FORAGE PROJECT

BRITISH AIRWAYS

DATA SCIENCE AND ANALYTICS

Random Forest-Based Analysis of Significant Factors in Customer Booking Behavior

Author:
Sidharth Nambiath

Date: January 5, 2024

Abstract

In the age of data-driven decision-making, understanding customer behavior and predicting their actions is crucial for businesses in the airline industry to remain competitive. This project focuses on the classification task of predicting whether potential customers will book a flight with British Airways or not based on their demographic attributes and the amenities provided by the airline. The objective is to identify the most influential features that drive the booking decision. Using feature engineering, data cleaning and data wrangling techniques to prepare the given dataset of various customers' booking data, a random forest classifier algorithm was trained to eventually rank the importance of the different features of the dataset. It was found, with an accuracy of 85%, that the purchase lead, that is, the time between the date of travel and the date of booking, was by far the most influential feature determining if a booking was to be made. With further analysis utilising domain knowledge, these results may prove to be insightful to marketers in the airline industry. For those who are interested, the Jupyter notebook used to perform this analysis can be found at (1).

1 Introduction

Though controversial, British Airways is often considered the first or second oldest airline in the world, having begun operations in 1919 (2). It also serves the second highest number of international destinations (3). In the past, airlines would sell tickets through agents via telephone, or in person. This meant that searching around for better pricing was an inefficient and time-consuming process. However, in the digital age, through comparison websites such as SkyScanner and Google Flights, it is easy, and often necessary, to compare flight prices, in order to get the best value for money. As a result, it has become paramount for airlines to be proactive in using customer data to ensure customer needs are met long before they step foot in the airport. Machine learning algorithms provide a powerful means to analyze this data, allowing airlines to gain valuable insights into passenger preferences, optimise pricing strategies, tailor in-flight services, and ultimately enhance the overall passenger experience based on what they desire.

2 Theoretical Background

2.1 Random Forest Algorithm

The random forest algorithm was introduced by Leo Breiman in 2001 (4). In the context of classification, a random forest consists of an ensemble of decision trees, which each cast a vote on which class they select. The output of the model is the majority vote decided upon by the ensemble (5). This method corrects for the tendency of decision trees to overfit (6). The word ”ensemble” was chosen for a reason - whilst decision trees are common supervised learning algorithms, forming an ensemble of decision trees, which undergoes a process of bootstrapping aggregation and feature randomness (8). This involves randomly selecting data with replacement sampling and training each model on this data independently, hence reducing the likelihood the model will overfit. The random forest algorithm has three principle hyper-parameters, those being the number of features, the number of trees, and the size of the node. The latter essentially describes the minimum number of data points required before creating a decision point (or node) within a singular tree.

2.2 Model Evaluation

In order to gauge the accuracy of the model, some form of model evaluation metric must be used.

2.2.1 Confusion Matrix

A confusion matrix is a matrix which compares the absolute number of true positives (P_{true}), false positives (P_{false}), true negatives (N_{true}) and false negatives (N_{false}). In other words, P_{false} is a Type I error, and N_{false} is a Type II error. Based on the results

