will obtain optimal hyperparameters by randomly choosing configurations [10], and GridSearchCV will obtain optimal hyperparameters by going through various combinations within a specified range [11]. By integrating the two tuning approaches, this study aims to provide a method to obtain the best and optimal combination of parameters, which improves the accuracy and efficiency of the classification model [12].

METHODOLOGY

a) Random Forest

Random Forest was introduced by Leo Breiman [13] which is an ensemble machine learning model for regression and classification. The ensemble property of the algorithm allows it to produce several decision trees that aim to attain a unified output [4]. It would build a decision using different samples and use the average for any regression problems and majority vote for classification problems [14] as shown in Figure 1.

Classification is a type of supervised machine learning methodology in which the objective of the model is to provide prediction on the category or class of new samples based on a set of features from a labeled dataset [15]. It follows a repetitive process that learns from the patterns and group data based on their predetermined classes. In classification algorithms, the dataset is divided into two sections (i.e., training data and testing data). It will use the training dataset, which is typically the larger set of data, to learn through the patterns or trends within it. The remaining testing dataset will be used for evaluating the model's performance by testing it to perform predictions on new unseen data which are not familiar to the model.

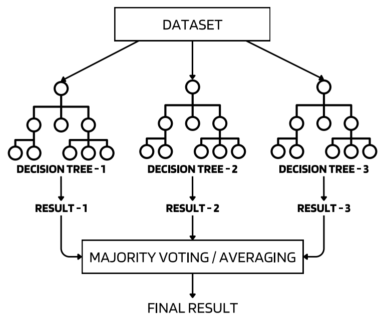


Figure 1. Visualization of Random Forest Model

b) System Framework

The framework of the proposed program recommendation system that is aimed at helping program advising for waitlisted applicants of the Pamantasan ng Lungsod ng Maynila (PLM) Admission Test (PLMAT) is presented in Figure 2.

A dataset of 1,882 applicants who participated in the PLMAT for the school year 2024 with confirmed programs were obtained from the admission office of PLM. Each record includes three program choices, academic strand, General Weighted Average (GWA), and program rating, which serves as the dataset's features. The confirmed program of each applicant will serve as the target label of the dataset.

The dataset was standardized to ensure consistency across all features. Records with missing target labels were removed, as their presence could compromise the analysis of program selection patterns. Finally, outliers were identified using the Z-score method and removed based on a defined threshold, minimizing the influence of extreme values and enhancing the dataset's overall reliability for analysis. This ensures that the model can effectively process and learn from a clean dataset, as the model's performance is tied to the quality of the data it utilizes [16].

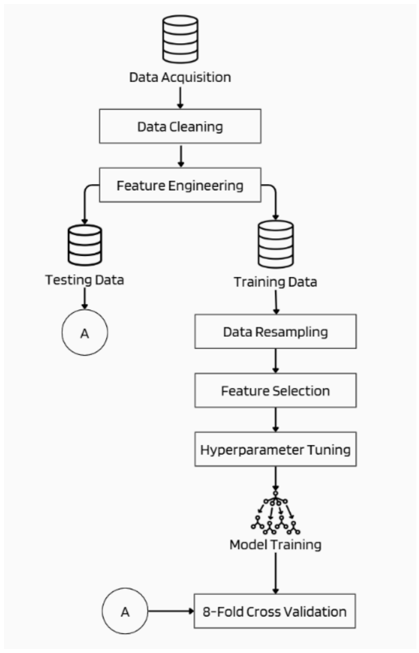


Figure 2. System Framework of Random Forest Model for Program Recommendation

c) Feature Engineering

Feature engineering was applied to the dataset by identifying the similarities between the classes of the program choices. Each class of the program choice falls under a specific college department. Interactions between the college departments were also identified to determine the similarities between the program choices of an applicant. By capturing these interactions, the model can gain insights into the relationship between features [8]. This approach would allow a systematic classification of programs based on underlying similarities and would reduce feature redundancy and improve the model's interpretability that would enhance the classification performance of the model [9].

Table 1 presents the classes for the new feature column based on the similarities of each class in the program choices. This will generate three new columns based on the determined departments of programs since an applicant has three program choices when they apply for the admission. The generated features will then be compared to each other, which will generate additional columns determining if the program choice of an applicant falls within the same department of the college.

Table 1. Departments PLM and their respective programs

Department	**Programs**
College of Architecture and Sustainable Built Environments (CASBE)	BSArch
College of Accountancy (CA)	BSA
College of Business Administration (CBA)	BSBAMM, BSBAFM, BSBABE, BSBAHRM, BSREM, BSTM, BSEntrep, BSHM
College of Engineering (CEng)	BSCHE, BSCE, BSCpE, BSEE, BSECE, BSME
College of Information Systems & Technology Management (CISTM)	BSIT, BSCS
College of Humanities, Arts and Social Sciences (CHASS)	BAC, BSSW
College of Nursing (CN)	BSN
College of Physical Therapy (CPT)	BSPT
College of Science (CS)	BSBio, BSMath, BSChem, BSPsy
College of Education (CED)	BSEd-Eng, BSEd-Fil, BSEd-Math, BSEd-SS, BSEd-Sci,
College of Public Administration (CPA)	BPA

The dataset is split into training and testing subsets using the holdout method [17], with 80% of the data allocated for training and 20% for testing. Data resampling will only be applied to the training dataset, while feature selection will be conducted on both the training and testing datasets. This approach aims to improve the quality of data and improve the efficiency of the subsequent processes.

