PREDICTING HOTEL RESERVATION OUTCOMES WITH MACHINE LEARNING APPROACH

ABSTRACT

The hotel industry is a primary service line industry that provides various travelers needs when they are away from home, whether those who are on personal trips, holidays, to business travelers. The travel and tourism industry itself contributes to 7.6% global GDP in 2022 and is responsible for 1 out of 5 jobs created worldwide between 2014-2019. In hotel operations, good practices in customer booking management and cancellations is important to make sure the hotel has good financial stability for sustainable business operations. Too many booking cancelations may lead to more empty rooms and less income for the hotel, however overbooking strategy may also damage the hotel's service reputation. Thus, this study aims to build the most optimal machine learning model to help hotel management in predicting customer booking outcomes.

The algorithms utilized in this study include the Decision Tree (DT), Support Vector Machine (SVM), and Random Forest (RF). Utilizing the Hotel Reservations Dataset from Kaggle, the data is split into a 70/30 ratio for training and testing, with implementation done in three variations: imbalanced data, balanced data, and balanced + normalized data. Implementation and validation done on multiple variations conclude that the best model comes from the Balanced Random Forest model. Hyperparameter tuning is then executed on all three models, with the DT model tuned on the *cp* value, the SVM model tuned on *epsilon* and *cost* value, and the RF model tuned on *ntree, mtry, nodesize*, and *maxnodes.* The tuned model performances concludes that the most optimal model comes from the tuned RF model with Mtry = 6 and Nodesize = 2, at Accuracy value of 89.54% and Balanced accuracy at 88.72%. The RF Gini values and DT tree structure also give insights that the important features impacting the prediction outcomes include *lead_time, no_of_special_requests* and *avg_price_per_room*; followed by *market_segment_types*, *arrival_month*, and *number_of_adults.*

Keywords: Hotel booking, Cancellation prediction, Random Forest, Decision Tree, Support Vector Machine

TABLE OF CONTENTS

1.0 INTRODUCTION

As part of the travel and tourism industry, hotels are a primary service that accommodates various range of travellers, ensuring comfort and convenience in doing their businesses while away from home. From short staying tourists to business travellers, the hotels have a primary objective in making sure the customers have a good experience while staying in their premises. Apart from serving rooms to travellers, hotels may also rely on bookings from their ballrooms or multifunction halls for weddings, meetings, or conferences, which may be directly related to rooms booked for the joining participants.

The World Travel and Tourism Council (WTTC) published that in 2022, the travel and tourism industry make up for 7.6% GDP contribution to global GDP value. Prior to the pandemic, the numbers are even much higher; with the industry accounting for 1 in 5 jobs created in the world between 2014 – 2019 or equivalent to 10.4% of global GDP in 2019 (WTTC, 2023). These statistics highlight the significant economic impact of the travel and tourism sector, however does not show the importance of efficient operations within hotels.

As businesses themselves, hotels need to closely monitor their customer bookings and cancellation statuses to maintain their financial stability and smooth-running operations. Problems related to cancellations may lead to a decline in the hotel's healthy financial status in the long run. Too many cancelations may lead more vacant rooms, leading to less earnings (Sánchez et al., 2020). Meanwhile utilizing an overbooking strategy to mitigate impact of cancellations may also lead to challenging situations where the hotel needs to compensate impacted customers with a refund, while damaging their own service quality reputation (Antonio et al., 2019a).

Customers themselves may cancel their hotel bookings due to a variety of reasons from common ones such as changing travel date, booked another place to stay, or even due to emergency situations. Online booking options with no advanced payment required also make it easier for customers to forfeit their booking on the scheduled date without any cancelation fee or notification to the hotel (Masiero et al., 2020). As much as these serves a better booking service and flexibility for customers, these digital options may also be damaging to the hotel if not managed properly.

As these cancellations an inevitable part of the business, ensuring a high number of confirmed bookings is an important aspect of a hotel's operation. Thus, it is important to develop a means to predict whether a booking made by a customer would end up being cancelled or not. In this study, machine learning algorithms will be utilized to build a suitable model on predicting hotel customers booking cancelation. In order to built these models, the Hotel Reservations Dataset obtained from Kaggle will be used. The dataset consists of both numerical variables such as number of children, room prices and number of nights booked, as well as categorical variables such as type of meal plan, room type, and market segment type. The target variable of the dataset is whether the booking end up being cancelled or not.

By leveraging the power of machine learning algorithms and utilizing the provided dataset, this study aims to develop an effective predictive model that can assist hotels in forecasting booking cancellations. The result of the study would then be able to be used by relevant stakeholders such as hotel managements to improve their strategies in mitigating booking cancellations.

Aim

From the study background explained above, the main aim of this study is to provide the most effective and suitable model in predicting hotel customers booking cancellation.

Objectives

In order to achieve the Aim of the study, the following are the objectives of the study:

1. To analyse the dataset for any patterns or relationships by doing data cleaning and exploration.
2. To compare the performance of machine learning algorithms in predicting the customer booking cancelations.
3. To investigate the key factors that contribute to customer booking cancellations, such as room types, repeated guests, and lead time.
4. To assess the robustness and reliability of the predictive models by conducting performance evaluation metrics.

Scope of the Study

This study focuses on developing a predictive model for hotel customer booking cancellations. This includes analyzing the dataset taken from Kaggle, applying and tuning machine learning algorithms, and evaluating their respective performance. The scope of the study is limited to investigating patterns and relationships within the dataset and does not extend to external factors such as economic conditions or hotel policies that may influence the result of the hotel booking cancellations. The study also does not cover implementation and deployment of the model on a real-world hotel data setting.

2.0 RELATED WORKS

In this section, related works done on predicting booking cancellations will be explained, describing the research objectives and results yielded. These works act as a benchmark tool for the model design and validation process.

2.1 Service Quality in Hotel Industry

In the field of hotel industry, studies utilizing machine learning to improve business processes have been done extensively. Specifically for predicting customer booking, booking related information are usually used as the input variables in making these machine learning models. A study by Satu et al. (2021) utilizes input variables from lead time, booking date, number of adults, children, and babies along with waiting lists and booking changes in order to predict the hotel booking outcome. Figure 1 showed the full list of variables used in the study, with the first variable *Canceled* as the target variable.

Canceled
Lead Time
Arrival Year
Arrival Month
Arrival Week Number
Arrival Day of Month
Stay Weekend Nights
Stays Week Nights
Adults
Children
Babies
Repeated Guest
Previous Cancellations
Previous Bookings Not Canceled
Booking Changes
Waiting List
adr
Required Car Parking Spaces
Total Special Requests
Reservation Status Date

Figure 1. Variable List (Satu et al., 2021)

The dataset used in the study was taken from Antonio et al. (2019b) which comprises of hotel booking information from one resort hotel and one city hotel. The study compares multiple traditional and ensemble machine learning techniques such as gradient boosting (GB), XG Boost, random forest (RF), and decision tree (DT) models. Among the traditional models, RF exhibits the highest accuracy, followed by DT, while for the boosting models, GB and XGB tops the list (Satu et al., 2021).

The authors of the dataset that Satu et al (2021) utilizes also performed a prediction study with the stated dataset in 2017, where boosted decision tree, decision forest, decision jungle, local deep support vector machine, and neural network algorithms are used to train the model. The dataset used specifically was 4 portuguese resort hotels. The study evaluated the performance metrics of the model for each of the hotel, and concludes with BDT and DF performing best overall, with the DF model being the best performer in 3 out of 4 hotels on accuracy mark up to 98.6% on hotel 3 testing. The AUC values of the BDT and DT models ranged from 0.93 – 0.977 (Antonio et al., 2017).

As the dataset utilized for this report originated from Kaggle, there are also several models sourced from the dataset that can act as a comparison. In processing and building the model, Sekeroglu (2023) utilizes Python to preprocess the data and running KNN, RF, and logistic regression (LR) models. Utilizing min-max normalization and one hot encoding, the dataset is split into 70/30 ratio for training and testing. The 5-KNN model reached an accuracy of 83.6%, with RF at 89.4% and LR at 80.2%.

Vinitnantharat (2023) on the other hand, utilizes 5 different classifiers in the modelling. After utilizing package pandas from Python to encode the categorical variables, the data undergo undersampling to balance the dominant target variable. The training process is done with around 18k observations, after outliers were eliminated from the training data. The performance of the models using the training set show an accuracy of 81.9% for DT, 85.1% for RF, 82.7% for KNN, 90.6% for SVC, and 76.7% for LR. The study by Vinitnantharat (2023) did not use any tuning or parameter setting in running the model.