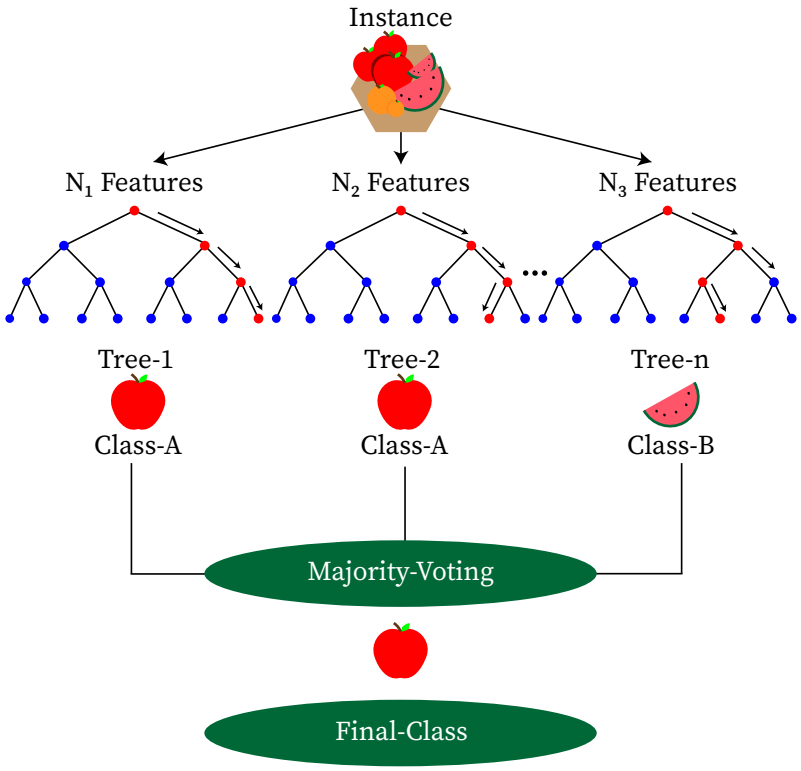

Figure 1: A high level illustration of the process undergone in a random forest algorithm, adapted from (7).

from figure 2, accuracy can be calculated as follows:

$$\text{Accuracy} = \frac{P_{true} + N_{true}}{P_{true} + N_{true} + P_{false} + N_{false}} \tag{1}$$

2.2.2 Classification Report

Statistical learning packages often summarise evaluation metrics in a classification report. These metrics (in addition to accuracy) include:

3 Data Collection and Pre-Processing

3.1 Data Collection

For this project, the aim was to identify the features that were most influential in whether a booking with British Airways would have been made. A dataset containing various booking details was provided through forage, which contained the columns outlined in Table 2, each with 50,000 rows and no null entries. The target variable was ”booking complete”, making this a binary classification problem (either YES or NO).

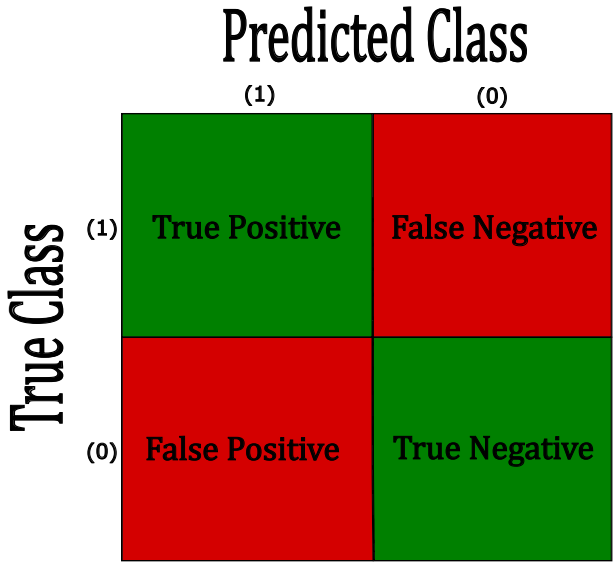

Figure 2: A typical confusion matrix for binary classification

.

Metrics	Definition	Formula
Precision	Precision is the proportion of true positives to the sum of all positives.	$\frac{P_{true}}{P_{true}+P_{false}}$
Recall	Recall is the proportion of true positives to the sum of true positives and false negatives.	$\frac{P_{true}}{P_{true}+N_{false}}$
F1 Score	The F1 is the harmonic mean of precision and recall. The model's performance is expected to be best when $F1 = 1.0$. This metric is particularly useful for uneven class distributions.	$\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$

Table 1: Evaluation metrics found in a classification report.

3.2 Pre-Processing

3.2.1 Data Conversion

As evident in Table 2, some of the columns did not contain integer or float data types. As a result, data conversions were made in order to prepare them for statistical analysis. The columns in question were "sales channel", "trip type", and "flight day". As these contain categorical variables (with a minimal number of unique options), a simple numerical mapping was sufficient, as demonstrated below for "trip type" which had entries of either "RoundTrip", "CircleTrip" or "OneWay".
However, for two remaining columns, "route" and "booking_origin", there were 799 and 104 unique entries respectively, thus mapping each unique entry to a numeric value would have been inefficient and time-consuming. Instead, a method known was one-hot encoding was employed. This essentially means thats each unique entry is transformed to its own column, and for each row, if that entry is true the element will be 1, and if that entry is false the element will be 0 (9). As an illustration, if there