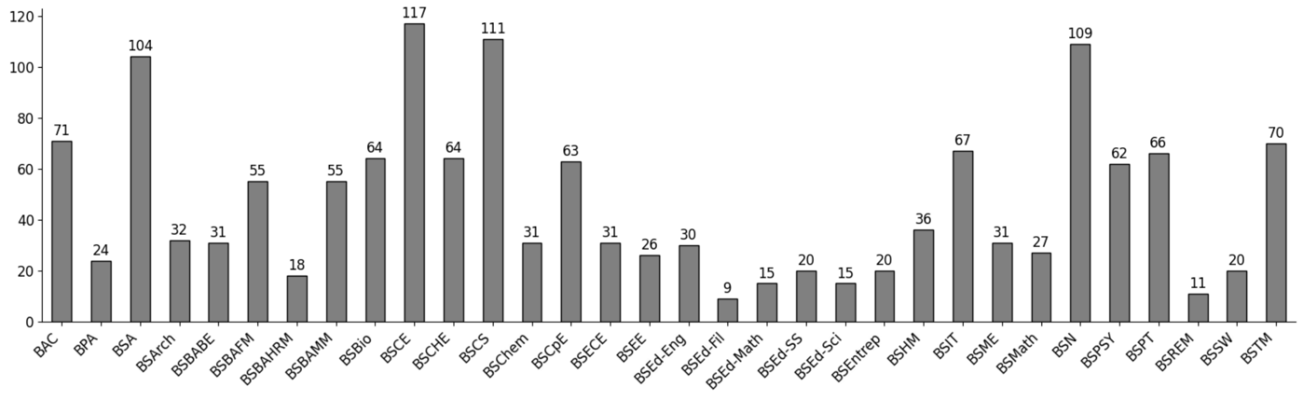


Figure 1. Sample Count and Classes of the Original Distribution of the Dataset

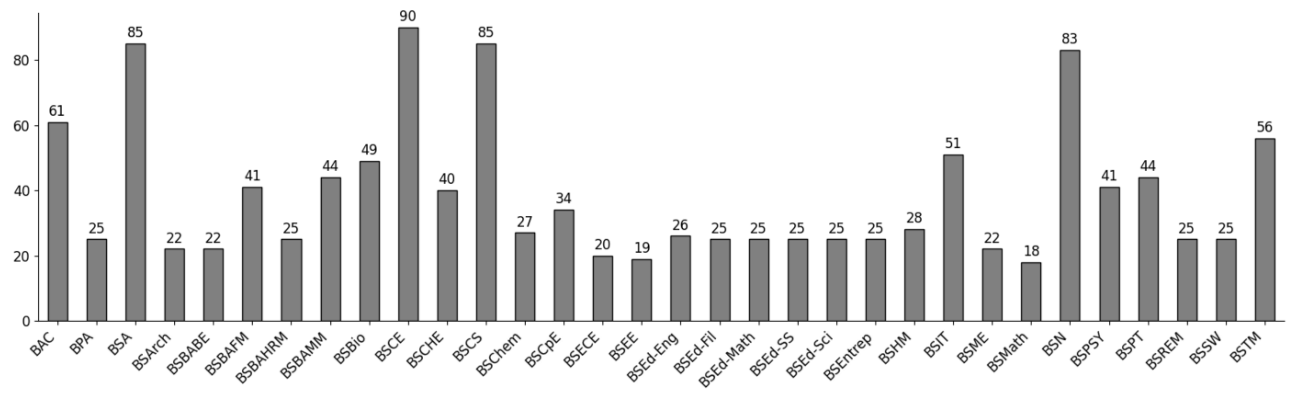


Figure 2. Sample Count and Classes of the Resampled Distribution of the Dataset

d) Data Resampling and Feature Selection

Data resampling was applied to the training dataset as it was observed that classes of the training dataset were not balanced as shown in Figure 3. Class imbalance occurs when a class of a dataset is considered a minority instance or a majority instance [18]. The minority class may be misclassified as the model will be biased towards the majority class to address the class imbalance in the dataset.

$$n_s(c_i) = \frac{n}{m} - \frac{n-m}{m(\eta_{c_i}+1)} \qquad (1)$$

Equation (1) [19] provides the calculation of the desired number of samples of each class ($n_s(c_i)$) to identify class imbalance within the dataset. The imbalance in the dataset is addressed using Synthetic Minority Oversampling Technique (SMOTE) for oversampling and Tomek Links for undersampling. This requires the total number of samples (n) of the dataset, the total number of classes (m), and a user-defined parameter (η).

The value η ranges from 0 to 1 and is directly proportional to n_s. When η is 0, there should be no more than one instance for a class to qualify as a minority class. If a class has no instances, the class does not exist. On the other hand, when η is 1, the resulting ns value is approximately 50% of the ideal number of samples ($Ideals$); $Ideals = \frac{n}{m}$.

Table 2. Resultant number of Minority Class after calculating n_s with varying η value

η	n_s Value	Number of Minority Classes
0.20	9	0
0.40	15	2
0.60	19	5
0.80	22	7
1	25	9

The minority and majority classes will be identified by (1) of the training dataset where there are 1505 samples with 32 distinct classes: n = 1505, m = 32. The number of samples in each class (n_{c_i}) is shown in Figure 3. Both n = 1505 and m = 32 values are substituted to (1) with different η values to calculate n_s. The class is said to be part of the minority if n_{c_i} is less than n_s, otherwise, it will be part of the majority class. The n_s value is shown in Table 2 and is compared with n_{c_i} values in Figure 3 to identify the minority class in the training dataset.

A dataset is considered imbalanced if at least one minority class is identified after calculating the n_s value corresponding to η. Analyzing the number of minority classes in Table 2 reveals that as the η value increases, the number of minority classes also rises, confirming the dataset's imbalance. To address this issue, SMOTE will be applied to handle the minority class, while Tomek Links will be used to manage the majority class.

SMOTE is used to oversample the minority class by replicating and generating new data points closely related to the existing sample [20]. The new samples are randomly generated by selecting one or more K-nearest neighbors for the minority classes, which effectively boosts the count of samples based on the calculated ns (number of samples). This approach particularly balances the dataset through increasing the samples within the minority class.

On the other hand, Tomek Link eliminates instances from the majority class closely related to the minority class by applying the nearest neighbor rule to a selected instance [21] instead of removing samples until it reaches the desired number of samples. Tomek Link will only partially address the class imbalance of the dataset, but the main objective is to clarify the decision boundary by eliminating borderline cases within the majority class.

To address class imbalance, the dataset shown in Figure 4 has been prepared for training the Random Forest model using a combination of SMOTE and Tomek Links. SMOTE has added synthetic samples to enhance the representation of the minority classes, while Tomek Links have partially reduced samples from the majority classes. This approach has successfully refined the class boundaries, improving the distinction between the majority and minority classes, while significantly increasing the overall number of samples in the dataset.