Still from Kaggle kernels, the report by Slimbensalah (2023) utilizes LR, DT, and RF model to evaluate the dataset. Utilizing R language, several binary variables such as *repeated_guest* are converted into Boolean data type. The models achieved final accuracy values ranging from 80% for LR, 82.6% for DT and 90.6% for RF. However, the report does not provide any class balancing, normalization, or data tuning processes.

The three Kaggle referenced reports above will be used as a benchmark towards the models designed in this assignment. To make improvements towards the models executed in the algorithms above, relevant preprocessing and data preparation steps such as class balancing and normalization are adopted. As from the three references, the Logistic regression model gives the lowest performance among other algorithms, thus the LR model is not chosen. Conversely, as Random Forest proved to be a good model in multiple studies, this model along with Decision Tree (DT) are chosen as the models to be constructed.

Research was also done regarding hotel booking cancelations using 4-star hotels data in Greece. Timamopoulos (2020) collected such as room rate, number of guests, room type, day / date of arrival, total hotel income (based on number of stay days) as the input variables from the modelling process. As the data used in the study is imbalanced, the study utilizes oversampling and SMOTE method to balance the data; and put a priority on Precision and Recall values to measure the performance of the models. The result of the study achieved an accuracy level of up to 99% with DT, gradient boosting, and XGBoost, with similarly high F1, recall, and precision values.

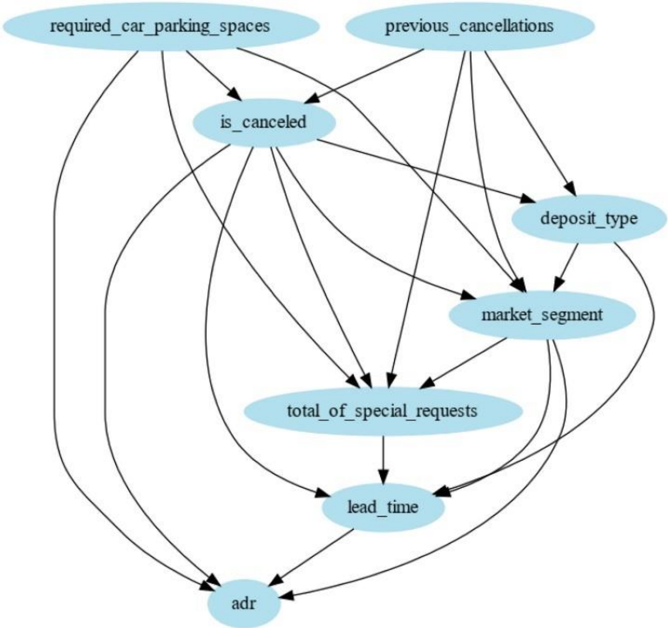

Figure 1. Variable selection with Importance Ranking (Chen et al., 2023)

Apart from the more traditional models, another hotel booking prediction model was done by Chen et al. (2023). The study proposed an integration of Bayesian networks (BN) and lasso regression / ANN model to predict the booking outcomes. Also utilizing the same data source as the study by Satu et al. (2021), the study uses variations of important variables as input for the model; yielding an accuracy measure of 81.8% utilizing the combination of BN-Lasso, and 83% with BN- ANN. One of the variables combinations selected is as presented in Figure 1. The study concludes that utilizing a hybrid of BN with machine learning models is effective in increasing the performance of the model.

2.2 Machine Learning Models

In machine learning, multiple algorithms exist, varying in their base concepts and field of use. Generally, machine learning models are divided into supervised and unsupervised learning. Supervised learning consists of models that need external assistance in order to solve a problem, while unsupervised learning relies on its own ability in discovering hidden patterns in the dataset (Bonaccorso, 2018). In its utilization, the algorithms employed would also yield different performance based on the model fit and number of variables in the dataset.

As the booking cancellation detection model requires input variables to be trained and able to execute the prediction, the models discussed in this section would cover the supervised learning algorithms that are utilized in this report. The models described in this section includes Decision Tree (DT), Random Forest (RF), and Support Vector Machines (SVM).

Decision Tree (DT)

Decision tree is an algorithm based on the tree structure, using a sequential decision-making process. The decision tree processes information starting from the root, where a feature is evaluated and branching out to a certain number of branches. As an algorithm initial based on choosing the right path, decision trees are originally designed to handle categorical variables, before developing to be capable of processing numerical variables as well (Bonaccorso, 2018). Aligning with its origin however, an evaluation between DT utilization as a classifier and a regressor shows better results in its classifying capability (Singh Kushwah et al., 2022). Compared to other algorithms, the decision tree has a simpler structure, making it rather fast in making predictions from a dataset. Its nature of being unaffected by values assumed in each feature also allows DT to perform efficiently with un-normalized datasets (Bonaccorso, 2018).

In its process, DT processes data by repeatedly sorting and inputting data into groups according to class labels. The mechanism involves data impurity indexes such as Gini impurity, entropy, or information gain (Li et al., 2021). These impurities determine the split of each node and are determined for every child node and is continued towards the final node is generated ((Lee et al., 2022).

Random Forest (RF)

Random Forest classifier is an ensemble algorithm that was developed from decision tree. Consisting of a set of decision trees, the RF model splits the data into several subsets to be trained in different trees. This method allows many trees to be trained in a weaker way, however with each being more specialized in a portion of the sample. The interpretation of the model is then done by combining the decision of these trees through majority vote or averaging (Bonaccorso, 2018).

Random Forest is also equipped with an *importance* feature, allowing for easy evaluation to see which input variables are the most important in contributing to the target variable decision (Breiman & Cutler, 2001). In this way, factor analysis and assessments are also made possible for simpler decision-making process (Bonaccorso, 2018).

Support Vector Machines (SVM)

Similar with the previous two algorithms, SVM are supervised learning models utilized for classification and regression analysis. Besides performing linear classifications, SVM are equipped with kernel mapping features to perform non-linear classifications by mapping the inputs into high dimensional spaces, drawing margins between the classes (Mahesh, 2018). Its capability to process multiple scenarios allow SVM to achieve high performance in different applications, and thus becoming a preferred choice for tasks with complicated separation of hyperplane (Bonaccorso, 2018). Figure 3 shows a mechanism of SVM with two support vectors.

In the field of hospitality and hotel research, SVM also has been used in multiple studies such as Sánchez et al. (2020) and Yuxuan et al. (2023) on identifying hotel booking cancellations. As such, SVM is the third chosen model utilized for model building in this report.

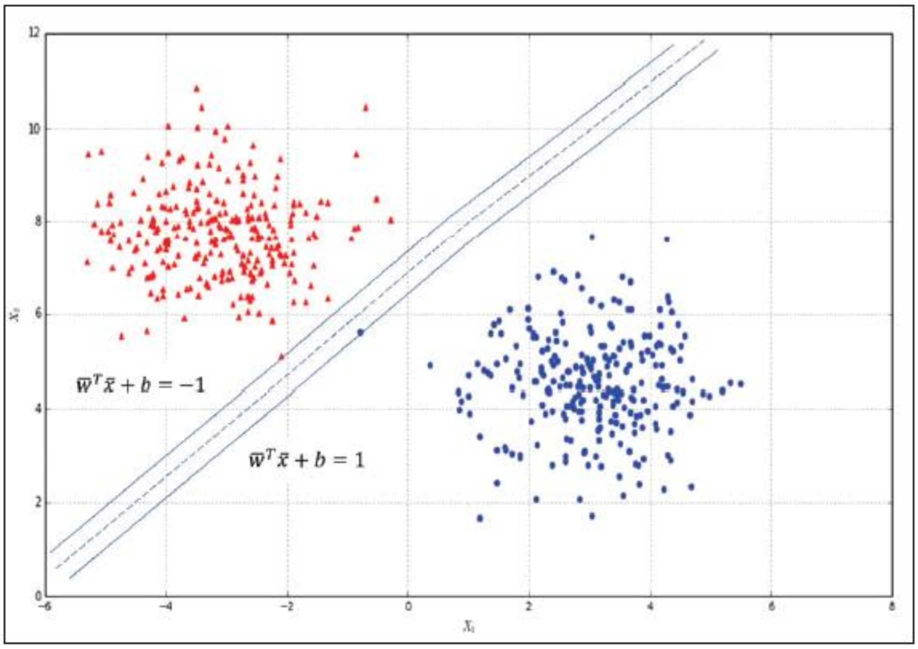

Figure 3. Support Vector Machine Graph with 2 Support Vectors

The summary of related works is summarized in Table 1. The table listed only the works with the aim of creating hotel prediction models that acts as benchmark towards this study model building and evaluation.