Column	Dtype	Description
num_passengers	int64	Number of passengers
sales_channel	object	Sales channel used for booking (internet, mobile)
trip_type	object	Type of trip (e.g. one-way, round trip)
purchase_lead	int64	Time between booking and travel dates (in days)
length_of_stay	int64	Length of stay for the trip (in days)
flight_hour	int64	Hour that the flight departs
flight_day	object	Day of the week of the flight
route	object	Flight route
booking_origin	object	Origin of the booking (country)
wants_extra_baggage	int64	Indicates if passengers want extra baggage
wants_preferred_seat	int64	Indicates if passengers want preferred seats
wants_in_flight_meals	int64	Indicates if passengers want in-flight meals
flight_duration	float64	Duration of the flight (in hours)
booking_complete	int64	Indicates if the booking process is complete

Table 2: Column Information

were a subset of the table before implementing one-hot encoding (see Table 3), then

Route	Categorical value of route	Booking Complete
LHRJFK	1	1
FRAMAN	2	0
NCLBLR	3	1
FRAMAN	4	1

Table 3: Route data prior to applying one-hot encoding.

after implementing one-hot encoding, the data is transformed into the following format in Table 4: where the `pandas` module provides an easy method for one-hot encoding: `pd.get_dummies()`.

Route: LHRJFK	Route: FRAMAN	Route: NCLBLR	Booking Complete
1	0	0	1
0	1	0	0
0	0	1	1
0	1	0	1

Table 4: Route data after applying one-hot encoding

Listing 1: Python Code for Mapping `trip_type`

```
import pandas as pd

# Load dataset
# df = pd.read_csv("data/customer_booking.csv",
        encoding="ISO-8859-1")

# Display all the unique values in the 'trip_type' column
print(df['trip_type'].unique())

# Create a mapping for 'trip_type'
mapping_trip = {"RoundTrip": 1, "CircleTrip": 2, "OneWay": 3}

# Map 'trip_type' values using the mapping
df["trip_type"] = df["trip_type"].map(mapping_trip)
```

3.2.2 Data Normalisation

Now that the data is all numeric, normalisation (also known as feature scaling) techniques can be applied. This involves re-scaling the data to a standardised scale, which can improve the accuracy of the model for data in which each of the columns has large differences in magnitude, thereby ensuring that undue importance is not given to columns with large magnitude values. Two commonly used methods for feature scaling are min-max scaling, and z-score standardisation. In the former, the minimum value is redefined as being zero, the maximum value is redefined as being one, and the remaining data are assigned decimal values between one and zero, as per equation 2.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2}$$

In the latter, the data is transformed such that its mean is zero and its standard deviation is one. This technique is less sensitive to outliers than min-max scaling, but they are more useful for algorithms that assume normality. It follows equation 3, where μ and σ are the mean and standard deviation of the data before transformation respectively.

$$x_{scaled} = \frac{x - \mu}{\sigma} \tag{3}$$

However, tree-based algorithms such as random forest are invariant to scaling transformations. This is due to the nature of said algorithms; nodes are split based on single features, without any influence from other features. That being said, it is good practice to feature scale nonetheless, as it means the data is ready to be used with other ML algorithms if necessary (10). For comparison, the run time of the algorithm was measured seven times using the data scaled and unscaled, as shown in figure 3.

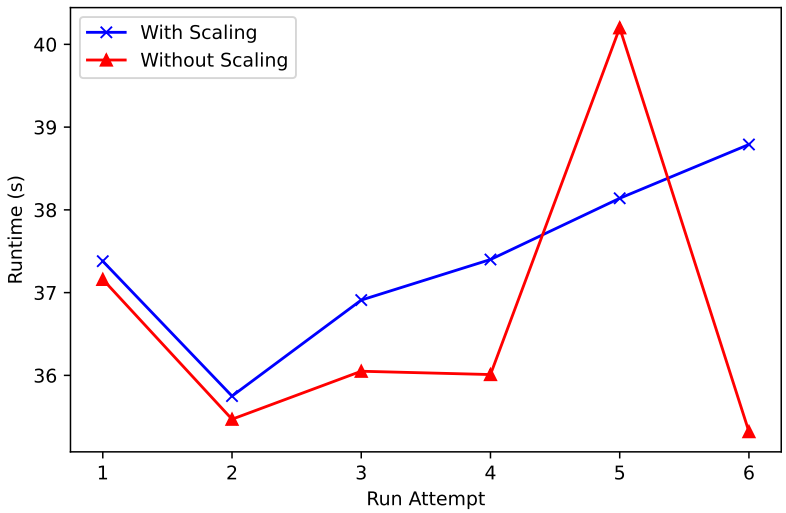

Figure 3: The runtime for the random forest algorithm using scaled data and unscaled data. Note that the time for the scaled data also includes the time elapsed in running the script that performs the scaling during the preprocessing stage.

