Predicting the Titanic Spaceship's Passengers with Ensemble Machine Learning Methods

Izaaz Rahman Akbar
(21/472855/PA/20348)
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
izaaz.rahman.akbar@mail.ugm.ac.id

Matthew Tan
(21/478240/PA/20736)
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
matthew.tan@mail.ugm.ac.id

Rabbani Nur Kumoro
(21/472599/PA/20310)
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
rabbani.nur.kumoro@mail.ugm.ac.id

***Abstract* —The aim of this project is to apply machine learning techniques and feature engineering methods to accurately predict the passenger's transportation status in the popular "Titanic" Kaggle competition. The dataset provided by Kaggle consists of a training set containing 8,693 instances with 14 columns, and a testing set containing 4,277 instances with 13 columns. To handle missing values in the dataset, we employed the Simple Imputer library to fill in the missing values. Additionally, we conducted exploratory data analysis to gain insights into the feature engineering. The features that are engineered will greatly contribute to the modelling process. For our predictive models, we utilized the ensemble methods such as Random Forest and XGBoost algorithms. Comparing their performance, the XGBoost algorithm achieved an accuracy score of 80.8%, while the random forest algorithm achieved a slightly lower score of 80.5%. These results highlight the effectiveness of our ensemble approach in solving the problem at hand.**

Keywords — Competition, Ensemble, Kaggle, Machine Learning, Random Forest, XGBoost

I. INTRODUCTION

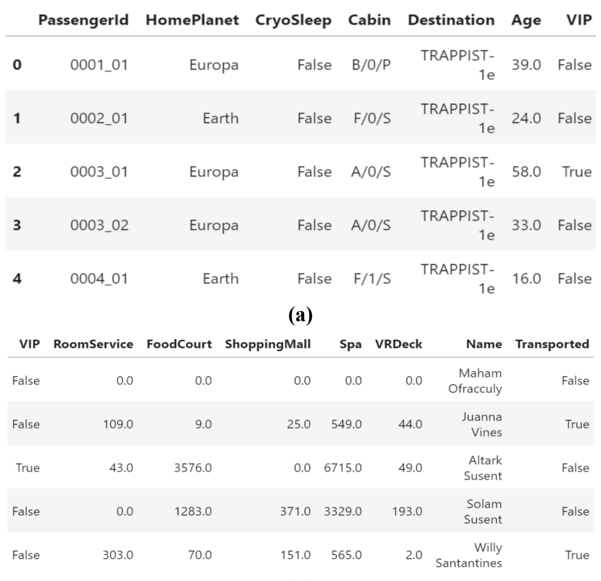
The provided Kaggle competition embarked on a futuristic journey into 2912, where the interstellar Spaceship Titanic encountered a catastrophic event **[1]**. Within a concealed dust cloud, the spacecraft has collided with a spacetime anomaly, mirroring the unfortunate destiny of its predecessor, the Olympic-class ocean liner ship, Titanic. However, unlike its Titanic, the spaceship remains physically intact although a significant predicament arises as nearly half of the passengers have been mysteriously transported to an alternate dimension. To help the rescue efforts to retrieve the lost passengers, our task is to leverage the data salvaged from the spaceship's compromised computer system. By utilizing the power of advanced algorithms and machine learning techniques, we attempt to solve this challenge through the data provided in search of patterns, correlations, and hidden insights that can shed light on the fate of these lost souls **[2]**.

Machine learning, a subfield of artificial intelligence, has empowered us to develop robust models capable of learning from and making predictions based on data. Machine-learning techniques allow us to understand the dataset's intrinsic structure, decipher the complex relationships between features and construct a predictive model that discerns the passengers who may have slipped into a parallel dimension **[3]**.

In this project, our primary objective is to maximize the power of machine learning techniques and advanced feature engineering methods to compete in the competition by providing the best prediction on whether the passenger has been transported or not. To achieve this, we attempt to conduct a comprehensive data analysis and examine the impact of each feature. Consequently, not only do we introduce new features to provide additional insights, but we also remove several existing features that may impede accurate predictions. By applying these measures, we aim to optimize the performance of our machine-learning models and uncover valuable patterns within the data.

II. DATASET

This spaceship titanic dataset involves predicting whether a passenger was transported to another dimension based on various features. The dataset can be obtained from Kaggle. In general, it consists of a training dataset with 8,693 instances and 14 columns, and a testing dataset with 4,277 instances and 13 columns. The columns in the training dataset are as described below in **Figure 1**.



	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP
0	0001_01	Europa	False	B/0/P	TRAPPIST-1e	39.0	False
1	0002_01	Earth	False	F/0/S	TRAPPIST-1e	24.0	False
2	0003_01	Europa	False	A/0/S	TRAPPIST-1e	58.0	True
3	0003_02	Europa	False	A/0/S	TRAPPIST-1e	33.0	False
4	0004_01	Earth	False	F/1/S	TRAPPIST-1e	16.0	False

(a)

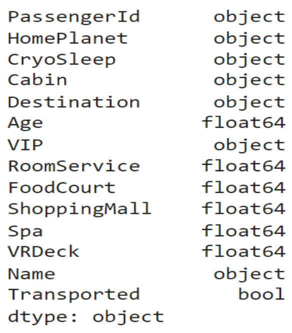
VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Name	Transported
False	0.0	0.0	0.0	0.0	0.0	Maham Ofracculy	False
False	109.0	9.0	25.0	549.0	44.0	Juanna Vines	True
True	43.0	3576.0	0.0	6715.0	49.0	Altark Susent	False
False	0.0	1283.0	371.0	3329.0	193.0	Solam Susent	False
False	303.0	70.0	151.0	565.0	2.0	Willy Santantines	True

(b)

Figure 1. (a) (b) Sample Training Dataset

The **Figure 2** below provides information about the training data type. Overall, we can observe that the

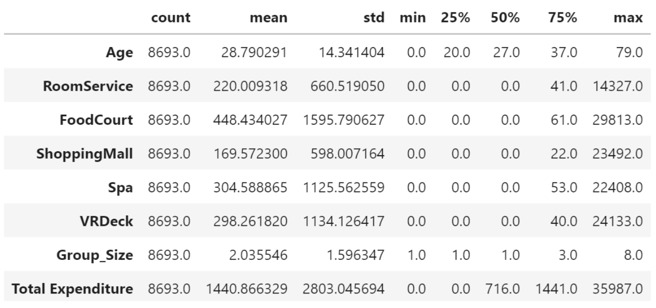
PassengerID, HomePlanet, CryoSleep, Cabin, Destination, VIP, and Name are object data types. On the other side, Age, RoomService, FoodCourt, ShoppingMall, Spa, and VRDeck are float64 data types. The transported feature is the labelled boolean which we will need to predict.



```
PassengerId      object
HomePlanet       object
CryoSleep        object
Cabin            object
Destination      object
Age             float64
VIP              object
RoomService     float64
FoodCourt       float64
ShoppingMall    float64
Spa             float64
VRDeck          float64
Name             object
Transported        bool
dtype: object
```