Table 1. Literature Review Matrix

No.	Title and Author(s)	Data Source	Method	Results
1	Performance Analysis of Machine Learning Techniques to Predict Hotel booking Cancellations in Hospitality Industry *Satu et al. (2021)*	1 resort + 1 city hotel, generated by Antonio et al. (2019b)	GB, RF, XGB, DT, LR, KNN, GNB With variations of feature selection (CFS, GRAE, IGAE)	CFS top 3 accuracy: GB, XGB, RF & DT GRAE top 3: XGB, GB, LR IGAE top 3: XGB,GB, LR

(Cont.)

Table 1. Literature Review Matrix (Cont.)

No.	**Title and Author(s)**	**Data Source**	**Method**	**Results**
2	Predicting hotel booking cancellations to decrease uncertainty and increase revenue *Antonio et al. (2017)*	4 Resort Portuguese hotels	Boosted DT, Decision Forest, Decision Jungle, Locally Deep SVM, NN	Separated results for each of the 4 hotels, best model oberall: Boosted DT, Decision Forest.
3	Big Data in Hotel Revenue Management: Exploring Cancellation Drivers to Gain Insights Into Booking Cancellation Behavior *Antonio et al. (2019a)*	4 City & 4 Resort Portuguese hotels databases (PMS) + National holidays + special events + weather records + online review reputation	Models built on different feature selection & no of observation: 1. PMS features (01/2016-11/2017) 2. PMS features (08/2016-11/2017) 3. PMS + additional data 4. Optimized model for C1 & R1 (city & resort hotel)	Model 3 & 4 perform best, with improved robustness from previous study by Antonio et al. (2017) Identification of important features contributes to higher model accuracy
4	Modeling-Tuning-ExplainableAI-EDA-PandasProfiling | Kaggle *Sekeroglu (2023)*	Kaggle	KNN, RF, LR with 10-fold cross validation	RF highest acc: 88% LR & KNN: 80%. Tuning (GridSearchCV) RF model generate 87% accuracy Lowest accuracy: LR at 79.9%
5	Hotel reservation classification | Kaggle Vinitnantharat (2023)	Kaggle	DT, RF, KNN, SVC, LR	Order of Accuracy from highest: RF 85% DT 82% LR 76%
6	RMarkdown - Booking Classification EDA + Modeling | Kaggle Slimbensalah (2023)	Kaggle	LR, DT, RF	RF highest Acc at 90%, followed by DT 83% and LR 80%
7	Anomaly Detection: Predicting hotel booking cancellations Timamopoulos (2020)	4-star hotel in Greece	LR, NB, KNN, SVM, DT, RF, GB, XGB with 10-fold cross validation	Best accuracy: DT & GB With DT performing 5x faster than 2nd best GB
8	Prediction of hotel booking cancellations: Integration of machine learning and probability model based on interpretable feature interaction Chen et al. (2023)	Comp study 1: 85274 entries from 2 Hotels – Portugal Comp study 2: Restricted data from travel service company	Calibrate Bayesian Network feature selection on probability model Lasso/ ANN-original Lasso/ ANN-original-bayesian Lasso/ ANN -ori-bayesian-interact Comparison with XGB	Proposed integration model has better predicition performance, with obtained BN estimators the most important predictors

3.0 METHODS AND IMPLEMENTATION

In this section, the data processing and model implementation using the R language are discussed. Before delving further into analysis of the data, the details of the Hotel Reservation Classification dataset are observed. The dataset was derived from Kaggle, consisting of customer hotel booking information such as number of people staying, the duration of stay, and room prices. The details of each feature in the dataset along with each data label are listed in Table 2.

Table 2. Hotel Reservation Dataset Details

No.	Variable Name	Data type	Contents
1	Booking_ID	chr	Unique values for each customer booking
2	no_of_adults	int	Number of adults staying (0-4)
3	no_of_children	int	Number of children staying (0-10)
4	no_of_weekend_nights	int	Number of weekend nights to stay (0-7)
5	no_of_week_nights	int	Number of week nights to stay (0-17)
6	type_of_meal_plan	chr	Type of meal plan choosen (Plan 1, Plan 2, Plan 3, Not chosen)
7	required_car_parking_space	int	Whether the customer need parking space (0: No, 1: Yes)
8	room_type_reserved	chr	The room type reserved during stay (Room type 1-7)
9	lead_time	int	Difference between date of stay and date of booking (0-443)
10	arrival_year	int	Year of arrival (2017, 2018)
11	arrival_month	int	Month of arrival (1-12)
12	arrival_date	int	Date of arrival (1-31)
13	market_segment_type	chr	Market segment classification (Aviation, Complementary, Corporate, Offline, Online)
14	repeated_guest	int	Whether guest is a repeat guest (0: No, 1: Yes)
15	no_of_previous_cancellations	int	Num of booking canceled before (0-13)
16	no_of_previous_bookings_not_canceled	int	Num of booking canceled before (0-58)
17	avg_price_per_room	num	Price of room (Decimal values 0 - 540)
18	no_of_special_requests	int	Num of special request by guest (0-5)
19	booking_status	chr	Target Variable (Canceled, Not_canceled)

Using the skim() function from package Skimr in R, the summary of the dataset is also generated in RStudio, as stated in Figure 4. The data consists of 36275 observations and 19 variables; where 14 are numeric variables and 5 are character variables. As stated in the details from Table 2, from the numerical variables itself there are two binary variables consisting of only 0s and 1s.

```
> skim(data)
── Data Summary ────────────────────────
                           Values
Name                       data
Number of rows             36275
Number of columns          19
_______________________
Column type frequency:
  character                5
  numeric                  14
________________________
Group variables            None

── Variable type: character ────────────────────────────────────────
  skim_variable        n_missing complete_rate min max empty n_unique whitespace
1 Booking_ID                   0             1   8   8     0    36275          0
2 type_of_meal_plan            0             1  11  12     0        4          0
3 room_type_reserved           0             1  11  11     0        7          0
4 market_segment_type          0             1   6  13     0        5          0
5 booking_status               0             1   8  12     0        2          0

── Variable type: numeric ──────────────────────────────────────────
   skim_variable                        n_missing complete_rate     mean     sd   p0  p25   p50  p75 p100 hist
 1 no_of_adults                                 0             1   1.84    0.519    0    2     2    2    4
 2 no_of_children                               0             1   0.105   0.403    0    0     0    0   10
 3 no_of_weekend_nights                         0             1   0.811   0.871    0    0     1    2    7
 4 no_of_week_nights                            0             1   2.20    1.41     0    1     2    3   17
 5 required_car_parking_space                   0             1   0.0310  0.173    0    0     0    0    1
 6 lead_time                                    0             1  85.2    85.9      0   17    57  126  443
 7 arrival_year                                 0             1 2018.     0.384 2017 2018  2018 2018 2018
 8 arrival_month                                0             1   7.42    3.07     1    5     8   10   12
 9 arrival_date                                 0             1  15.6     8.74     1    8    16   23   31
10 repeated_guest                               0             1   0.0256  0.158    0    0     0    0    1
11 no_of_previous_cancellations                 0             1   0.0233  0.368    0    0     0    0   13
12 no_of_previous_bookings_not_canceled         0             1   0.153   1.75     0    0     0    0   58
13 avg_price_per_room                           0             1 103.     35.1      0   80.3  99.4 120  540
14 no_of_special_requests                       0             1   0.620   0.786    0    0     0    1    5
>
```

Figure 4. Dataset Summary - skim() function

Utilizing the data above, the Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM) models are to be implemented on the model. The data first go through cleaning and transformation processes before then is split into training and testing data for the implementation and validation phase. The overall stages of the model building are shown in Figure 5. These models are chosen due to their proven superiority in classification modelling, with multiple studies related to hotel booking cancellations utilizing similar models in their research as described in the related works section.