The dataset will undergo feature selection using Recursive Feature Elimination (RFE), a technique that fits a model and systematically eliminates the least important features based on their coefficients or importance scores [22]. RFE evaluates and ranks features by iteratively searching through subsets of the training dataset, starting with all features and progressively removing the least significant ones until the desired number of features is reached [23]. This process helps identify and retain the most important features for optimal model training.

e) Hyperparameter Tuning

Hyperparameters are configuration variables set before training an algorithm to enhance performance [24]. Tuning the hyperparameters of a model to find the best combination of parameters for the dataset improves the accuracy and efficiency of a model [12]. The hyperparameter tuning approach the study would utilize is a hybrid between RandomizedSearchCV and GridSearchCV.

The hyperparameter tuning will focus on the following key hyperparameters:

1) **n_estimator**: the number of trees in the forest. This determines the number of decision trees used by the model
2) **max_depth**: the maximum depth of each tree. This controls how deep each tree can grow
3) **min_samples_split**: the minimum number of samples required to split an internal node; and
4) **min_samples_leaf**: the minimum number of samples needed to be present at a leaf node

Table 3. Broader Parameter Space for Initial Search

Parameters	Lower Bound	Upper Bound	Steps
n_estimators	100	1000	50
max_depth	10	50	10
min_samples_split	2	10	1
min_samples_leaf	1	4	1

Table 4. Narrow Parameter Space for GridSearchCV

Parameters	Lower Bound	Upper Bound	Steps
n_estimators	-100	+100	50
max_depth	-10	+10	5
min_samples_split	-1	+2	1
min_samples_leaf	0	+2	1

Optimizing these hyperparameters enhances the model by achieving a balance between complexity and generalization. Adjusting the number of generated trees and controlling its depth, allows the model to avoid overfitting while capturing key patterns in the data. Having a threshold for splits and leaves ensures that each tree remains meaningful, improving accuracy and interpretability of the model.

RandomizedSearchCV is a hyperparameter tuning technique that optimizes the model's performance by randomly sampling from a specific range of hyperparameter values. RandomizedSearchCV will obtain optimal hyperparameters by randomly selecting the best set of parameters from the search space [10]. This is executed through randomly testing parameter combinations from a wider range.

GridSearchCV is a hyperparameter tuning method that continuously searches through a specific set of hyperparameter combinations that it finds to be the best performing one for the model. This approach is an exhaustive search technique as it checks every possible combination of hyperparameters [7]. Although it is computationally demanding, it will obtain the optimal hyperparameters by going through each combination [11].

The parameter grid in Table 3 will guide the initial hyperparameter tuning of the Random Forest model. The best hyperparameters found by RandomizedSearchCV will define a refined parameter grid in Table 4 by adding and subtracting the outlined values to serve as the lower and upper bound of the grid, which GridSearchCV will then apply to further tune the model.

f) Evaluation

The Enhanced Random Forest model will be compared to four other classification models: Traditional Random Forest, Adaptive Boosting, Gradient Boosting and Extra Trees Classifier. Adaptive Boosting (AdaBoost) is a boosting machine learning that combines weak classifiers iteratively and turning those into strong classifiers by adjusting the weights of training samples to improve the overall performance of the model [25]. Gradient boosting is a method that also combines multiple weak learners to develop the final model. It follows an order to train these models sequentially by allocating more weights on the instances with erroneous outputs [26]. Extra Trees Classifier is a variant of Random Forest that uses a random selection of a splitting value for the nodes instead of using a calculated and optimal value to split the data [27]. This algorithm aims to make the individual trees unique to make them diversified and uncorrelated which generally reduces the variance.

The models will be evaluated using cross-validation. Cross-validation evaluates the performance of a model by dividing the dataset into subsets or folds [28]. Accuracy, f1-score, and Area Under the Receiver Operating Characteristic Curve (ROC-AUC) One-vs-Rest (OVR) will be the scoring metrics of the cross-validation.

$$Cross\ Validation\ Score = \frac{1}{k}\sum_{i=1}^{k} Performance(D_i)$$

The dataset will be divided into 8-fold (k) with approximately the same size, and it will iterate (i) over each fold where subset i will be the test set, and the other k-1 subset will be the training set. The model will be trained with the training set, and the test set will be used to evaluate the model to calculate the performance of the model based on the scoring metric. The average of the scoring metric across all k iterations will serve as the cross-validation score of the model.

$$Accuracy = \frac{TP - TN}{TP + TN + FP + FN}$$

Accuracy is a fundamental performance metric in machine learning that measures the proportion of correctly predicted instances out of the total predictions made by the model. It is calculated as the ratio of true positives (TP) and true negatives (TN) to the overall number of instances. While accuracy is intuitive and easy to interpret, it works best when the dataset has a balanced class distribution. In situations involving significant class imbalance, accuracy can be misleading, as it might give an overly optimistic assessment of the model's performance. For instance, in a dataset where one class dominates, a model could achieve high accuracy simply by predicting the majority class every time, even if it fails to identify instances of the minority class.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\ Score = 2 \cdot \frac{Precision\ \cdot Recall}{Precision + Recall}$$

The F1-score is a harmonic means of precision and recall, providing a single metric that balances these two critical aspects of model performance. Precision measures the proportion of TP among all positive predictions (TP+FP), while recall assesses the proportion of TP correctly identified out of all actual positives (TP+FN). The F1-score is particularly valuable in scenarios with imbalanced datasets, as it emphasizes both false positives (FP) and false negatives (FN). A high F1-score indicates that the model performs well in identifying positive instances without disproportionately sacrificing precision or recall. Unlike accuracy, which may overlook minority class performance, the F1-score offers a more nuanced evaluation, making it a preferred choice for tasks like fraud detection, medical diagnosis, or any domain where correctly identifying minority classes is crucial.

$$FPR = \frac{FP}{FP + TN}$$

$$AUC_i = \int_0^1 Recall_i(FPR_i)d(\,FPR_i)$$

ROC-AUC is a key performance metric as it determines the model's ability to differentiate between classes. It considers the Recall against the False Positive Rate (FPR) across all classification thresholds. By evaluating the ROC-AUC, the metric provides a single value that summarizes the model's capacity to distinguish between positive and negative instances. A higher ROC-AUC indicates better performance in discriminating between classes, with a value of 1.0 representing perfect classification and 0.5 representing random guessing.

$$AUC_{OvR} = \frac{1}{K}\sum_{i=1}^{K} AUC_i$$

In the context of multi-class classification, the OVR strategy is often employed to handle multiple categories. OVR evaluates the performance of each class individually by comparing it against all other classes combined into a single category. For each class, a ROC-AUC score is calculated, which measures how well the model can separate the target class from the rest. The results are then aggregated by computing the average of these individual ROC-AUC scores to provide an overall performance metric for the multi-class model. This approach not only enables the evaluation of each class independently but also captures the model's ability to generalize across all classes, making it a robust and interpretable metric for multi-class problems.