Figure 2. Train Data Type

From **Figure 3** Below, it was observed that RoomService, FoodCourt, ShoppingMall, Spa, and VRDeck features had over 50% of data equal to 0 and exhibited right-skewed distributions. These findings suggest the presence of significant outliers. To address this, a log transformation will be applied in a more detailed manner in the section below to normalize the data and mitigate the impact of outliers.



	count	mean	std	min	25%	50%	75%	max
Age	8693.0	28.790291	14.341404	0.0	20.0	27.0	37.0	79.0
RoomService	8693.0	220.009318	660.519050	0.0	0.0	0.0	41.0	14327.0
FoodCourt	8693.0	448.434027	1595.790627	0.0	0.0	0.0	61.0	29813.0
ShoppingMall	8693.0	169.572300	598.007164	0.0	0.0	0.0	22.0	23492.0
Spa	8693.0	304.588865	1125.562559	0.0	0.0	0.0	53.0	22408.0
VRDeck	8693.0	298.261820	1134.126417	0.0	0.0	0.0	40.0	24133.0
Group_Size	8693.0	2.035546	1.596347	1.0	1.0	1.0	3.0	8.0
Total Expenditure	8693.0	1440.866329	2803.045694	0.0	0.0	716.0	1441.0	35987.0

Figure 3. Statistical Information of Numerical Features

III. Methodology

The methodology presented in this paper outlines the process and techniques employed to develop an effective machine learning model for addressing the problem at hand. This section describes the various stages involved, including data collection, pre-processing, feature engineering, model selection, and evaluation. The methodology serves as a guide for the implementation of the proposed machine learning solution and provides insights into the decisions made throughout the experiment.

The **Figure 4** below represents the flowchart of our methodology to solve the problem. We first start with obtaining the dataset from Kaggle and split the dataset into train and test dataset. Next, we perform data pre-processing to the train dataset which consists of filling missing values and dropping certain features with high cardinality as it was deemed unnecessary.

The following step is EDA, where the pre-processed data is analyzed to gain insights into its characteristics. Statistical methods and visualization are utilized to understand the distribution, relationships, and patterns within the data. EDA helps in identifying key features, understanding data quality, and uncovering potential challenges.

After that, we perform feature engineering on the dataset where we obtain new features from the given features to be able to extract relevant information and represent it in a format that the machine learning algorithm can effectively utilize.

Next, we build our machine learning model. We chose two different models to compare to solve our problem, which are Random Forest and Extreme Gradient Boosting. This is where we feed our processed training dataset into. With machine learning modelling, comes hyperparameter tuning, where we perform hyperparameter tuning to obtain the best configuration for our machine learning model.

Lastly, we use the trained model to be tested on the test dataset and obtain the result. We perform performance calculations to see how well our model is in solving the problem. Performance metrics such as accuracy, precision, recall, and F1 score are used to calculate the performance. With this, we can evaluate the process to be able to improve our system for better performance and result.

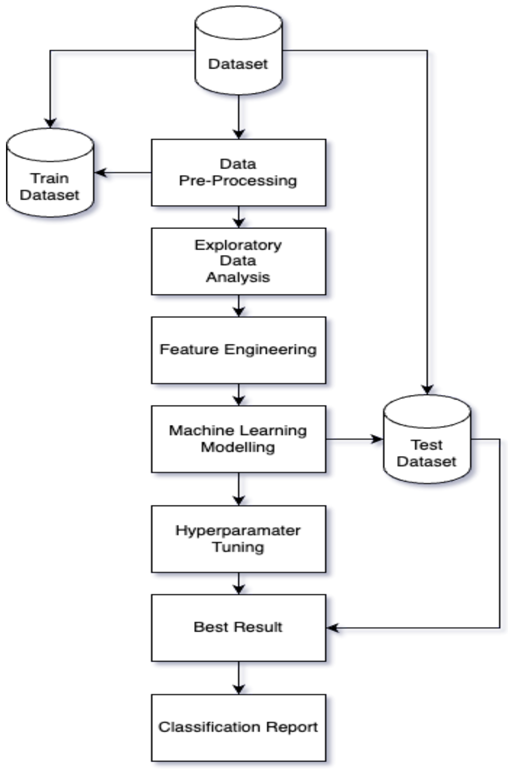
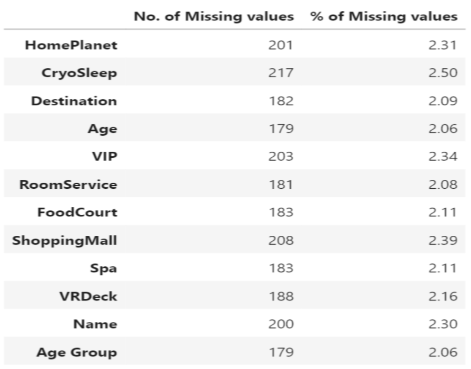


Figure 4. Flowchart Process

A. Data Pre-Processing

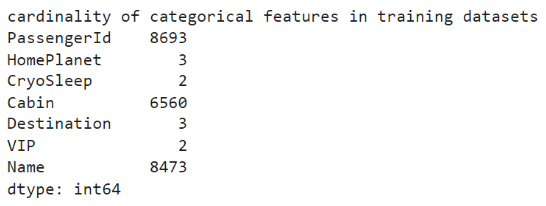
In order to handle missing values in the dataset presented on **Figure 5,** we employed the Simple Imputer library, which enabled us to fill in the missing values. Prior to the next steps that will be addressed further in the passage below, it was determined that certain features exhibited high cardinality and could be dropped from further analysis. As a result, the decision was made to remove the PassengerId, Cabin, Name, Group, and Surname features from the dataset, streamlining the data preparation phase for subsequent modelling tasks.