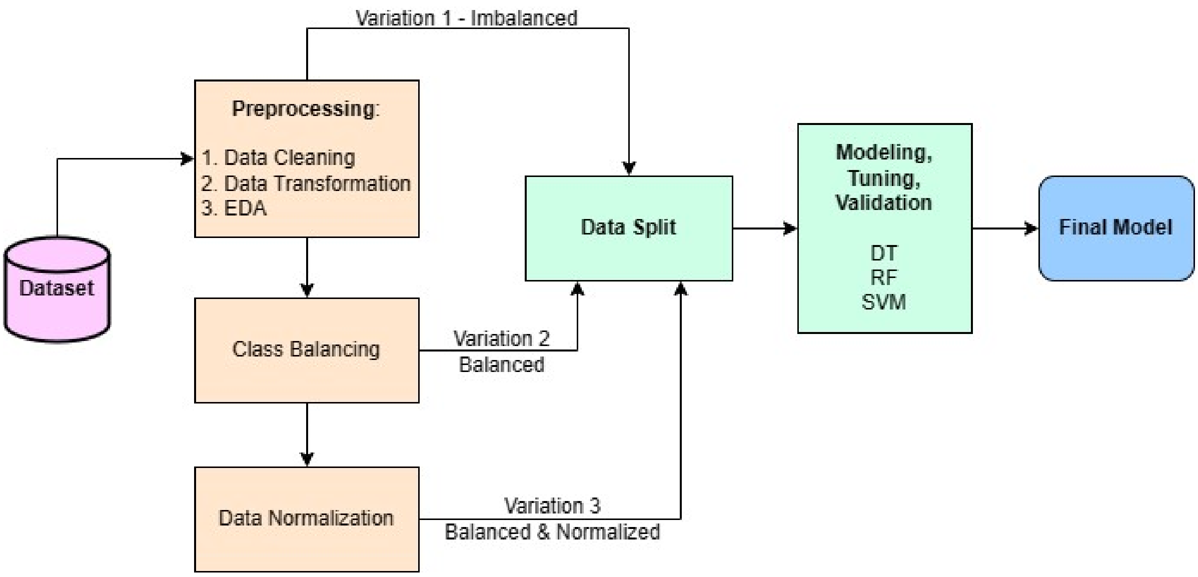

Figure 5. Model Building Framework

DT from its roots is particularly suited for classification. It also has a simple structure that allows for fast training process, which is suitable to process large datasets in a shorter amount of time versus other models. RF, a machine learning model developed from DT, is more complex as it utilizes multiple number of trees in its decision-making process. The RF model also proved to be one of the best models in analyzing classification problems, thus the adoption of the method. Lastly, the support vector machine is chosen. Also a supervised learning algorithm dedicated as a classification and regression tool, the algorithm is capable to process both linear and non linear classifications. This allows the model to exhibit high performance in many different applications.

In this study, the performance metrics observed includes the Accuracy, F1, Precision, and Recall values, Representations of ROC and AUC are also included for several of the models. The Accuracy represents how well the model is able to correctly predict the prediction outcomes. Precision is the measure of accuracy in making positive predictions, while Recall measures the completeness of positive predictions. F1 gives the weighted harmonic mean of Precision and Recall. All performance metrics have 1.0 as the highest value and 0 the lowest.

3.1 Data Preparation

This section consists of data cleaning, transformation, and data exploration procedures. All data manipulation processes are conducted using R language in RStudio.

Data Cleaning

From the data skim in Figure 4, it can be seen that there are no missing or duplicate values in the dataset. Checking process done using the colSums(is.na()) and duplicated() functions also shows similar results. Therefore, imputation process is not needed. In the case where missing values are present in the dataset, imputation can be done with several R packages such as using the PreProcess function from Caret package for imputing continuous variables and imputing values with the mode of the observations for categorical variables. Apart from these methods, packages such as Mice and MissForest are also feasible alternatives.

Data Transformation

Next, data transformation process is executed. Firstly, the booking_id column which consists of all unique variables are removed. Next, the target variable booking_status is changed into booking_canceled, with its contents transformed into Boolean. 'canceled' is changed to True while 'not canceled' is converted to False.

```
> summary(as.factor(data$type_of_meal_plan))
 Meal Plan 1  Meal Plan 2  Meal Plan 3 Not Selected
       27835         3305            5         5130
> summary(as.factor(data$room_type_reserved))
Room_Type 1 Room_Type 2 Room_Type 3 Room_Type 4
      28130         692           7        6057
Room_Type 5 Room_Type 6 Room_Type 7
        265         966         158
> summary(as.factor(data$market_segment_type))
    Aviation Complementary     Corporate
         125           391          2017
     Offline        Online
       10528         23214
```

```
> summary(data4$type_of_meal_plan)
    0     1     2     3
 5129 27802  3302     5
> summary(data4$room_type_reserved)
    1     2     3     4     5     6     7
28105   692     7  6049   263   964   158
> summary(data4$market_segment_type)
    1     2     3     4     5
  125   390  2011 10518 23194
```

Figure 6. *Meal Plan*, *Room Type*, & *Market Segment* Variable Transformation, Before (Left) & After (Right)

The character variables type_of_meal_plan and room_type_reserved is modified in values to be represented as integers. E.g. Room_Type 1 is changed to int value 1 and Meal Plan 2 is changed to int value 2. Entries with type_of_meal_plan Not Selected is changed to 0. The character factor variable market_segment_type is also converted to factor and represented as numerical values 1-5. The before and after transformation for these variables are shown in Figure 6.

```
#date format
data3 <- data.frame(
  data2[, 1:11],
  date = as.Date(paste(data2$arrival_date,
                       data2$arrival_month,
                       data2$arrival_year, sep="-"),
                 format = "%d-%m-%Y"),
  data2[, 12:ncol(data2)]
)
#check missing date
summary(data3$date)
data3 = data3[complete.cases(data3$date), ]
```

Figure 7. *Date* variable creation

Next, data transformation is done on the arrival year, month, and date variable. As the combination of these three columns represents a date format, therefore a new variable *date* is created to represent a date data type variable. To ensure the combination column has no missing values due to inconsistent values, a complete.cases() function is used (Figure 7). The inconsistent values here refers to possibility of false combination of date and month; such as the 31st of February. The complete.cases() filter here eliminates 36 observations, thus the dataset at this point consists of 36238 entries.

Exploratory Data Analysis (EDA)

To get an overview of the data, Exploratory Data Analysis (EDA) is then performed on the 36238 entries. Firstly, Figure 8 displays the demography of the target variable *booking_canceled*. This data shows that the dataset is imbalanced, thus class balancing is to be considered in model building and evaluation.

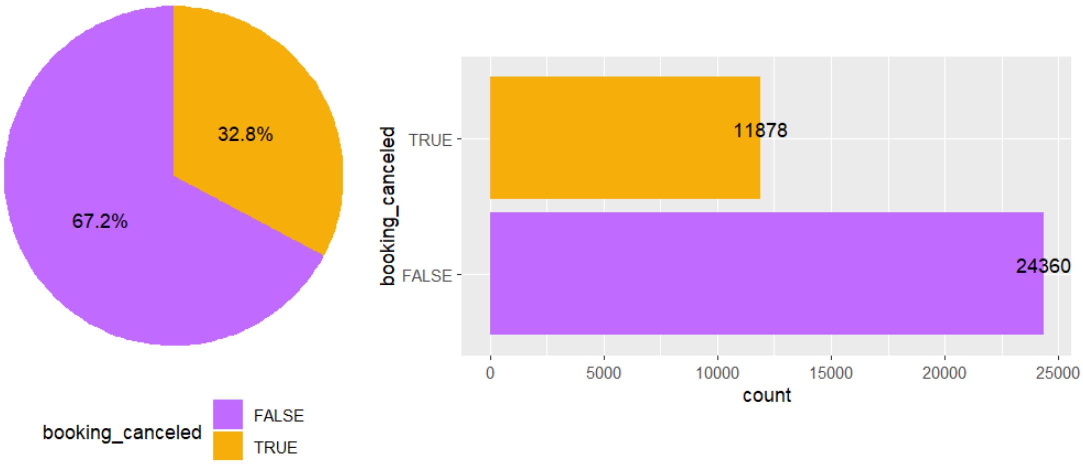

Figure 8. *booking_canceled* Distribution

Then, Figure 9 displays the demography of factor variables including the multilevel variables *type_of_meal_plan*, *room_type_reserved*, *market_segment_type*, and the 2 level factor variables *required_car_parking_space* and *repeated_guest*.