RESULTS AND DISCUSSION

This section presents the result of the use of feature engineering, feature selection, and hyperparameter tuning and the performances of the Enhanced Random Forest model, Traditional Random Forest model, AdaBoost, Gradient Boosting, and Extra Trees Classifier.

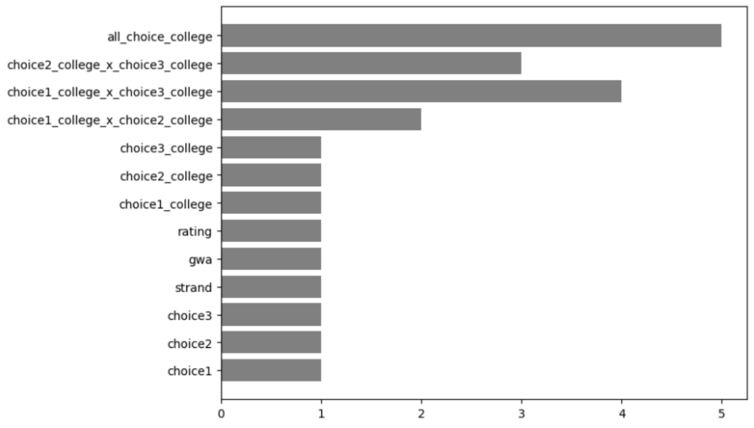


Figure 3. Feature Ranking of all features of the dataset

Figure 5 presents the ranking of all the features based on their importance in training the model including the generated features with the use of feature engineering. The features are ranked from one to five, where a feature that is ranked one indicates the most important feature, and the other ranks are considered less important. Out of the 13 features that are evaluated, the following features were retained for the model training: ***choice1, choice2, choice3, strand, gwa, rating, choice1_college, choice2_college, choice3_college***. The features were identified as most informative in the model's classification tasks.

The features that were grouped based on the interrelationship of classes for the Program Choices, ***choice1_college, choice2_college, and choice3_college***, were retained. The retention of these features indicates the model has captured the interaction between the classes of the Program Choices. The addition of these features added nuance in understanding the applicant's preferences beyond their ranks. While the interaction between the interrelationship of classes of Program Choices are considered by the model not important and was eliminated by RFE. This suggests that the interaction terms among departments does not add any substantial predictive value. This suggests that while the grouped choices have direct predictive importance, the combination of departments has no significant enhancement towards the model's predictive abilities.

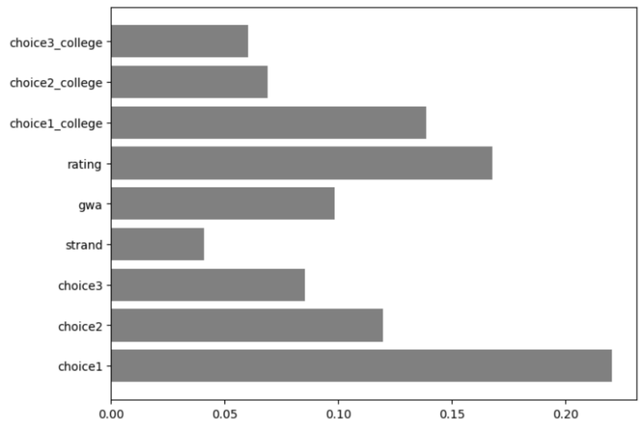


Figure 4. Feature importance of the remaining features after RFE

Figure 6 details the relative importance of each selected feature and the final features of the dataset that will be utilized by the Enhanced Random Forest in its training. Among the features, ***choice1*** was considered the most important feature within the dataset. This suggests that the first program choice of the applicant is a dominant factor in predicting the outcome, which also reflects to the model's sensitivity to the primary choice of an applicant. This suggests a high likelihood that an applicant will choose or be accepted in their first choice in the admission, thus holding the most predictive weight.

The ***rating*** feature gained the second highest importance within the dataset which suggests that applicants' standing in the admission is a strong predictor. An outcome is expected if the rating of the applicant directly influences the likelihood of acceptance within a program. ***choice1_college*** and ***choice2*** show considerable importance which suggests that the specific department associated with first choice and the second choice of the applicant has meaningful contribution. The high importance of ***choice1_college*** indicates that not only the choice itself but also the specific department tied to the choice plays a role in predicting the outcome, possibly due to the unique characteristics or strength of a certain department. On the other hand, ***strand*** and ***choice3_college*** are among the least important features. The low importance of strand suggests that while the academic strand of the applicant can be a factor, it does not heavily influence the outcome compared to other variables. This indicates that the predictive model is more influenced by the choice and ratings than by academic background alone, or this can also suggest that strand is not discriminatory.

Table 5. Returned Hyperparameters based on the Tuning Approach

	Default Parameters	**Randomized SearchCV only**	**GridSearch CV only**	**Hybrid Approach**
n_estimators	100	850	800	800
max_depth	None	20	30	30
min_samples_split	2	3	3	3
min_samples_leaf	1	1	1	1
Time (minutes)		1.42	31.47	5.81

In tuning the hyperparameters for the Enhanced Random Forest, three approaches, RandomizedSearchCV only, GridSearchCV only, and the hybrid approach, were utilized to determine the optimal hyperparameters of the model. Table 5 shows the returned hyperparameters of the different approaches and the default parameter of the model. Default parameters can provide a quick benchmark, but this does not tailor the model's complexity and learning capabilities to a specific dataset.

Using RandomizedSearchCV, it has explored a broad hyperparameter space based on the parameter grid presented in Table 2. It has performed 5-fold cross-validation for each of the 150 randomly selected parameter combinations, totaling 750 fits. The RandomizedSearchCV tuning took approximately 1.42 minutes to finish searching for the optimal parameter. It has effectively sampled from a wide parameter space and swiftly concluded to a potential and near-optimal parameter, while using a lower computational cost.