	No. of Missing values	% of Missing values
HomePlanet	201	2.31
CryoSleep	217	2.50
Destination	182	2.09
Age	179	2.06
VIP	203	2.34
RoomService	181	2.08
FoodCourt	183	2.11
ShoppingMall	208	2.39
Spa	183	2.11
VRDeck	188	2.16
Name	200	2.30
Age Group	179	2.06

Figure 5. Missing Values in Dataset

Figure 6. below reveals that the features PassengerId, Cabin, and Name exhibit high cardinality, with 8,693, 6,560, and 8,473 instances in the training dataset. Typically, features with high cardinality are dropped. However, in this project, we will employ Feature Engineering techniques to derive new features from these variables. This approach is adopted to leverage the abundance of data, as larger datasets tend to yield improved predictive performance from machine learning models. After the feature engineering process, we will drop features which have high cardinalities such as passengerId, Cabin, and Name.



```
cardinality of categorical features in training datasets
PassengerId     8693
HomePlanet         3
CryoSleep          2
Cabin           6560
Destination        3
VIP                2
Name            8473
dtype: int64
```

Figure 6. Checking Cardinality of Categorical Features

B. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a vital step in data analysis, enabling us to gain a comprehensive understanding of the dataset and uncover valuable insights. EDA techniques, such as bar graphs, line graphs, histograms, and pie charts, are employed to visually explore the data. In the passage below, we will delve into specific EDA analyses that will provide us with essential insights to guide feature engineering and modelling. Through these analyses, we aim to extract meaningful patterns and relationships within the dataset, which will ultimately contribute to the success of our feature engineering and modelling process.

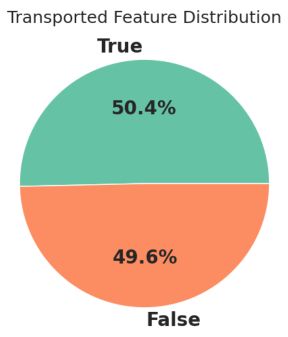


Figure 7. Transported Feature Distribution

The pie chart from **Figure 7** shows that the Transported feature within our dataset demonstrates a high level of balance between the two classes with a true value of 50.4% and a false of 49.6%. Consequently, there is no reason to employ techniques such as under-sampling or over-sampling to address class imbalance as the data is already balanced.

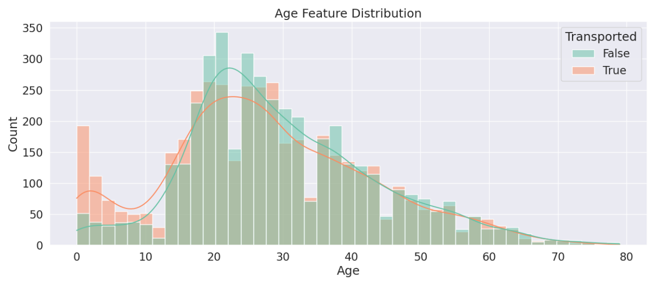


Figure 8. Age Feature Distribution

The bar and line graph from **Figure 8** provide key insights about the age feature. First, it is notable that a significant portion of the passengers fell within the age range of 18 to 32 years. Moreover, within the age bracket of 0 to 18 years, passengers were found to have a higher likelihood of being transported, especially newborns. On the other hand, passengers between the ages of 18 to 32 years showed a comparatively lower tendency to be transported when compared to those who were not transported. Interestingly, for passengers above the age of 32, there appeared to be a relatively balanced distribution between those who were transported and those who were not.

Furthermore, we create a new feature called "Age-Category. by categorizing the ages into different groups or ranges. This can better capture the underlying patterns and associations between age and the likelihood of being transported. We believe that this feature will enable us to incorporate the age information in a more meaningful way during our subsequent modeling and analysis phases.

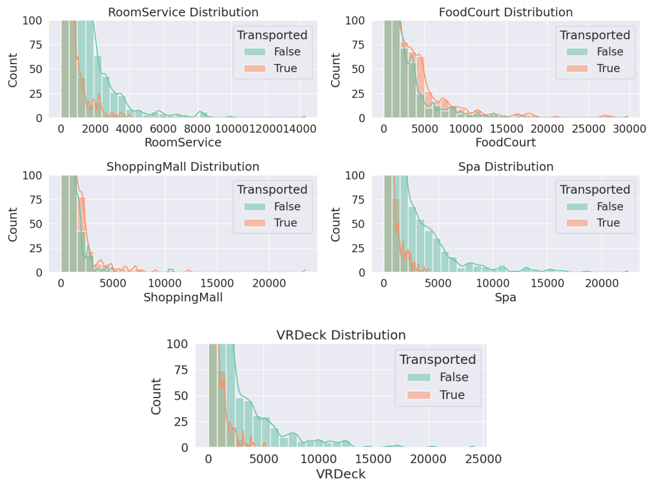


Figure 9. Expenditure Distribution

The bar graph of the five expenditure data from **Figure 9** has revealed that a significant portion of the passengers did not seem to incur any expenses. Hence, the instances with higher expenditure values can be considered outliers in our dataset, as they deviate from the predominant trend of minimal spending. Additionally, it is observed that the distribution of expenses in RoomService, Spa, and VRDeck amenities exhibit similarities, while FoodCourt and ShoppingMall expenditures also display comparable distributions. Also, all the expenditure features demonstrate a right-skewed distribution, indicating that a majority of passengers have low or no expenses.

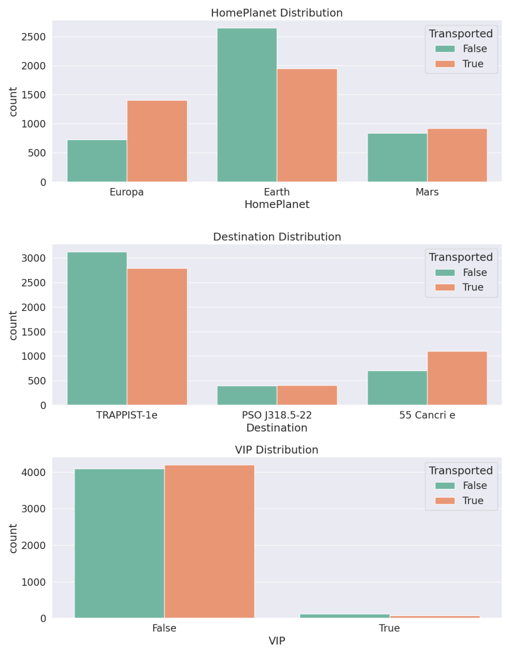


Figure 10. Three Features Distribution

The bar graphs from **Figure 10** provide information that the majority of passengers from the Homeplanet feature were from Earth. However, passengers from Earth were comparatively less likely to be transported. In contrast, passengers from Mars and Europa exhibited different patterns, with those from Europa being highly transported and those from Mars having an equal likelihood of being transported. In the Destination feature, it was found that most passengers were transported to Trappist-1e. This indicates a strong preference for this particular destination among the passengers. Last, Regarding the VIP feature, one category overwhelmingly dominated other categories. As a result, this feature may not be useful for our modelling purposes and could potentially lead to overfitting.