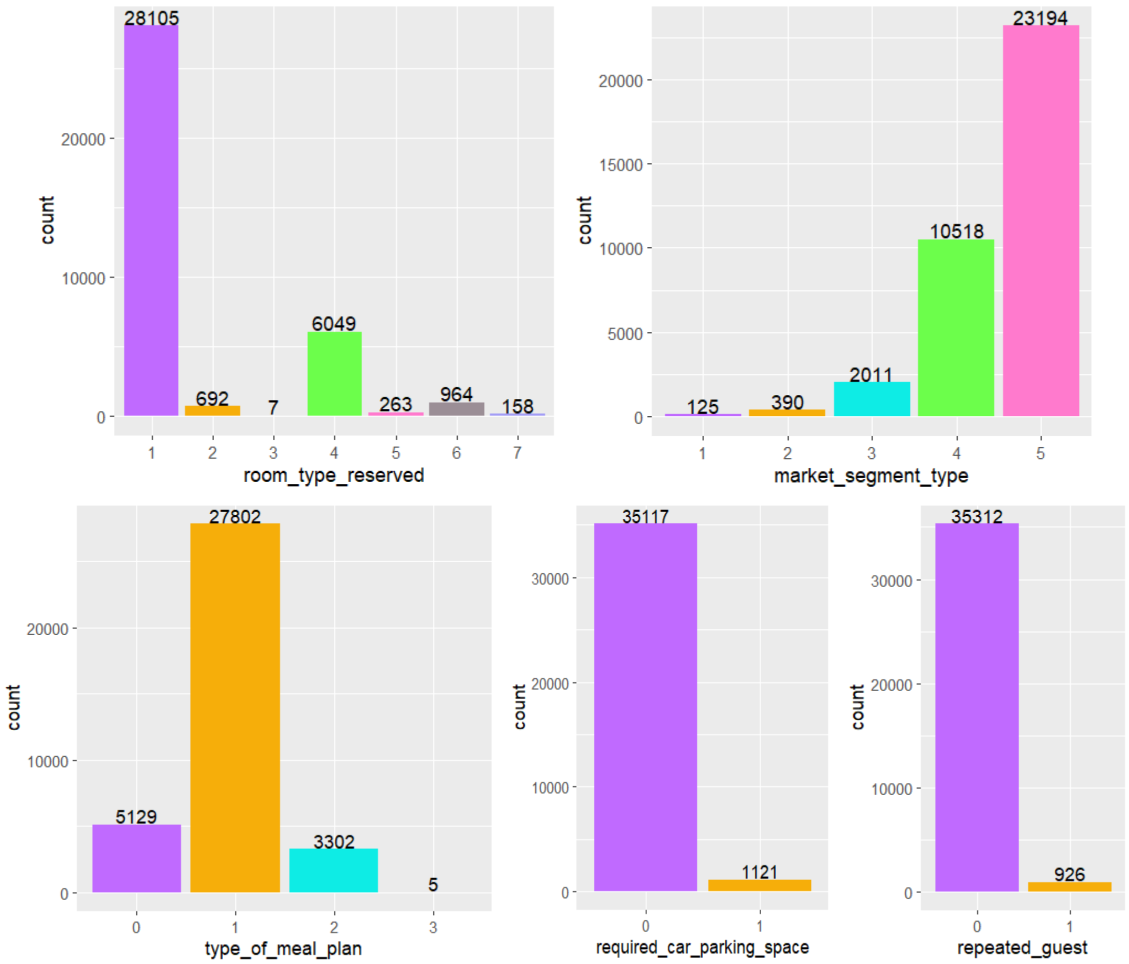

Figure 9. Bar Graph for Factor Variables

Figure 10 displays the correlation matrix between all variables in the dataset. In relation to the target variable *booking_canceled*, *lead_time* shows the strongest positive correlation. As 1 = canceled and 0 = not canceled, therefore a rise in lead time leads to higher probability of a booking being canceled. Other variables that shows significant positive correlation with the target variable are *arrival_year*, *market_segment_type*, and *avg_price_per_room*. Variable with notable negative correlation is *no_of_special_requests*; with cancelations becoming lower with more special requests made by the customer.

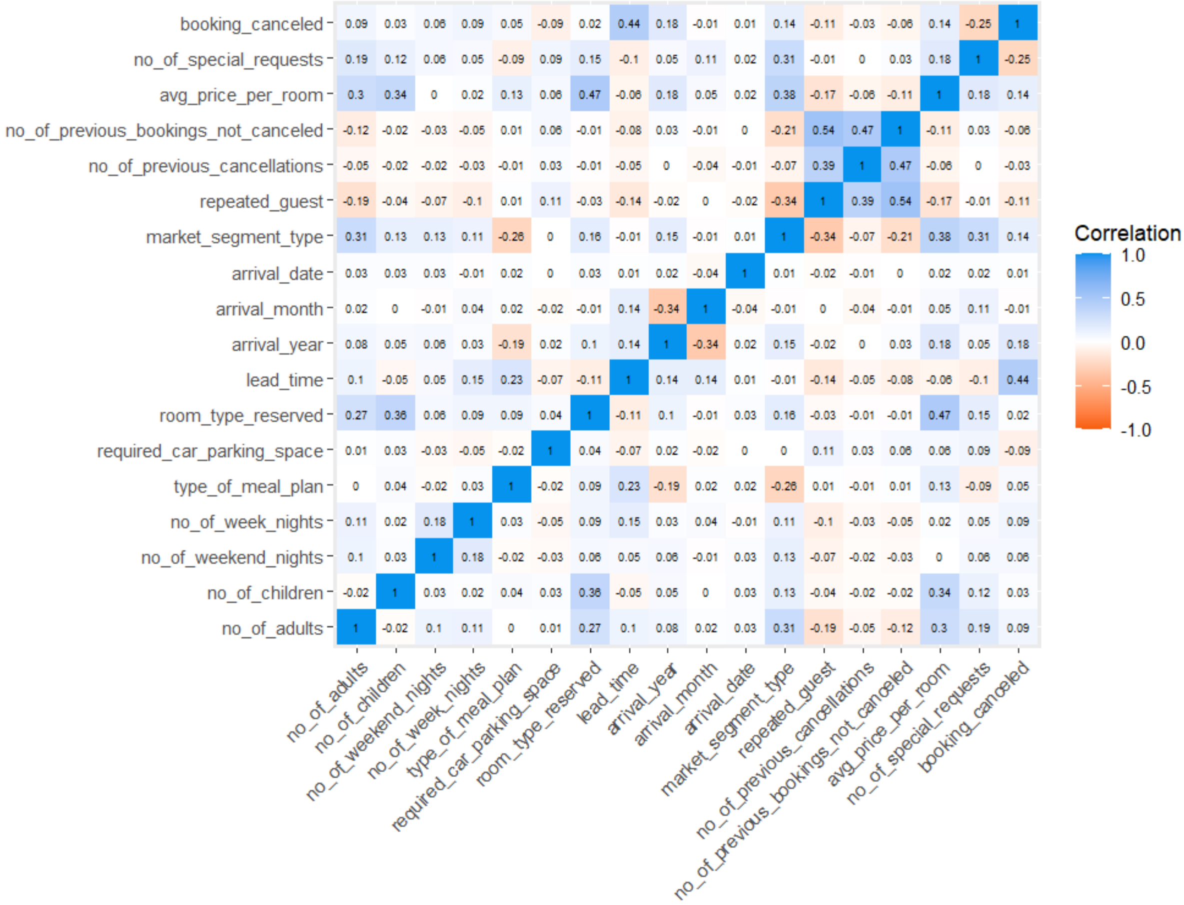

Figure 10. Correlation Matrix

Zooming in on the Lead time variable, Figure 11 presents the variation of lead time with the number of bookings canceled. The data shows that most of the data with not cancelled status came from shorter lead times, with higher lead times showing higher cancelations by the 150 days mark. The graph also give insight that most of the hotel bookings are done within 50 days lead time period, with the booking number gradually decreasing as lead time increases.

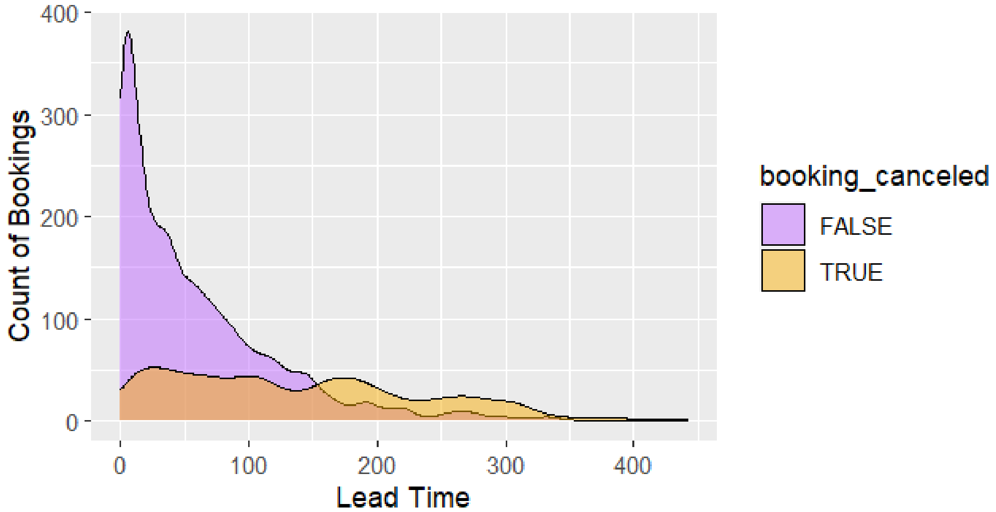

Figure 11. Lead Time to Booking Outcome

Figure 12 presents an overview of room prices correlates with lead time and how prices may impact the booking outcome. The scatterplot between lead time and price shows that there is a notable portion of very low prices which deviates from the normal distribution of prices, which mostly ranges from 70-250. The scatterplot also displayed that as lead time decreases, prices tend to be higher, while bookings made longer from the check in date serves more lower prices. The line plot on the right side shows how prices differ for bookings that are cancelled and those that are not. Although in general the lines are similar, there is a spike on the left-most side of the not canceled booking outcome *False* which may represent bookings made with high discounts or from event promotions. This spike shows that bookings with the lowest prices are more likely to not be cancelled, despite similarity in cancellation trend when prices are within normal range.