GridSearchCV has exhaustively searched over the parameter grid defined in Table 3 which would involve testing every possible combination of parameters leading to 3420 total combinations and performing a 5-fold cross-validation on each, resulting in 17,100 fits. The tuning approach took significantly longer with an approximate time of 31.47 minutes. Although GridSearchCV is computationally expensive, it has ensured a comprehensive exploration of the parameter space, which increases the likelihood of finding the true optimal hyperparameter set for the model.

The hybrid approach that the Enhanced Random Forest utilizes aims to combine the speed of RandomizedSearchCV with the thoroughness of GridSearchCV. This two-step process allows RandomizedSearchCV to identify a promising region of the hyperparameter space, using the same parameter grid in Table 3 that will take approximately 1.42 minutes. The best parameters found by RandomizedSearchCV will be the starting point of the parameter grid for the subsequent GridSearchCV. With the ranges shown in Table 4, GridSearchCV will perform a more focused search, conducting 1,500 fits over 300 combinations with 5-fold cross-validation. This step took approximately 4.39 minutes to return an optimal parameter. The entire hybrid process approximately took 5.81 minutes. Notably, the hybrid approach has returned the same parameter as the GridSearchCV that has searched the entire parameter space, but it has significantly reduced the computational cost for searching the optimal parameter.

The generated features and the remaining important features of the dataset will be utilized by the Enhanced Random Forest model along with the tuned hyperparameters produced by the hybrid approach of RandomizedSearchCV and GridSearchCV. The performance of the Enhanced Random Forest model is compared with the Traditional Random Forest, AdaBoost, Gradient Boosting, and Extra Trees Classifier are evaluated using different scoring metrics. Each metric was assessed using 8-fold cross-validation, providing the mean scores and their respective standard deviation.

Table 6. Comparative analysis of different classification models

	Accuracy	**F1-Score**	**ROC-AUC**
Enhanced Random Forest	0.7510 ± 0.0226	0.7423 ± 0.0219	0.9657 ± 0.0068
Traditional Random Forest	0.6468 ± 0.0246	0.6303 ± 0.0284	0.9485 ± 0.0091
AdaBoost	0.6952 ± 0.0468	0.6866 ± 0.0452	0.9532 ± 0.0093
Gradient Boosting	0.6638 ± 0.0387	0.6523 ± 0.0399	0.9100 ± 0.0152
Extra Trees Classifier	0.6839 ± 0.0377	0.6693 ± 0.0372	0.9574 ± 0.0103

The Enhanced Random Forest model showed a significant increase in performance in comparison to the other models across the different performance metrics as shown in Table 6. The model has attained an accuracy of 0.7510 ± 0.0226 and an F1 Score of 0.7423 ± 0.0219, surpassing the Traditional Random Forest model, which gained 0.6468 ± 0.0246 for its accuracy and 0.6303 ± 0.0284 for its F1 Score. It also outperformed AdaBoost (0.6952 ± 0.0468 and 0.6866 ± 0.0452), Gradient Boosting (0.6638 ± 0.0387 and 0.6523 ± 0.0399), and Extra Trees Classifier (0.6839 ± 0.0377 and 0.6693 ± 0.0372), which suggests that the feature engineering, feature selection, and tuning were effective and has increased the general classification accuracy while maintaining a balance between precision and recall.

In terms of ROC-AUC, the Enhanced Random Forest has attained the highest score at 0.9657 ± 0.0068 compared to the Traditional Random Forest (0.9485 ± 0.0091), AdaBoost (0.9532 ± 0.0093), Gradient Boosting (0.9100 ± 0.0152), and Extra Trees (0.9574 ± 0.0103). Although the Extra Trees Classifier and AdaBoost produced great ROC-AUC scores, their relatively lower Accuracy and F1 Scores indicate that while they can rank instances effectively, it can be prone to misclassification errors in terms of its overall performance and class balance.

The relatively lower standard deviation of the Enhanced Random Forest model underscores its robust and consistent performance across multiple folds. This suggests that the hybrid hyperparameter tuning approach, along with careful feature engineering and selection, significantly increased the model's capability to capture underlying patterns in the dataset.

To fully utilize the Enhanced Random Forest model, the PLMAT Course Recommendation web application was developed to streamline the decision-making process for the possible program that would fit the student applicants' academic profiles and preferences. By applying the Enhanced algorithm, this web application will use that model to create predictions for program recommendations on a new set of admission applicants. This will be beneficial for the PLM admission office as it can facilitate the potential program allocations for waitlisted applicants while eliminating some manual processing involved in the decision-making process of assigning programs. This web application was developed using the Python programming language with the use of Streamlit, a Python framework specifically used for Artificial Intelligence or Machine Learning projects with functionalities that deliver user-friendly and interactive design. By leveraging these open-source technologies, the application would not only efficiently organize the admission workflow but would also enhance the overall experience for its users. With its simplicity and straightforwardness, its users would easily navigate and access the application without needing a certain level of technical expertise.

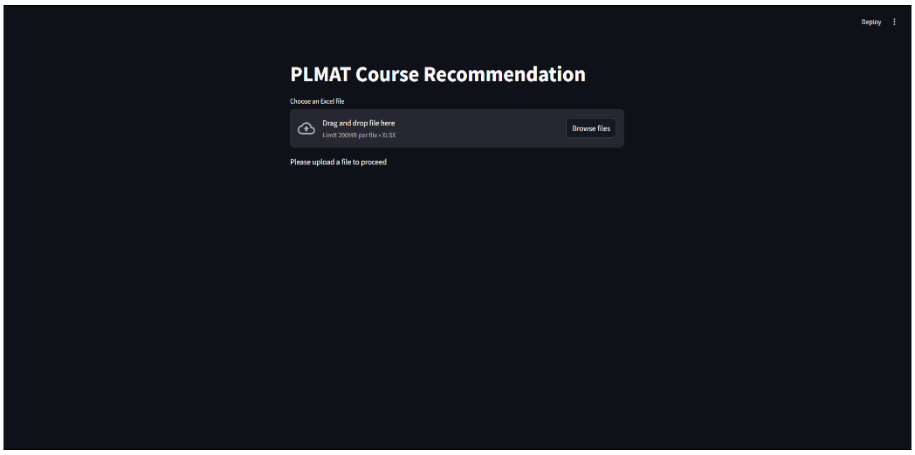
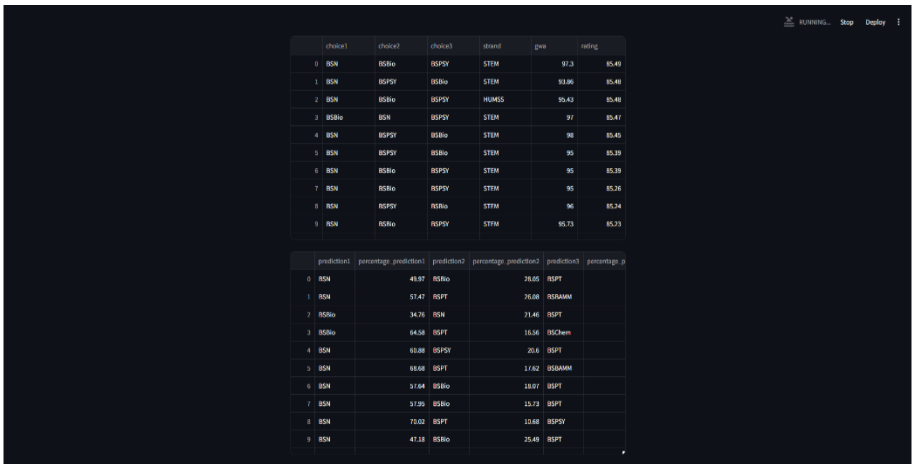