C. Feature Engineering

The feature engineering focuses on enhancing the dataset by creating new features, handling categorical variables and data, as well as applying transformations. Several new features were introduced, such as "Group_Size" and "Travelling Solo" derived from the "PassengerId" feature, and "Cabin_Deck," "Cabin_Number," and "Cabin_Side" derived from the "Cabin" feature. These new features provided insights into group sizes, solo travellers, cabin locations, and passenger transportation outcomes. Additionally, the "Age" feature was utilized to create the "Age Group" feature, allowing for a more refined categorization based on transportation likelihood.

Accordingly, we also perform feature engineering on Expenditures feature which result in the creation of the "Total Expenditure" feature and the "No Spending" boolean feature. Overall, the feature engineering which aims to prepare the data before performing modelling will be further addressed in the sections below.

1. Creating New Feature From "PassengerId" Feature

In the dataset, each passenger ID follows the format “gggg_pp”, where “gggg” represents the group the passenger is travelling with and pp indicates the number of people within the group. From this, a new feature called "Group_Size" can be created to indicate the total number of members within each group. Additionally, another feature, "Travelling Solo" can be introduced to identify whether a passenger is travelling alone or in a group.

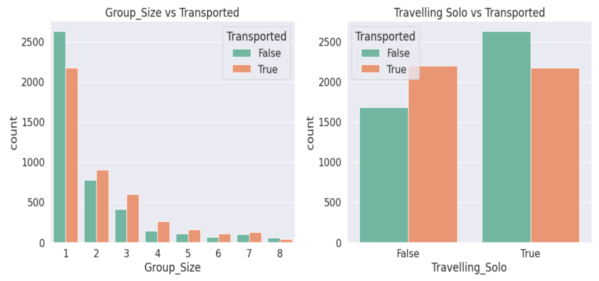


Figure 11. New feature from PassengerID feature

Upon analysis, we observed that in **Figure 11** a significant proportion of passengers were travelling alone, as indicated by the Group_Size feature. Apparently, we view that passengers travelling solo had a relatively lower likelihood of being transported compared to those travelling in a group. Furthermore, as we don't require Group & Member features any more so we will drop those features from both datasets.

2. Creating New Feature using "Cabin" Feature

The Cabin feature is in the format "deck/num/side, where the deck represents the deck location, num denotes the deck number, and the side can be either P for port or S for starboard. To leverage this information, three new features can be created: Cabin_Deck, Cabin_Number, and Cabin_Side. It is important to note that the Cabin feature contains NaN values, which need to be handled appropriately during the splitting process.

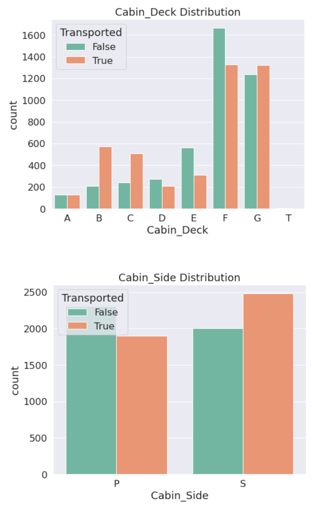


Figure 12. New feature from Cabin feature

The bar graphs in **Figure 12** above provide information that the majority of passengers were located on decks F and G. Conversely, there were only a few passengers in Cabin_Deck T. Passengers in Cabin_Deck B and C had a high likelihood of being transported. Regarding Cabin_Side, it was observed that approximately half of the passengers were located on side S, while the other half were on side P. Notably, passengers on side S had a significantly higher likelihood of being transported, while those on side P were equally likely to be transported.

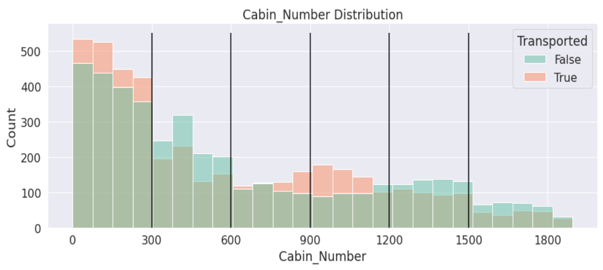


Figure 13. Cabin_Number Distribution

From the **Figure 13** above, we have an idea to divide the Cabin_Number into different regions, with each region encompassing a group of 300 passengers. Hence, a new feature called "Cabin_Regions" is created to indicate the region to which a passenger's cabin number belongs. This additional feature provides a more refined categorization of cabin numbers, allowing for a better understanding of the distribution and patterns within the dataset.

3. Creating New Feature "Cabin_Regions" From "Cabin_Number

Figure 14 below provides information that the Cabin_Number feature is no longer necessary and can be dropped from the dataset. We can see that passengers from Cabin_Region1 are highly likely to be transported, whereas the likelihood of transportation decreases as the cabin region number increases. This trend suggests a correlation between cabin region and transportation outcomes, allowing for a deeper understanding of the factors influencing passenger transportation.

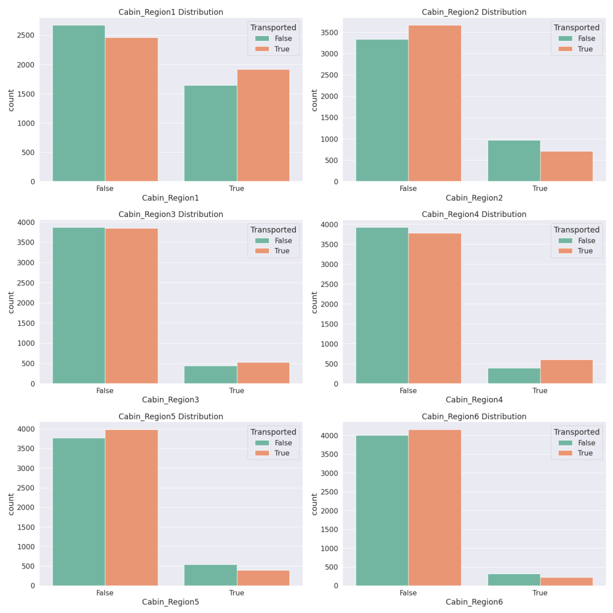


Figure 14. Cabin Region Distribution

4. Creating New Feature From "Age"

It was observed that the ages could be divided into different groups based on their transportation status. To capture this information effectively, a new feature called "Age Group" will be created, where the Age values will be segmented into distinct groups based on the insights gained from the EDA.