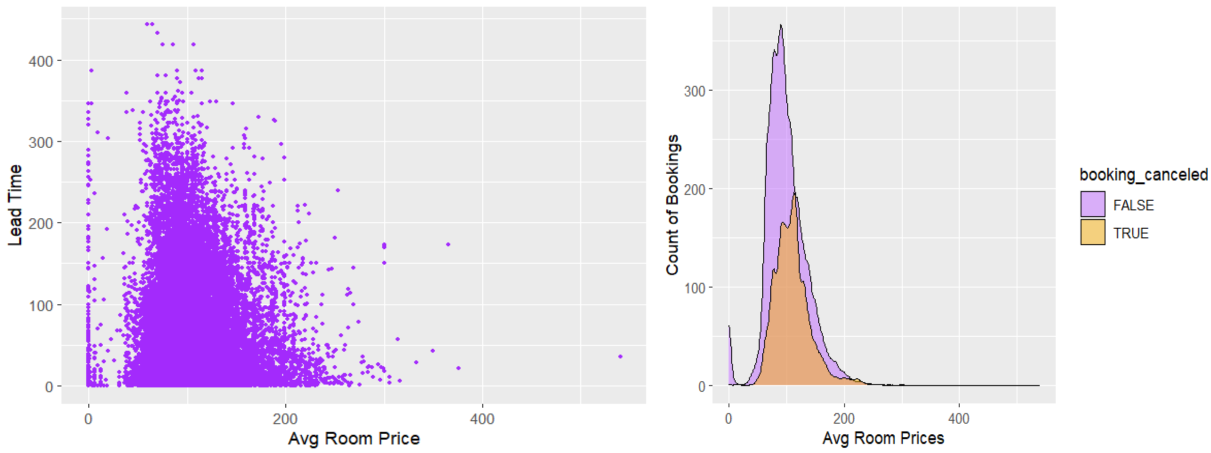

Figure 12. Room Prices to Lead Time and Booking Outcome

Figure 13 shows the distribution of number of children and adults of the hotel bookings. The children bar plot shows that mostly a booking is made with no children, with the most frequent amount for those with children is between one and two. Meanwhile for the number of adults, the most general count is at two adults, with solo travelers on second place and those with three adults at third place.

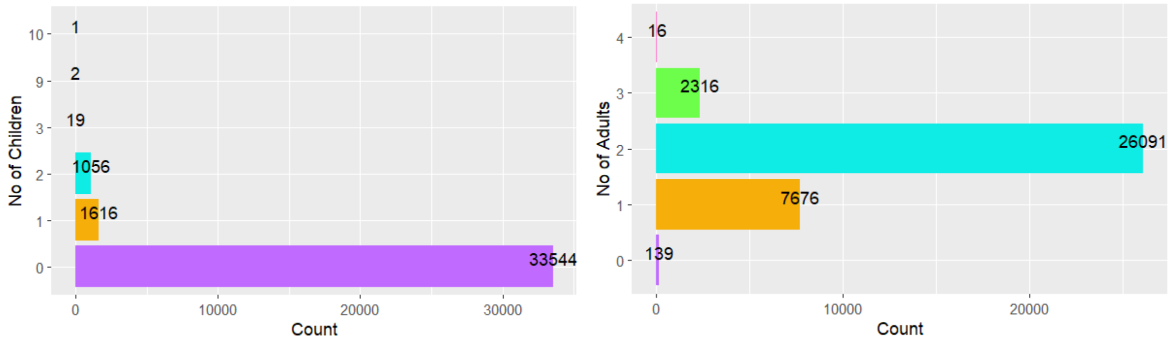

Figure 13. No of Children and Adults Distribution

Lastlly, the distribution of bookings and prices are presented in a monthly format in Figure 14. The upper graph shows that bookings are mostly done for arrivals on the month of October, with September and August on second and third place. The lowest arrivals came from the earlier period of the year from January to March. From the lower scatter and line graph, it can be seen that number of bookings grow over time, with significant increase in year 2018. Average room prices also increase gradually, with notable trend of rising prices from the month 7 to 10 (July – Oct), which aligns with the top contributor of arrival month that occurs in the same period of the year.

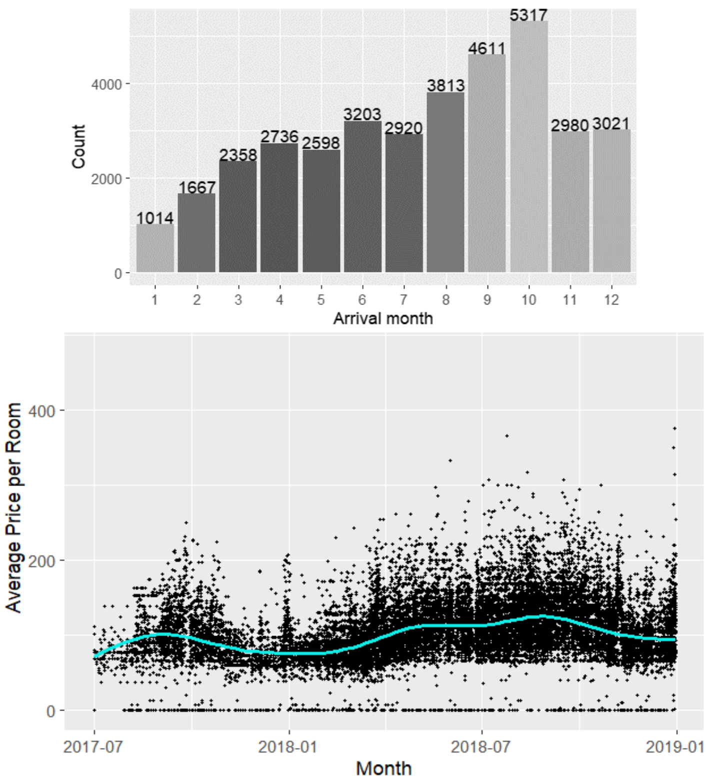

Figure 14. Arrival Month to Average Room Prices

3.2 Model Implementation and Validation

This study implemented three machine learning models, including Decision Tree (DT), Random Forest (RF), and Support Vector Machines (SVM). To prepare for the training phase, the data is split in 70:30 ratio for training and testing. This split considers the relatively high amount of data, at >30k entries in total. The split is done utilizing createDataPartition function with Caret package. The train and test set are then assigned to each respective data frame. The split process is as presented in Figure 15.

```
#split data
library(caret)
set.seed(77)
for_training <- createDataPartition(data4$booking_canceled, p = 0.7, list = FALSE)
train_set <- data4[for_training, ]
test_set <- data4[-for_training, ]
```

Figure 15. Data Split Process

After the data is split to train and test set, the model implementation is done. In order to analyze different variations of the models, implementation and validation is done in three different phases, in the order of:

1. Using the imbalanced data without class balancing procedures,
2. Using balanced data, including variations of tuning procedures, and
3. Using balanced data with numerical variables normalized.

Firstly, the dataset is trained without class balancing procedures. Then, to compare with the imbalanced data model results, the dataset are balanced and training is conducted once more. In the third scenario, the balanced dataset is normalized before training to ensure the algorithms are training the variables with equal measure. These variations are done to find out to what degree these pre-training preparations affect the training result and model performance.