Deploy
PLMAT Course Recommendation
Choose an Excel file
Drag and drop file here
Limit 200MB per file • XLSX
Browse files
Please upload a file to proceed

Figure 7. PLMAT Course Recommendation Main Interface

Figure 7 shows the initial interface of this web application provides a straightforward design for users to upload their datasets. They will be prompted to upload their data to the uploader through an Excel file (xlsx) with a file size limit of 200 MB. The uploading of files can either be through drag and drop or browsing of files within the system. This user-friendly design allows the upload feature to be efficient and accessible, which serves as the initial step for the program recommendation process.

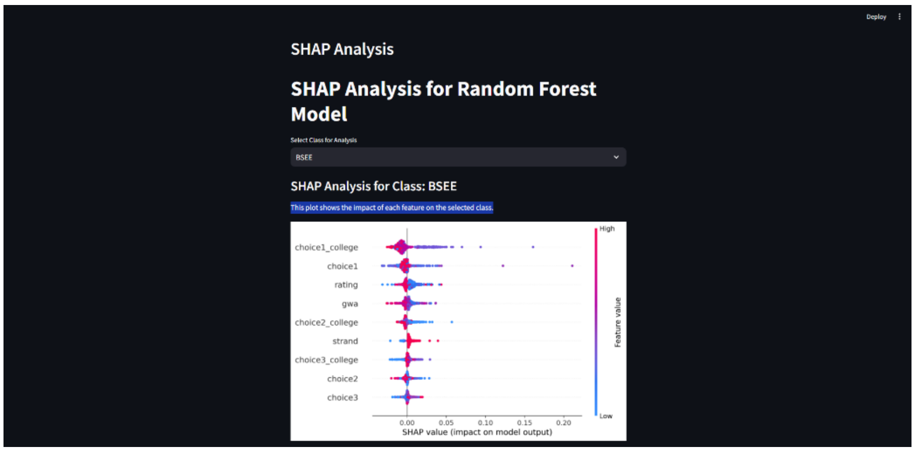


	choice1	choice2	choice3	strand	gwa	rating
0	BSN	BSBio	BSPSY	STEM	97.3	85.49
1	BSN	BSPSY	BSBio	STEM	93.86	85.48
2	BSN	BSBio	BSPSY	HUMSS	95.43	85.48
3	BSBio	BSN	BSPSY	STEM	97	85.47
4	BSN	BSPSY	BSBio	STEM	98	85.45
5	BSN	BSPSY	BSBio	STEM	95	85.39
6	BSN	BSBio	BSPSY	STEM	95	85.39
7	BSN	BSBio	BSPSY	STEM	95	85.26
8	BSN	BSPSY	BSBio	STEM	96	85.24
9	BSN	BSBio	BSPSY	STEM	95.73	85.23

	prediction1	percentage_prediction1	prediction2	percentage_prediction2	prediction3	percentage_p
0	BSN	49.97	BSBio	28.05	BSPT	
1	BSN	57.47	BSPT	26.08	BSBAMM	
2	BSBio	34.76	BSN	21.46	BSPT	
3	BSBio	64.58	BSPT	16.56	BSChem	
4	BSN	60.88	BSPSY	20.6	BSPT	
5	BSN	68.68	BSPT	17.62	BSBAMM	
6	BSN	57.64	BSBio	18.07	BSPT	
7	BSN	57.95	BSBio	15.73	BSPT	
8	BSN	70.02	BSPT	10.68	BSPSY	
9	BSN	47.18	BSBio	25.49	BSPT	

Figure 8. Data Preview and Predictions of Random Forest Model with Corresponding Vote Percentage

After the uploading of the Excel file, the model will automatically analyze the data. As shown in Figure 8, the application will then display the uploaded dataset in a tabular format for a quick overview of the input data. Underneath the input, the output, which displays the corresponding predictions of each sample data, will be displayed in a tabular form as well. It will show the first top five programs recommended by the model along with the percentage of the prediction. This corresponds to the proportion of the trees that voted for that class. The inclusion of the download button is to combine the input and the output into a single Excel file for ease of processing.



SHAP Analysis

SHAP Analysis for Random Forest Model

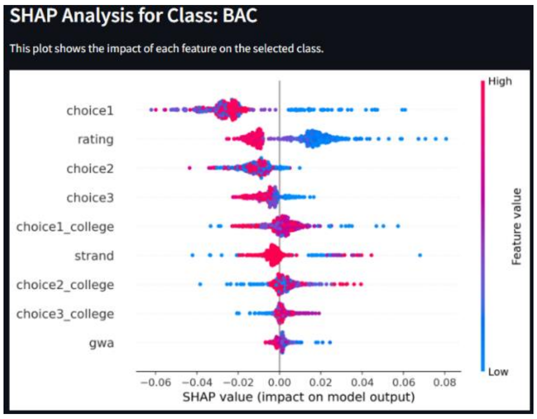
Select Class for Analysis: BSEE

SHAP Analysis for Class: BSEE

This plot shows the impact of each feature on the selected class.

Figure 9. SHAP Interface of PLMAT Course Recommendation Website

As displayed in Figure 9, this section of the web application provides a comprehensive SHAP (Shapley Additive Explanations) analysis for the Random Forest model. It presents two visualizations that illustrate the impact of each feature on a selected class, which are selected through a drop-down menu.