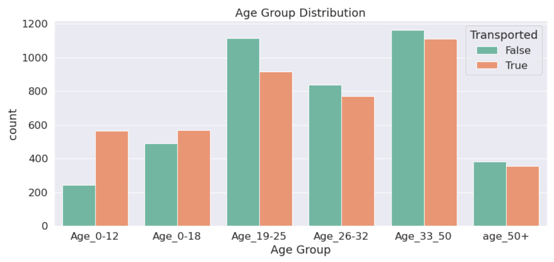


Figure 15. Age Group Distribution

Overall, the table on **Figure 15** provides information that Age groups of 0-12 and 0-18 are more likely to be transported compared to those who are not transported. Consequently, passengers in the Age groups of 19-25, 26-32, and 33-50 exhibit a lower likelihood of being transported compared to their counterparts who are not transported. Lastly, passengers in the Age group of 50 and above display a nearly equal likelihood of being transported.

5. Creating New Features Using All Expenditure Features

Following that, we can create a new feature called Total Expenditure by combining all the individual expenditures. Additionally, we can derive a boolean feature named No Spending from the Total Expenditure feature, indicating whether a passenger has spent zero expenses. This feature will be set to True for passengers with no expenditure. Furthermore, to capture the different levels of expenses, we can split the Total Expenditure into distinct categories and create a new feature called Expenditure Category.

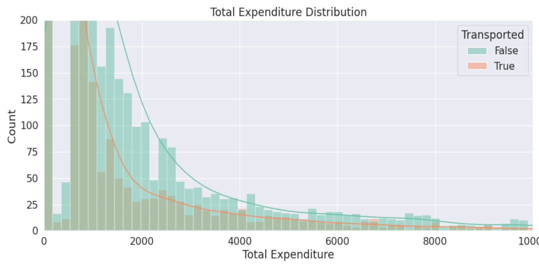


Figure 16. Age Group Distribution

Based on the measures of central tendency obtained from the **Figure 16** above, we can define the following expense categories: No Expense, Low Expense, Medium Expense, and High Expense. Passengers with a Total Expenditure of zero will fall under the No Expense category, while those with Total Expenditure ranging from 1 to 716 will be classified as Low Expense. Similarly, passengers with Total Expenditure between 717 and 1441 will be categorized as Medium Expense, and those with a Total Expenditure exceeding 1441 will be classified as High Expense. By creating these expenditure categories, we can effectively capture the varying levels of expenses incurred by the passengers, enabling the machine learning model to learn and identify patterns associated with different expenditure levels.

6. **Applying Log Transformation on Expenditure Features**

We observed that the Expenditure features (RoomService, FoodCourt, ShoppingMall, Spa, and VRDeck) exhibited a right-skewed distribution which indicates a large number of outliers. To address this issue and achieve a more normalized distribution, a log transformation will be applied to these features.

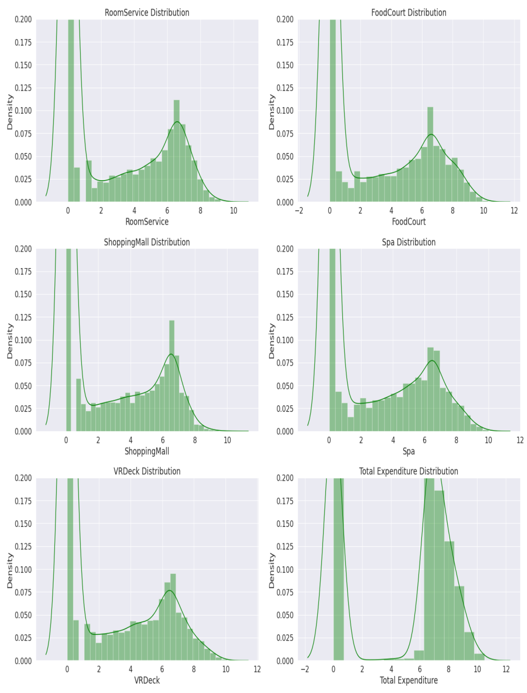


Figure 17. Expenditure Features

The bar and line graph presented on **Figure 17** above, shows the log transformation being applied to the aforementioned expenditure features. As a result, those six features are no longer skewed to either direction. Consequently, it is essential to also apply the same transformation to the "Total Expenditure" feature. This ensures that all relevant variables used in the modeling process are consistently transformed, enhancing the model's ability to capture meaningful patterns and make accurate predictions. By normalizing the data and mitigating the impact of outliers, the transformed features will facilitate more robust and reliable modeling, ultimately leading to improved performance and insights.

7. **Feature Encoding**

In order to prepare our data for encoding and analysis, we need to handle certain features that contain boolean values presented on **Figure 18**. These features include CryoSleep, VIP, Travelling_Solo, No Spending, Cabin_Region1, Cabin_Region2, Cabin_Region3, Cabin_Region4, Cabin_Region5, and Cabin_Region6. To ensure compatibility and facilitate encoding, we will convert the data type of these features.

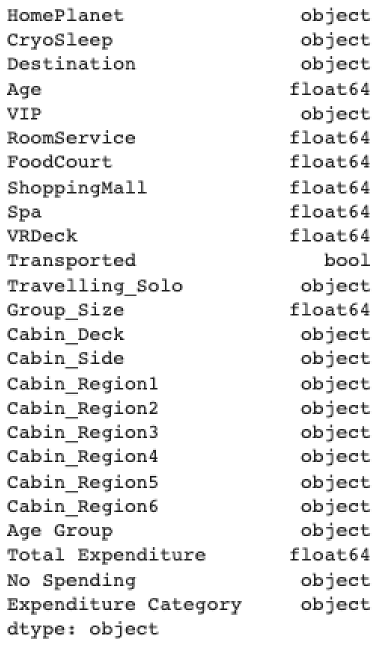

```
HomePlanet              object
CryoSleep               object
Destination             object
Age                    float64
VIP                     object
RoomService            float64
FoodCourt              float64
ShoppingMall           float64
Spa                    float64
VRDeck                 float64
Transported               bool
Travelling_Solo         object
Group_Size             float64
Cabin_Deck              object
Cabin_Side              object
Cabin_Region1           object
Cabin_Region2           object
Cabin_Region3           object
Cabin_Region4           object
Cabin_Region5           object
Cabin_Region6           object
Age Group               object
Total Expenditure      float64
No Spending             object
Expenditure Category    object
dtype: object
```

Figure 18. New Data Type on All Features

For nominal categorical features, we will utilize One Hot Encoding. This technique will transform the categorical variables into binary vectors, where each category is represented by a separate binary column. This encoding approach allows for the representation of categorical data in a machine-learning model that requires numerical inputs. On the other hand, for ordinal categorical features, we will employ Label Encoding. This technique assigns a unique numerical label to each category, thereby converting the ordinal variables into a numerical format that preserves the ordinal relationship between the categories.