Imbalanced Data

With the initial model implementation, the algorithms are run with no tuning parameters, only including the mandatory target variable *booking_canceled* and training set as input for the training phase. The model building codes are as displayed in Figure 16. The packages used to train the models are the randomForest, rpart & rpart.plot and e1071 package respectively for RF, DT, and SVM models.

```
#Random Forest Model
library(randomForest)

mod_rf = randomForest(booking_canceled ~ ., data = train_set)
rf_predict = predict(mod_rf, newdata = test_set)

confusionMatrix(rf_predict, test_set$booking_canceled)

#Decision Tree Model
library(rpart)
library(rpart.plot)
mod_dt = rpart(booking_canceled ~ ., data = train_set, method = "class")
dt_predict = predict(mod_dt, newdata = test_set, type = "class")

# class method & type - model for classification
confusionMatrix(dt_predict, test_set$booking_canceled)


#Support Vector Machine
library(e1071)
mod_svm = svm(booking_canceled ~ ., data = train_set)
svm_predict = predict(mod_svm, newdata = test_set)

confusionMatrix(svm_predict, test_set$booking_canceled)
```

Figure 16. Imbalanced Data Model Building

The DT, RF, and SVM models are then validated in performance using the test set as input, with confusion matrix generated for each model. With the imbalanced dataset, the accuracy of each model is 82.76%, 90.24%, and 83.4% for DT, RF, and SVM respectively. Table 3 summarizes the performance metrics of each model.

Table 3. Imbalanced Model Performance Metrics

No.	Model	Accuracy	F1	Precision	Recall	Balanced Accuracy
1	RF	0.902401	0.929016	0.908889	0.950055	0.877357
2	SVM	0.833962	0.881818	0.845449	0.921456	0.78798
3	DT	0.827615	0.876499	0.845411	0.909962	0.784338

As stated in the table, the Random Forest model performs best, with accuracy reaching 90.2%, while Decision Tree gives the lowest metrics at only 82.76%. The F1, Precision, and Recall values ranking also aligns with accuracy value, with RF being far superior to the other two algorithms. The ROC graph and AUC values at Figure 17 also show similar results, with random forest giving the least amount of false positive and false negatives.

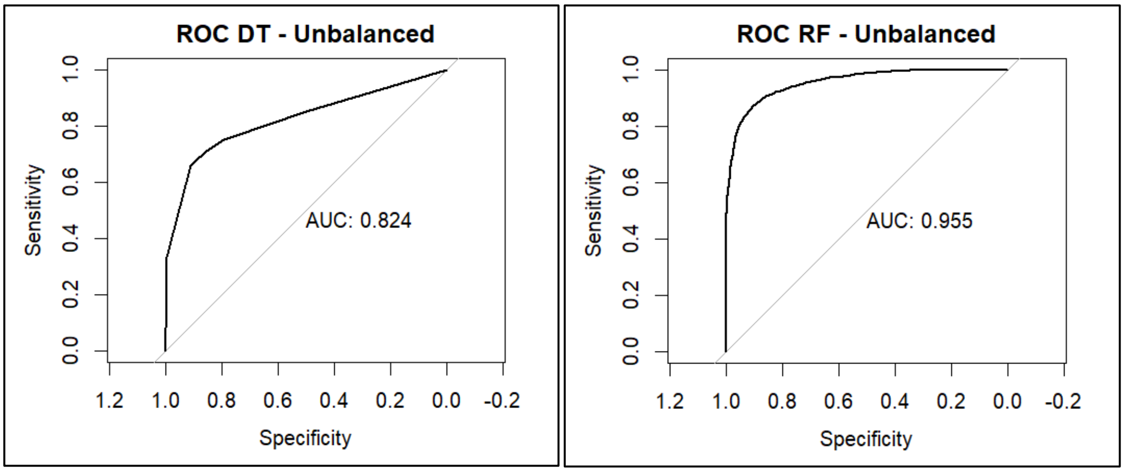

Figure 17. ROC and AUC – Imbalanced Data

Balanced Data

To give a more balanced results in the model implementation and act as a validation process, class balancing is done using the Synthetic Minority Oversampling Technique (SMOTE) technique with SmoteClassif function from UBL package. With the package, the parameter C.perc is set to 'balanced'. The result of the class balancing makes the data stands at 29997 observations, with a 60:40 ratio on majority class at 18119 entries and the minority at 11878 entries. This allows a reduced imbalance in the dataset. As the class balancing is done prior to any data transformation procedures, then the data transformation as explained in the data preparation section is repeated prior to training.

After data transformation is done, similar training procedures are done using the 29997 entries. The data is split to 70:30 ratio to maintain a similar benchmark with the imbalanced data, and the DT, RF, and SVM models are run. With the balanced dataset, the overall accuracy of the models dropped very slightly in value, with DT model being the most affected. The performance details of the balanced dataset is presented in Table 4. However, the Balanced Accuracy value of the RF model increased by 1% and SVM model increase by a significant 3%. Only the balanced accuracy of DT does not align with the performance of other models, declining by a small increment of 0.1%. The results also show that the RF model shows overfitting of the model, while the SVM and DT models show the model is not

Table 4. Balanced Model Performance Metrics

Data Set		Train	Test				
No.	**Model**	**Accuracy**	**Accuracy**	**F1**	**Precision**	**Recall**	**BalancedAcc**
1	RF Balanced	0.9831	0.893532	0.913051	0.900949	0.925483	0.885138
2	SVM Balanced	0.8345	0.825183	0.856596	0.848934	0.864397	0.814882
3	DT Balanced	0.8088	0.80429	0.845404	0.808429	0.885925	0.782844

The ROC and AUC value of the balanced DT and RF models also decline versus the imbalanced dataset model, align with how the overall Accuracy, Precision, and Recall values changed. The ROC graph for balanced dataset is presented in Figure 18.

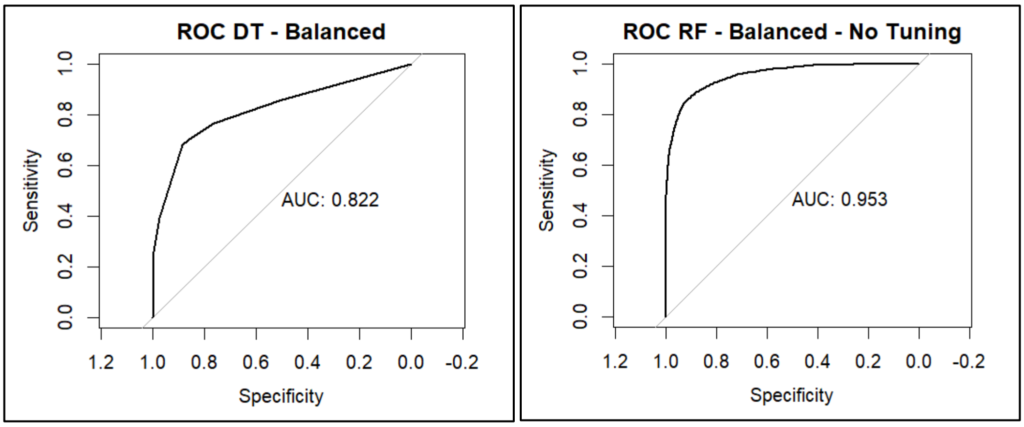

Figure 18. ROC and AUC – Balanced Data – No Tuning

After the initial model implementation is done, optimization - tuning procedures are experimented on all three models to see how the model performances may improve if modifications are given to the default model settings. The tuning procedures are first explained for the DT model, continued with the SVM, and lastly the RF model.

Table 5. Tuning Variations on Balanced Decision Tree

Tune No.	**CP**	**Train Accuracy**	**Test Accuracy**	**Balanced Accuracy**	**Result**
Original	-	0.8088	0.80429	0.782844	*[Benchmark]*
1	0.1	-	0.7261	0.6762	Decline
2	0.05	-	0.7864	0.7755	Decline
3	0.01	-	0.8043	0.7828	Similar
4	0.005	0.8332	0.8179	0.8099	Improve
5	**0.001**	**0.8632**	**0.8537**	**0.8398**	**Improve**

For the Decision Tree model, the tuning parameters chosen includes the *cp* value, which stands for *complexity parameter*. The tuning is done with *cp* value variation of 0.001, 0.005, 0.01, 0.05, and 0.1. The tuning result shows that the model shows the best result with *cp* value of 0.001, where the accuracy is successfully increased by 6% from the initial model. The tree structure comparison with the initial model also show increased complexity on how the model evaluates prediction for each instance of the data (Figure 19). Validation with the training set also shows similar accuracy level between the training and test set, showing no tendency of overfitting, as the numbers stated in Table 5.