.

4 Algorithm Implementation

The implementation was done solely using the scikit-learn python package; a statistical learning module built upon numpy, scipy and matplotlib (11). Firstly, the data features were scaled using the min-max scaler as described in 3.2.2, shown in code snippet below.

Listing 2: Python Code for Mapping `trip_type`

```
from sklearn.preprocessing import MinMaxScaler

# Specify the columns to scale
features = ['num_passengers', 'purchase_lead', 'length_of_stay'
    , 'flight_hour', 'flight_duration', 'flight_day', '
    sales_channel', 'trip_type', 'wants_extra_baggage', '
    wants_preferred_seat', 'wants_in_flight_meals']
scaler = MinMaxScaler()

# Scale the selected columns and convert to DataFrame
df_feat = pd.DataFrame(scaler.fit_transform(df[features]),
    columns=features)
```

This creates a new dataframe (`df_feat`), containing only the features and not the target variable ("booking complete"). Then, a training set and testing set is prepared as per the following code snippet.

Listing 3: Python Code for train-test split

```
# Importing necessary libraries
from sklearn.model_selection import train_test_split
```

```
3
4 # Splitting data into a training set and test set
5 X = df_feat
6 y = df['booking_complete'] # The target variable
7 X_train, X_test, y_train, y_test = train_test_split(X, y,
     test_size = 0.2, random_state = 42) # Using Pareto Principle
```

The train-test split was done according to the Pareto Principle, which essentially states that 80% of an effect is determined by 20% of the cause (12). From this point it is possible to optimise the hyperparameters, via GridSearch (where the model is evaluated for every combination of the hyperparameters and the best combination is chosen), or RandomSearch (where the model chooses random combinations of the hyperparameters before evaluating each combination), illustrated in figure 4 and 5 (13). However, for this task of ranking feature importance, it was not worth

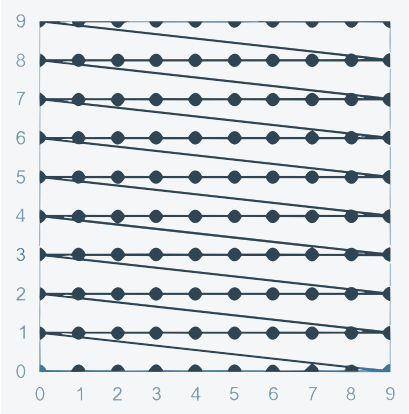

Figure 4: GridSearch Illustration

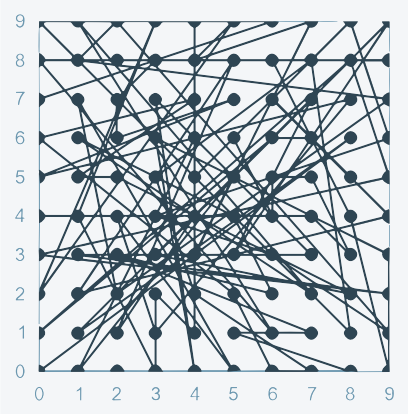

Figure 5: RandomSearch Illustration

the time and computational power to search for the optimised hyperparameters. This will become more apparent when the results are seen. Scikit-learn by default uses the same hyperparameter set, which can be found in (14). The fitting of the model is therefore very simple as demonstrated in the code snippet below, using the `RandomForestClassifier` from scikit-learn.

Listing 4: Python Code for fitting the Random Forest model

```
# Importing necessary libraries
from sklearn.ensemble import RandomForestClassifier

# Initialising and Fitting the Random Forest model
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test) # predicting on test data
```

The model essentially trains itself on the training features in `X_train` and the training target variables in `y_train` and then based on this, it selects test features in `X_test` and uses the model training to make predictions on whether those features led to a booking. This is then compared to the actual result, found in `y_test` to determine the accuracy of the model. As aforementioned in section 2.2, the evaluation metrics are summarised in the confusion matrix and classification report, which we can find as per the following code snippet.