8. **Feature Scaling and Splitting**

In this method, feature scaling is applied using the StandardScaler from the scikit-learn library. The dataset is transformed using the scaler, resulting in a scaled dataset'. Additionally, the 'test_df' dataset is also transformed using the same scaler.

Next, the data is split into training and testing sets for models that don't require scaled data. The dataset is split into two parts which are train and test dataset. The test size is 20% of the data while the remaining 80% is used for training.

The random state parameter is set to 0 for reproducibility. The shapes of the resulting training and testing sets are printed to provide an overview of their dimensions. Also, the data for models that require scaled data is also spitted and printed with the same function and format as the models that don't require scaled data.

D. Modeling

In the modeling phase of our project, we will utilize the power of Ensemble Learning, a powerful technique that involves the combination of two or more machine learning algorithms to achieve improved performance compared to using the algorithms individually. Instead of relying on a single model, ensemble learning leverages the predictions made by individual learners **[4]**. These learners, often referred to as base models or weak learners, are combined using a predefined combination rule to produce a single prediction that is more accurate and reliable **[5]**.

By harnessing the collective intelligence of multiple models, an ensemble can often achieve superior performance compared to any single model on its own **[6]**. This approach aims to create a more robust and accurate predictive model by leveraging the strengths of different algorithms and mitigating the weaknesses of individual models. Specifically, we will employ two prominent ensemble methods: Random Forest and eXtreme Gradient Boosting.

To further enhance the performance of these ensemble models, we will use hyperparameter tuning. Hyperparameters are settings that influence the learning process of the algorithms and can significantly impact their performance **[7]**. We will employ GridSearch to systematically search through a range of hyperparameter values and select the best combination that optimizes the model's performance. By fine-tuning the hyperparameters, we can ensure that our Random Forest and eXtreme Gradient Boosting models are optimized for our specific task and deliver the best possible results.

1. Random Forest

Random Forest (RF) is an ensemble learning algorithm that combines the collective power of multiple decision trees. It was developed by **[8]** with the aim of achieving high accuracy in learning tasks and has been proven to be effective in numerous datasets **[9]**. RF constructs an ensemble of decision trees by bootstrapping the training data, with each tree using a randomly selected subset of features for splitting **[10]**. This randomness enhances the algorithm's ability to handle missing data while maintaining accuracy. As a tree-based ensemble learning technique, RF utilizes bagging, where each node in the decision tree selects the best split from a random subset of predictors, providing robustness against overfitting **[11]**.

The Random Forest algorithm creates a forest of randomly constructed decision trees to make predictions as depicted in **Figure 19**. Unlike standard decision trees, where each node is split using the best split among all variables, RF introduces an additional layer of randomness **[12]**. By combining the outputs of individual trees, RF takes advantage of the diverse perspectives offered by different subsets of features and training data samples. This approach enables RF to deliver improved accuracy and generalization compared to using a single decision tree.

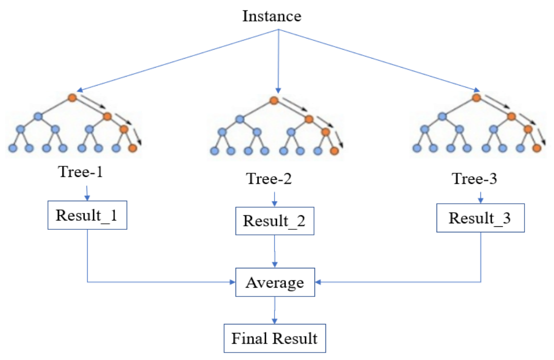


Figure 19. Random Forest Illustration

During the construction of decision trees, a subset of features is randomly chosen at each node, reducing the risk of overfitting and promoting robustness **[8]**. The final prediction is determined by aggregating the predictions of all the decision trees, often through averaging or majority voting. By employing bootstrapping, where training data samples are randomly selected with replacement, RF assembles randomized decisions and bases the final decision on the majority vote. This combination of techniques makes Random Forest a powerful ensemble learning algorithm for predictive modelling.

2. eXtreme Gradient Boosting

eXtreme Gradient Boosting (XGBoost) is one of the ensemble decision tree-based algorithms, which follows the principle of gradient boosting framework **[13]**. It is a scalable and highly accurate algorithm used for classification tasks. Different from Random Forest, each tree model in XGBoost minimizes the residual from its previous tree model. The traditional gradient-boosted decision trees use only the first derivative of error information. However, XGBoost performs the second-order Taylor expansion of the loss function and uses both the first and second derivatives. The visualization of XGBoost can be seen in **Figure 20** below. It is to be noted that the residual from the first decision tree is fed to the second decision tree so as to reduce the residual and this continues **[14]**.

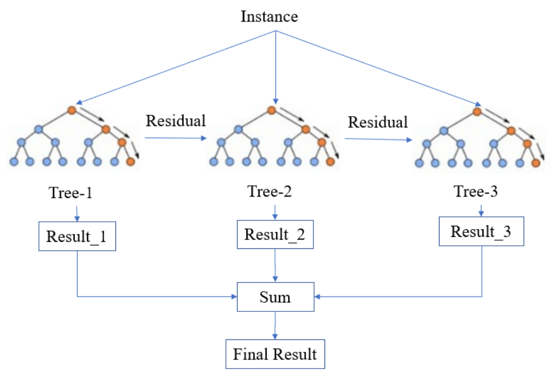


Figure 20. eXtreme Gradient Boosting Illustration

In addition, the XGBoost tool supports customized loss functions. Unlike other gradient boosting algorithms, XGBoost utilizes a regularization model formalization within its loss function to effectively control the overfitting of the data **[15]**. This regularization term helps strike a balance between fitting the training data well and avoiding overfitting, leading to a more robust and accurate model. By incorporating this regularization term into its loss function, XGBoost effectively prevents overfitting, allowing for better generalization and enhanced predictive performance **[16]**.