SHAP Analysis for Class: BAC

This plot shows the impact of each feature on the selected class.

Figure 10. SHAP Dot Plot Visualization for a Selected Class

The first summary dot plot as shown in Figure 10 demonstrates the impact of each feature to the prediction for the selected class. Basically, each dot in the visualization represents a single prediction, with a color indicating the feature value (e.g., red for high feature value and blue for low feature value). The wide spread of the dots across the x-axis means that the feature has varying impacts on various samples. Furthermore, the clustering of dots in a certain spot indicates the relative effect of that sample to the predictions. Clustered dots on the positive side of the x-axis means that lower or higher feature value drives higher predictions for that sample. Contrary to this, clustered dots on the negative side suggest a lower prediction for the sample.

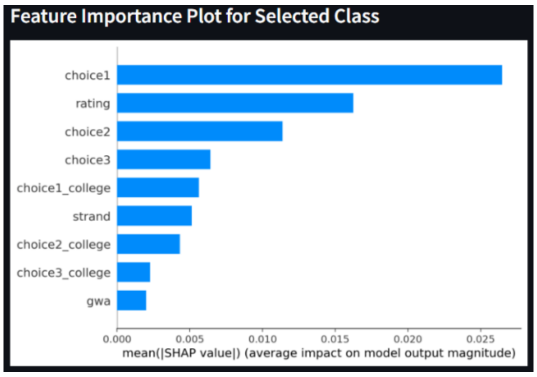


Figure 11. SHAP Analysis Bar Plot Visualization

As presented in Figure 11, the next bar plot is a more simplified visualization that shows the importance for the selected class. It shows an ordered plot of the features that have the highest impact on the model's predictions. It uses the mean absolute SHAP values for each feature which represents the average contribution of that feature across all samples. The length of each bar is proportional to the significance of the feature that influences the model for its prediction. This means that the larger values indicate higher importance whereas lower values mean less importance as a determining factor for the target class.

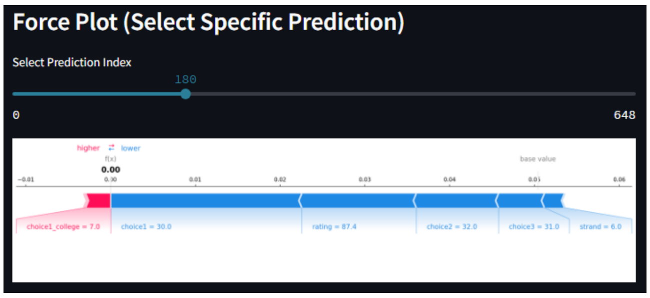


Figure 12. SHAP Force Plot Visualization

Lastly, Figure 12 allows the user to select a specific prediction index using a slider. It will show how individual features influence a specific prediction for the selected class. The base value represents the average model prediction which is typically the value that the model would predict if no features were provided. The colored arrows represent how each feature contributes to moving the prediction from the base value. The width of each arrow indicates the extent of the feature's effect on the prediction. Red arrows signify positive contribution which means it moves the prediction higher, while blue arrows emphasize negative contribution, or it moves the prediction lower.

CONCLUSION

This study highlights the enhancement of the Random Forest algorithm for program recommendation of waitlisted applicants through feature engineering, feature selection, and hybrid hyperparameter tuning. The findings have shown that the application of feature engineering to the dataset, particularly the corresponding college departments associated with the program choices of the applicants, significantly improved the model's classification capabilities. Although certain interacting features were influential to the model's performance, other features were deemed irrelevant. Hence, the application of recursive feature elimination (RFE) was a vital step to reduce the noise within the dataset. This iterative process of eliminating the lowest predicting factors among the features allowed much more reliable predictions for the model as it utilized the strong indicators in the instances rather than the features that had lower importance. Furthermore, the application of RandomizedSearchCV and GridSearchCV to optimize the hyperparameter settings of the Random Forest Model has reduced the possibility of overfitting and improved the classification accuracy of the model. Using RandomizedSearchCV for the initial set of hyperparameters allowed a much smaller search space for the optimal settings. Then, GridSearchCV further refines the best-performing parameters from the already narrow search space. This allowed the model to attain the optimal set of hyperparameters without compromising the computational costs.

In comparison with the other models, the enhanced Random Forest algorithm outperformed the other models by huge margins across all metrics. These results feature the model's robustness and its capabilities in balancing precision and recall while accurately predicting the classes of each instance. In conclusion, this study established a significant improvement in the predictive performance of the Random Forest algorithm in relation to program recommendations. This emphasizes the effectiveness of the methodology employed in this study, which can represent an optimized solution for machine learning models in similar areas, contributing to a more effective and efficient decision-making strategy for the education field.

Acknowledgements:

We thank John Vincent Guanco, Gian Karlo Ramos, and Erwin D. Marcelo for validating the datasets, training and testing the model and architecture of the study. Special thanks to Vivien A. Agustin for their assistance and understanding throughout the study.

REFERENCES

[1] J. F. Luesia, I. Benítez, R. Company-Córdoba, I. Gómez-Gómez, and M. Sánchez-Martín, “Assessing the relevance of academic competencies in college admission tests from a higher-order thinking perspective: A systematic review,” Think. Ski. Creat., vol. 48, p. 101251, Jun. 2023, doi: 10.1016/j.tsc.2023.101251.

[2] I. I. Yousafzai and B. Jamil, “Relationship between admission criteria and academic performance: A correlational study in nursing students,” Pak. J. Med. Sci., vol. 35, no. 3, Art. no. 3, May 2019, doi: 10.12669/pjms.35.3.217.

[3] “Reliability and Validity of the College Admission Test for K-12 Graduates,” International Journal of Education, Learning and Development (IJELD). Accessed: Nov. 19, 2024. [Online]. Available: https://eajournals.org/ijeld/vol11-issue-1-2023/reliability-and-validity-of-the-college-admission-test-for-k-12-graduates/

[4] “Comparative analysis of machine learning models for breast cancer prediction and diagnosis: a dual-dataset approach | Awan | Indonesian Journal of Electrical Engineering and Computer Science.” Accessed: Nov. 19, 2024. [Online]. Available: https://ijeecs.iaescore.com/index.php/IJEECS/article/view/36388

[5] Y. Zhao, Y. Huang, Z. Wang, and X. Liu, “A new feature selection method based on importance measures for crude oil return forecasting,” Neurocomputing, vol. 581, p. 127470, May 2024, doi: 10.1016/j.neucom.2024.127470.