Listing 5: Python Code for model evaluation

```
# Importing necessary libraries
from sklearn.metrics import accuracy_score,
    classification_report, confusion_matrix,
    ConfusionMatrixDisplay

# Model Evaluation
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Classification Report:\n", report)
print("Confusion Matrix:\n", cm)
```

In the Jupyter notebook, there is also code to produce graphical representations of the confusion matrix, shown in figure 6. As class 0 refers to customers that did not make a booking, and class 1 refers to those that did, it indicates 207 instances were true positives, compared to 185 false positives. This means the model was rather poor at classifying instances where a customer would make a booking. Conversely, the model was very good at classifying instances where the customer would not make a booking, having 1273 false negatives and 8335 true negatives. Using equation 1, the accuracy is calculated as roughly 85%. This tells us the model overall performed well across all classes. The classification report offers further evaluation statistics based on the formulae in Table 1. They once again paint the same picture; the model performed well on class 0 (no booking), but poorly on class 1 (booking).

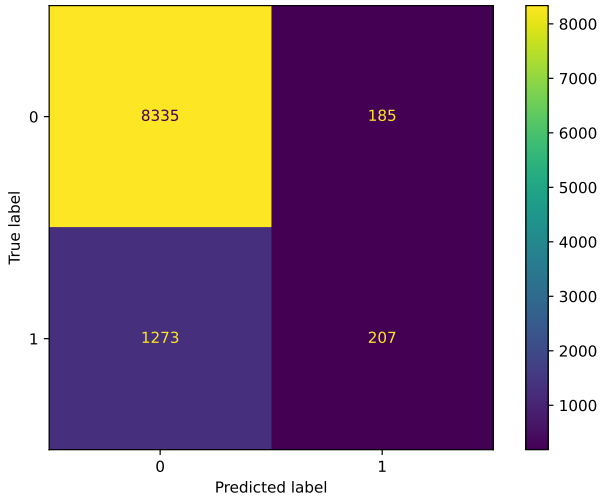

Figure 6: The confusion matrix for the Random Forest model's predictions on the test set

.

Table 5: Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.87	0.98	0.92	8520
1	0.53	0.14	0.22	1480
Accuracy			0.85	10000
Macro Avg	0.70	0.56	0.57	10000
Weighted Avg	0.82	0.85	0.82	10000

5 Results

Ultimately the aim of this project was to rank the feature importance in terms of how the features influence whether one does or doesn't make a booking. This is very simple to do with scikit-learn, utilising the `rf.feature_importances_` method to extract the feature importances, and then simply sorting them. Due to the one-hot encoding process in section 3.2.1, the number of features is now 915, therefore it is impractical to extract the importance of every feature. Instead, the top 15 features were visualised using the `seaborn` python package, as shown in figure 7.

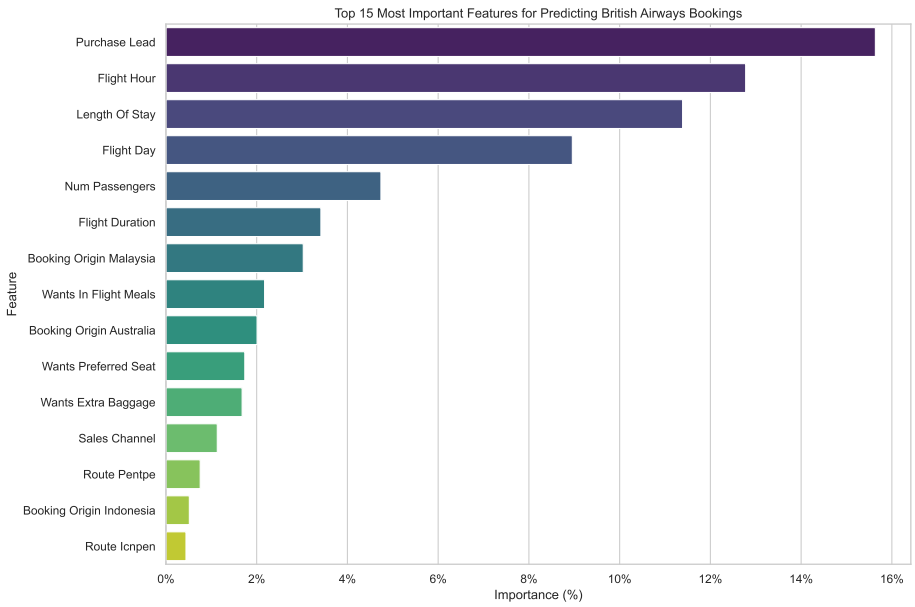

Figure 7: The feature importances represented by a bar plot

.