The XGBoost algorithm offers several advantages that set it apart from other machine learning algorithms. It requires minimal feature engineering as it can handle situations like data normalization and feature scaling. It also has the ability to handle missing values, providing flexibility in data preprocessing. XGBoost provides feature importance rankings, aiding in understanding input features and facilitating feature selection. Moreover, it is known for its speed, especially when dealing with large datasets, and it exhibits robustness against overfitting.

3. Cross Validation and Hyperparameter Tuning

In our modelling phase, we employ grid-based cross-validation hyperparameter tuning to optimize our model. This involves optimizing key parameters such as the number of trees in a forest, the maximum number of features for splitting in child nodes, the tree depth, and the splitting criterion. To evaluate the performance of machine learning models, it is common practice to divide the complete dataset into three subsets: training, validation, and testing. The training set is used to train the model, while the validation set monitors its performance. Finally, the model's ability to generalize is assessed by testing it on a set of unseen samples which is the testing set **[17]**.

However, this approach reduces the size of the dataset, potentially leading to an insufficiently trained model. To overcome this, cross-validation is commonly employed **[17]**. In this project, we use K-fold cross-validation, where the data is divided into K subsets, with one subset used for validation and the other K-1 subsets used for training. This resampling technique helps avoid over-reduction of the training set and provides a more robust method of performance evaluation of the model.

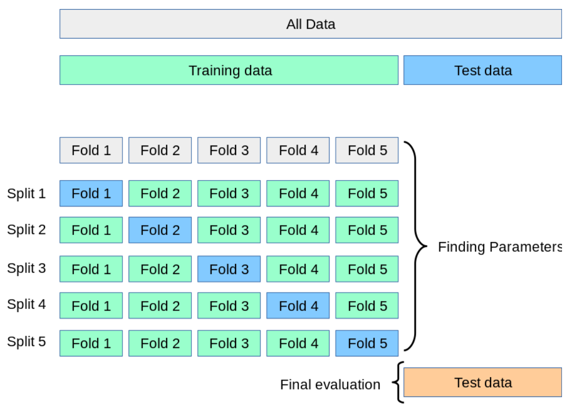


Figure 21. Five-Fold Cross Validation with GridSearch

Hyperparameter tuning plays a crucial role in developing reliable machine-learning models by reducing overfitting and enhancing adaptability to new data. The selection of optimal hyperparameters significantly contributes to improving the model's accuracy **[18-19]**. To automate this process and avoid manual tuning, various techniques have been developed, including grid search and random search hyperparameter optimization **[20]**. Grid search explores all possible values within a predefined domain for the hyperparameters, while random search selects distinct hyperparameter values randomly over a specified number of iterations **[20]**. In our approach, we utilized the GridSearchCV function, to perform hyperparameter tuning using a grid search technique with five-fold cross-validation. This method involved dividing the data into five subsets for training and selecting the best hyperparameters for the model. **Figure 21** depicts the five-fold cross-validation used in this project for training and for the hyperparameter selection of the model.

E. Performance Metrics

The evaluation metrics were selected to provide a comprehensive understanding of the model's performance in terms of overall accuracy and the ability to correctly identify positive and negative instances. The **Figure 22** below provides us with the four metrics that we used and how to calculate them, while **Table I** below describes the variables that we used in the formula.

$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}}$$

$$\text{recall} = \frac{\text{tp}}{\text{tp} + \text{fn}}$$

$$\text{accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}}$$

$$F_1 \text{ score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Figure 22. Metrics Formula

Table I. Metrics Variable Explanation

Variables	Description
True Positive (TP)	Instances that are correctly predicted as positive (belonging to the positive class
True Negative (TN)	Instances that are correctly predicted as negative (belonging to the negative class
False Positive (FP)	Instances that are incorrectly predicted as positive (predicted as positive but actually belonging to the negative class
False Negative (FN)	Instances that are incorrectly predicted as negative (predicted as

	negative but actually belonging to the positive class

The F1 score is a balanced measure that combines precision and recall, providing an overall assessment of the model's performance by considering both the false positives and false negatives. It is particularly useful in scenarios where there is an imbalance between the positive and negative classes. Accuracy is a commonly used metric that measures the percentage of correctly classified instances out of the total. It provides an overall measure of the model's correctness but may not be suitable for imbalanced datasets.

Precision measures the proportion of correctly predicted positive instances out of the total predicted positives. It is valuable in situations where minimizing false positives is crucial, such as spam detection or disease diagnosis. Recall, also known as sensitivity or true positive rate, measures the proportion of correctly predicted positive instances out of the actual positive instances. It is important in scenarios where identifying all positive instances is a priority, such as fraud detection or disease screening.

These metrics were chosen to assess different aspects of the classification performance and to provide a comprehensive evaluation of the models' effectiveness. By considering accuracy, precision, recall, and F1 score together, a more nuanced understanding of the models' performance can be gained.

IV. Result and Analysis

We obtained the outcome from applying the model to the test dataset and we provided a detailed analysis of the results. The evaluation encompasses various performance metrics, including accuracy, precision, recall, and F1 score to assess the model's predictive capabilities.

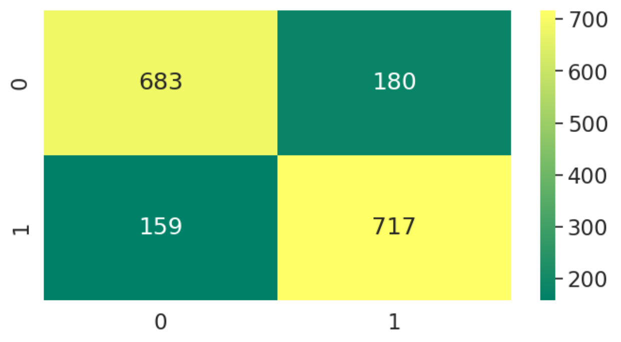


Figure 22. Random Forest Confusion Matrix

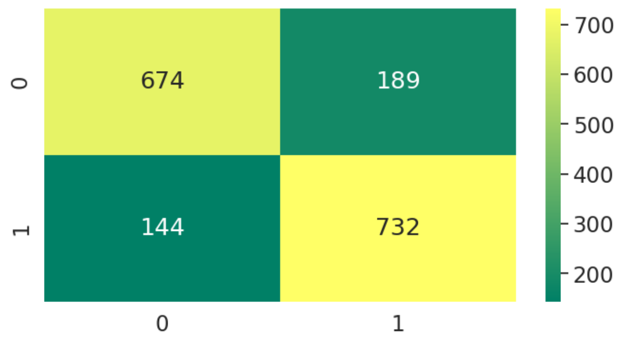


Figure 23. eXtreme Gradient Boosting Confusion Matrix

Figure 22 and **Figure 23** above provides information about the confusion matrix of the Random Forest model and XGBoost model, where the y-axis represents the model prediction with 0 being the passenger does not get transported to another dimension and 1 being the passenger does get transported to another dimension. On the other hand, the x-axis represents the actual results of each passenger with 0 being the passenger does not get transported to another dimension and 1 being the passenger does get transported to another dimension. The Random Forest model was able to predict correctly that 683 passengers were in fact not transported to another dimension and 717 passengers were in fact transported to another dimension, while the model predicted incorrectly for the rest of the passengers in the test dataset.