[6] M. Imani and H. R. Arabnia, “Hyperparameter Optimization and Combined Data Sampling Techniques in Machine Learning for Customer Churn Prediction: A Comparative Analysis,” Technologies, vol. 11, no. 6, Art. no. 6, Dec. 2023, doi: 10.3390/technologies11060167.

[7] M. Ogunsanya, J. Isichei, and S. Desai, “Grid search hyperparameter tuning in additive manufacturing processes,” Manuf. Lett., vol. 35, pp. 1031–1042, Aug. 2023, doi: 10.1016/j.mfglet.2023.08.056.

[8] W. Zhao and P. Li, “Blockwise Feature Interaction in Recommendation Systems,” Jun. 28, 2023, arXiv: arXiv:2306.15881. doi: 10.48550/arXiv.2306.15881.

[9] I. Urbaniak and P. Biskup, “Recovering MRI magnetic field strength properties using machine learning based on image compression quality scores,” Optim. Eng., Oct. 2024, doi: 10.1007/s11081-024-09928-x.

[10] L. Zahedi, F. G. Mohammadi, S. Rezapour, M. W. Ohland, and M. H. Amini, “Search Algorithms for Automated Hyper-Parameter Tuning,” Apr. 29, 2021, arXiv: arXiv:2104.14677. doi: 10.48550/arXiv.2104.14677.

[11] T. Thanh Ngoc, L. Van Dai, and C. Minh Thuyen, “Support Vector Regression based on Grid Search method of Hyperparameters for Load Forecasting,” Acta Polytech. Hung., vol. 18, no. 2, pp. 143–158, 2021, doi: 10.12700/APH.18.2.2021.2.8.

[12] L. Wang, L. Zhao, X. Liu, J. Fu, and A. Zhang, “SepPCNET: Deeping Learning on a 3D Surface Electrostatic Potential Point Cloud for Enhanced Toxicity Classification and Its Application to Suspected Environmental Estrogens,” Environ. Sci. Technol., vol. 55, no. 14, pp. 9958–9967, Jul. 2021, doi: 10.1021/acs.est.1c01228.

[13] L. Breiman, “Random Forests,” Mach. Learn., vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[14] Sruthi, “Understanding Random Forest Algorithm With Examples,” Analytics Vidhya. Accessed: Nov. 19, 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[15] A. F. A. H. Alnuaimi and T. H. K. Albaldawi, “An overview of machine learning classification techniques,” BIO Web Conf., vol. 97, p. 00133, 2024, doi: 10.1051/bioconf/20249700133.

[16] S. Misra, O. Osogba, and M. Powers, “Chapter 1 - Unsupervised outlier detection techniques for well logs and geophysical data,” in Machine Learning for Subsurface Characterization, S. Misra, H. Li, and J. He, Eds., Gulf Professional Publishing, 2020, pp. 1–37. doi: 10.1016/B978-0-12-817736-5.00001-6.

[17] “Data Mining: The Textbook | SpringerLink.” Accessed: Nov. 19, 2024. [Online]. Available: https://link.springer.com/book/10.1007/978-3-319-14142-8

[18] E. F. Swana, W. Doorsamy, and P. Bokoro, “Tomek Link and SMOTE Approaches for Machine Fault Classification with an Imbalanced Dataset,” Sensors, vol. 22, no. 9, Art. no. 9, Jan. 2022, doi: 10.3390/s22093246.

[19] R. Kavya, J. Christopher, S. Panda, and Y. B. Lazarus, “Machine Learning and XAI approaches for Allergy Diagnosis,” Biomed. Signal Process. Control, vol. 69, p. 102681, Aug. 2021, doi: 10.1016/j.bspc.2021.102681.

[20] R. Efendi, T. Wahyono, and I. Widiasari, LSTM SMOTE: An Effective Strategies for DDoS Detection in Imbalanced Network Environments. 2024. doi: 10.20944/preprints202407.1825.v1.

[21] S. Sawangarreerak and P. Thanathamathee, “Random Forest with Sampling Techniques for Handling Imbalanced Prediction of University Student Depression,” Information, vol. 11, no. 11, Art. no. 11, Nov. 2020, doi: 10.3390/info11110519.

[22] A. M. Priyatno and T. Widiyaningtyas, “A SYSTEMATIC LITERATURE REVIEW: RECURSIVE FEATURE ELIMINATION ALGORITHMS,” JITK J. Ilmu Pengetah. Dan Teknol. Komput., vol. 9, no. 2, Art. no. 2, Feb. 2024, doi: 10.33480/jitk.v9i2.5015.

[23] Y. Yin et al., “IGRF-RFE: a hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset,” J. Big Data, vol. 10, no. 1, p. 15, Feb. 2023, doi: 10.1186/s40537-023-00694-8.

[24] H. Yuliana et al., “Hyperparameter Optimization of Random Forest Algorithm to Enhance Performance Metric Evaluation of 5G Coverage Prediction | Buletin Pos dan Telekomunikasi.” Accessed: Dec. 02, 2024. [Online]. Available: https://bpostel.kominfo.go.id/index.php/bpostel/article/view/390

[25] A. de Giorgio, G. Cola, and L. Wang, “Systematic review of class imbalance problems in manufacturing,” J. Manuf. Syst., vol. 71, pp. 620–644, Dec. 2023, doi: 10.1016/j.jmsy.2023.10.014.

[26] A. A. Khan, O. Chaudhari, and R. Chandra, “A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation,” Expert Syst. Appl., vol. 244, p. 122778, Jun. 2024, doi: 10.1016/j.eswa.2023.122778.

[27] R. Shafique, A. Mehmood, S. ullah, and G. S. Choi, “Cardiovascular Disease Prediction System Using Extra Trees Classifier,” Sep. 16, 2019, Research Square. doi: 10.21203/rs.2.14454/v1.

[28] D. Wilimitis and C. G. Walsh, “Practical Considerations and Applied Examples of Cross-Validation for Model Development and Evaluation in Health Care: Tutorial,” JMIR AI, vol. 2, p. e49023, Dec. 2023, doi: 10.2196/49023.