6 Evaluation and Conclusion

The results in figure 7 tells us that ”purchase lead” is the most influential feature in determining whether a customer makes a booking with British Airways. This means the time between the date of booking and the date of travel is what customers find most important, according to this dataset. Following this, the hour of departure; length of stay; and day of the flight are most significant for customers. From the one-hot encoded features (booking origin and route), it is worth performing more research on bookings originating in Australia and Malaysia, and the flight routes Penang to Taipei, Balcanoona to Penang, as these show the higher levels of significance in determining booking patterns.
There is a lot of scope for further analysis; given more time and/or computational power, it would be useful to use one of either GridSearch or RandomSearch from section 4 to find the optimal parameters, hopefully providing a more accurate model. Additionally, a completely new machine learning algorithm could be implemented to analyse the same data, allowing for comparisons with the output of the random forest model. Suitable examples could include Gradient Boosted Decision Trees (XGBoost) (15) and/or Support Vector Machine (16), though the latter may struggle since it requires a clear separation between the two classes.

References

[1] S. Nambiath. (2024) Github repository for project. Accessed on 5th January 2024. [Online]. Available: https://github.com/sid051201/BA/tree/main pages 2

[2] M. Ros. (2023) 10 oldest airlines in the world. Online. CNN Travel. [Online]. Available: https://edition.cnn.com/travel/article/worlds-oldest-airlines/index.html pages 3

[3] ——. (2022, April) Here's our breakdown of which airlines fly to the most destinations. Online. The Points Guy. [Online]. Available: https://thepointsguy.com/news/airlines-with-most-destinations/ pages 3

[4] L. Breiman, "Random forests," **Machine Learning**, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324 pages 3

[5] J. L. Speiser, M. E. Miller, J. Tooze, and E. Ip, "A comparison of random forest variable selection methods for classification prediction modeling," **Expert Systems with Applications**, vol. 134, pp. 93–101, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419303574 pages 3

[6] T. Hastie, R. Tibshirani, and J. Friedman, **The Elements of Statistical Learning**, 2nd ed. Springer, 2008. pages 3

[7] E. R. Sruthi, "Understand random forest algorithms with examples (updated 2024)," 21 Dec 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/ pages 4

[8] T. H. Lee, A. Ullah, and R. Wang, "Bootstrap aggregating and random forest," University of California at Riverside, Department of Economics, Working Papers 201918, 2019. [Online]. Available: https://EconPapers.repec.org/RePEc:ucr:wpaper:201918 pages 3

[9] A. Popov, "1 - feature engineering methods," in **Advanced Methods in Biomedical Signal Processing and Analysis**, K. Pal, S. Ari, A. Bit, and S. Bhattacharyya, Eds. Academic Press, 2023, pp. 1–29. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780323859554000041 pages 5

[10] A. Bhandari, "Feature engineering: Scaling, normalization, and standardization (updated 2023)," 27 Oct 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/ pages 7

[11] scikit learn, "scikit-learn machine learning in python," 2023. [Online]. Available: https://scikit-learn.org/stable/ pages 8

[12] H. Jack, "Chapter 10 - general design topics," in **Engineering Design, Planning, and Management (Second Edition)**, second edition ed., H. Jack, Ed. Academic Press, 2022, pp. 371–427. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128210550000104 pages 9

[13] D. Senapati, "https://medium.com/@senapati.dipak97/grid-search-vs-random-search-d34c92946318," 29 Aug 2018. [Online]. Available: https://medium.com/@senapati.dipak97/grid-search-vs-random-search-d34c92946318 pages 9

[14] Scitkit-learn, "sklearn.ensemble.randomforestclassifier," 2023. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html pages 9

[15] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)**, 2016, pp. 2075–2084. [Online]. Available: http://dx.doi.org/10.1145/2939672.2939785 pages 12

[16] V. N. Vapnik and A. Y. Chervonenkis, "On a class of pattern-recognition algorithms," **Autom. Remote Control**, vol. 25, no. 6, pp. 937–945, 1964. [Online]. Available: http://mi.mathnet.ru/at11678 pages 12