With XGBoost, the model was able to predict correctly that 674 passengers were in fact not transported to another dimension and 732 passengers were in fact transported to another dimension. From the two confusion matrices, we can see that the XGBoost model performs better at predicting the true negatives of the dataset, while Random Forest model performs better at predicting the true positives of the dataset.

Table 2. Model Comparison

Model / Metrics	*Accuracy*	*Recall*	*Precision*	*F1-Score*
Random Forest	80.5%	0.81	**0.79**	0.80
XGBoost	**80.8%**	**0.83**	**0.79**	**0.81**

The presented **Table 2** provides an evaluation of two machine learning models, namely Random Forest and XGBoost, based on several performance metrics. These models were assessed in terms of accuracy, recall, precision, and F1-score.

Random Forest achieved an accuracy of 80.5%, demonstrating its ability to classify correctly. Furthermore, it exhibited a recall value of 0.81, indicating its competence in identifying positive instances, while maintaining a precision of 0.79, reflecting its capability to accurately label positive predictions. The F1-score, which combines precision and recall, yielded a value of 0.80 for Random Forest, highlighting its overall effectiveness.

On the other hand, XGBoost exhibited slightly better performance with an accuracy of 80.8%. It demonstrated a higher recall value of 0.83, revealing its proficiency in correctly identifying positive instances while maintaining a precision of 0.79. Consequently, the XGBoost model achieved an F1-score of 0.81, suggesting its superior overall performance compared to Random Forest. These results demonstrate the effectiveness of both ensemble models in the context of the evaluated metrics.

V. Conclusion

Based on the results of the proposed models, the XGBoost model shows high accuracy in predicting passengers' fate whether they got transported to another dimension or not, compared to Random Forest model. With 8,693 instances during training, the accuracy was 80.8%. The model evaluation results with test data showed a precision of 0.79, recall of 0.83, and F1-Score of 0.81, the

XGBoost model was considered capable of obtaining accurate results in predicting passengers' fate between transported and not transported to another dimension.

However, this project was limited to using machine learning model only, due to the limitation placed upon this project requirements. As such, the performance results are not as high as it can be, and a more robust and improved model and process may be needed to be developed to reach better performance results when limited to the machine learning model only. In future work, a deep learning approach may be used to solve the problem and for better performance results.

References

[1] "Spaceship Titanic," Kaggle, https://www.kaggle.com/competitions/spaceship-titanic (accessed Jun. 11, 2023).

[2] E. G, I. N. Sneddon, and N. B, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences, vol. 247, no. 935, pp. 529–551, 1955. doi:10.1098/rsta.1955.0005

[3] E. Ekinci, S. İ. Omurca, and N. Acun, "A Comparative Study on Machine Learning Techniques Using Titanic Dataset," 7th International Conference on Advanced Technologies , Apr. 2018. doi:978-605-68537-1-5

[4] I. D. Mienye and Y. Sun, "A survey of Ensemble Learning: Concepts, algorithms, applications, and prospects," IEEE Access, vol. 10, pp. 99129–99149, 2022. doi:10.1109/access.2022.3207287

[5] F. Opolka, "Ensemble Learning", Technische Universitat Munchen. 2017.

[6] R. Brecht and A. Bihlo, Towards replacing precipitation ensemble predictions systems using machine learning, 2023. doi:10.2139/ssrn.4437064

[7] J. Wu, X.-Y. Chen , L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization," Journal of Electronic Science and Technology, vol. 17, no. 1, pp. 26–40, Mar. 2019. doi:10.11989/JEST.1674-862X.80904120

[8] Breiman, L. "Random Forests". Machine Learning, 45, 5-32. 2001.

[9] S. Janitza and R. Hornung, "On the overestimation of Random Forest's out-of-bag error," PLOS ONE, vol. 13, no. 8, 2018. doi:10.1371/journal.pone.0201904

[10] R. Díaz-Uriarte and S. Alvarez de Andrés, "Gene selection and classification of microarray data using Random Forest," BMC Bioinformatics, vol. 7, no. 1, 2006. doi:10.1186/1471-2105-7-3

[11] R. Punnoose and P. Ajit, "Prediction of employee turnover in organizations using machine learning algorithms," International Journal of Advanced Research in Artificial Intelligence, vol. 5, no. 9, 2016. doi:10.14569/ijarai.2016.050904

[12] A. Liaw and M. Wiener, "Classification and Regression by randomForest," Northwestern University, vol. 2, no. 3, Dec. 2002.

[13] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," The Annals of Statistics, vol. 29, no. 5, 2001. doi:10.1214/aos/1013203451

[14] W. Wang, G. Chakraborty, and B. Chakraborty, "Predicting the risk of chronic kidney disease (CKD) using machine learning algorithm," Applied Sciences, vol. 11, no. 1, p. 202, 2020. doi:10.3390/app11010202

[15] R. Punnoose and P. Ajit, "Prediction of employee turnover in organizations using machine learning algorithms," International Journal of Advanced Research in Artificial Intelligence, vol. 5, no. 9, 2016. doi:10.14569/ijarai.2016.050904

[16] Y. Li and W. Chen, "A comparative performance assessment of ensemble learning for credit scoring," Mathematics, vol. 8, no. 10, p. 1756, 2020. doi:10.3390/math8101756

[17] M. Kuhn and K. Johnson, Applied Predictive Modeling. New York: Springer, 2019.

[18] Bardenet, R., Brendel, M., Kégl, B., & Sebag, M. (2013). Collaborative hyperparameter tuning. International Conference on Machine Learning.

[19] Bergstra, J., Yamins, D. &amp; Cox, D.. (2013). Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. Proceedings of the 30th International Conference on Machine Learning, in <i>Proceedings of Machine Learning Research 28(1):115-123.

[20] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," Journal of Machine Learning Research, vol. 13, no. 1, Mar. 2012.