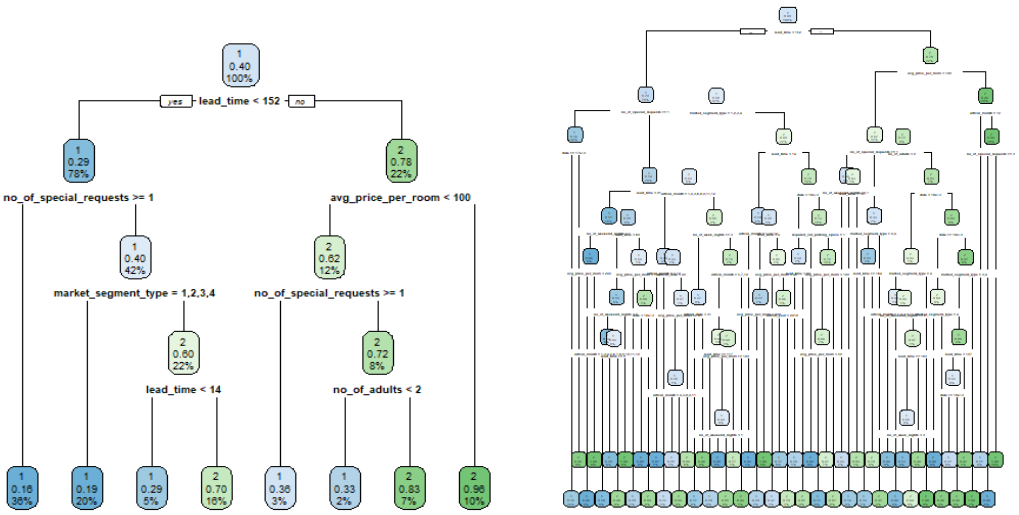

Figure 19. DT Tree Structure Before (Left) and After (Right) Tuning

For the SVM model, the tuning procedures is done with the *epsilon* and *cost* parameters. Epsilon determines the level of error allowed in the model, while cost shows the trade-off between accuracy and complexity. A lower *epsilon* and higher *cost* value will result in a more accurate model, but would be more prone to overfitting. Table 6 shows the tuning variations for SVM model. The result shows that the epsilon tuning does not show significant difference in performance, thus the tuning is focused on the *cost* parameter. Both *cost* = 20 and 40 shows the best accuracy results for the model. However as change in test accuracy between the two model is not significant, the *cost = 20* is chosen as the best SVM tuned model. This is due to the validation process with the train set data shows a larger improvement in training accuracy, which indicates the model is starting to overfit.

Table 6. Tuning Variations on Balanced Support Vector Machine

Tune No.	**Epsilon**	**Cost**	**Train Accuracy**	**Test Accuracy**	**Balanced Accuracy**	**Result**
Original	-	-	0.8345	0.825183	0.814882	*[Benchmark]*
1	0	2	-	0.8294	0.8207	Improve
2	0	4	0.8448	0.8306	0.8224	Improve
3	1	4	0.8448	0.8306	0.8224	Improve
4	2	4	0.8448	0.8306	0.8224	Improve
5	0	8	0.8507	0.8360	0.8272	Improve
6	0	12	0.8539	0.8373	0.8288	Improve
7	**0**	**20**	**0.8589**	**0.8403**	**0.8320**	**Improve**
8	0	40	0.8661	0.8443	0.8359	Improve

Lastly, the tuning for RF model is executed. In tuning the model, firstly the parameters of the original model are checked by manually calling the parameter of the generated model:

```
> mod_rf_bal$ntree
[1] 500
> mod_rf_bal$mtry
[1] 4
> mod_rf_bal$maxnodes
NULL
> mod_rf_bal$nodesize
NULL
> mod_rf_bal$sampsize
NULL
> mod_rf_bal$replace
NULL
```

```
#tuning
mod_rftuned_bal = randomForest(booking_canceled ~ ., data = train_set_balanced,
                               ntree = 600,
                               mtry = 6,
                               nodesize = 2,
                               importance = TRUE
                               )
rftuned_predict_bal = predict(mod_rftuned_bal, newdata = test_set_balanced)

confusionMatrix(rftuned_predict_bal, test_set_balanced$booking_canceled)
```

Figure 20. RF Tuning

Then, variations of tuning parameters are executed to validate the model results and find the best most satisfactory results. In the tuning process, the parameter Ntree, Mtry, Importance, Nodesize, and Maxnodes are involved. Figure 20 shows a variation of the training with tuning done on several parameters. The complete record of tuning variations is shown in Appendix D. The summary of tuning variations are presented in Table 7. From the results, the RF model shows the highest accuracy and performance utilizing the original Ntree value of 500, with a modified Mtry of 6, and Nodesize of 2.

Table 7. Tuning Variations on Balanced Random Forest

Tune No.	Ntree	Mtry	Importance	Nodesize	Maxnodes	Accuracy	Balanced Accuracy	Result
Original	*500*	*4*	*TRUE*	*NULL*	*NULL*	*0.893532*	*0.885138*	*[Benchmark]*
1	**600**	**6**	-	-	-	0.8943	0.8867	Improve
2	**600**	4	-	-	-	0.8928	0.8847	Decline
3	500	**6**	-	-	-	0.8942	0.8861	Improve
4	500	**6**	**TRUE**	**2**	-	**0.8954**	**0.8872**	**Improve**
5	500	**6**	**TRUE**	**2**	**50 000**	0.8944	0.8864	Improve
6	500	**6**	**TRUE**	**2**	**100 000**	0.8946	0.8867	Improve

Although overall, the tuning for RF model does not produce significant improvements, the Random Forest model with *Ntree = 500, Mtry = 6,* with *Nodesize 2* still shows the best performance amongst the other DT and SVM tuned models. The ROC graph of the most optimal tuned DT and RF model in Figure 21 also show that the second best tuned model DT has increased value in AUC, however is still lower versus the balanced and tuned RF model. Thus, the RF tuned model is chosen as the final model that fits best in predicting hotel booking outcomes. From the importance factor of the RF model, the most important features that influence the prediction value are also extracted. This is further discussed in the Analysis section.

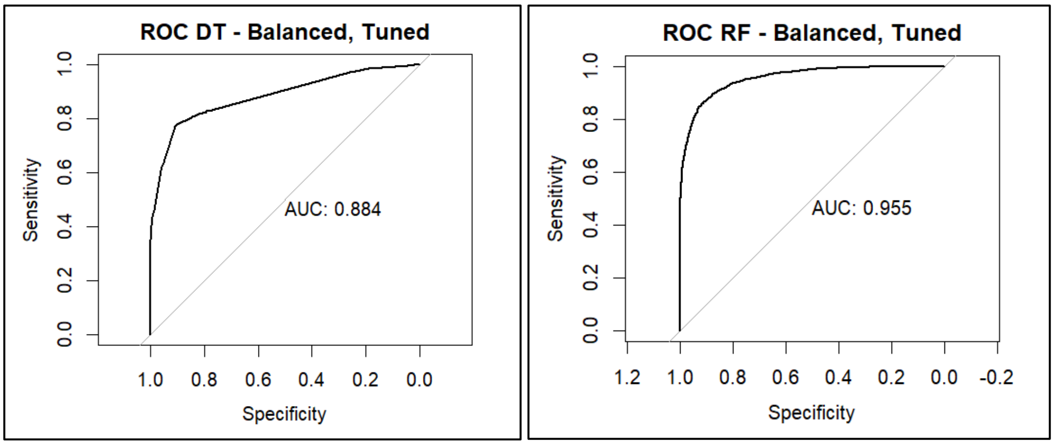

Figure 21. ROC and AUC – Balanced and Tuned RF

Balanced and Normalized Data

Lastly, the model implementation is also done on balanced and normalized dataset. However the results shown from this normalization procedure do not give any significant improvements to the model performances. Table 8 shows the performance of each model when run with normalized numerical variables.

Table 8. Balanced and Normalized Model Performance Metrics

No.	Model	Accuracy	F1	Precision	Recall	BalancedAcc
1	RF Bal + Norm	0.893421	0.912874	0.901651	0.924379	0.885288
2	SVM Bal + Norm	0.825183	0.856596	0.848934	0.864397	0.814882
3	DT Bal + Norm	0.80429	0.845404	0.808429	0.885925	0.782844

In summary, the various model implementation and validation showed that the Random Forest model is superior in giving much higher accuracy, F1, precision, and recall values compared to DT and SVM. The class balancing done impacts SVM the highest, with 3% increase in Balanced accuracy, compared to only +1% in RF and -0.1% in the DT model. However, significant improvement in balanced accuracy still puts the SVM model at second place, with the RF accuracy at 89.3% with the balanced dataset. The RF model's F1, Precision, and Recall values are even higher, with values all above 90%, showing the robustness of the model and minimizing false positive and false negatives. Figure 22 shows a benchmark of the Accuracy of the three model implementations done. The complete record of Confusion Matrices values are attached in Appendix A-C.

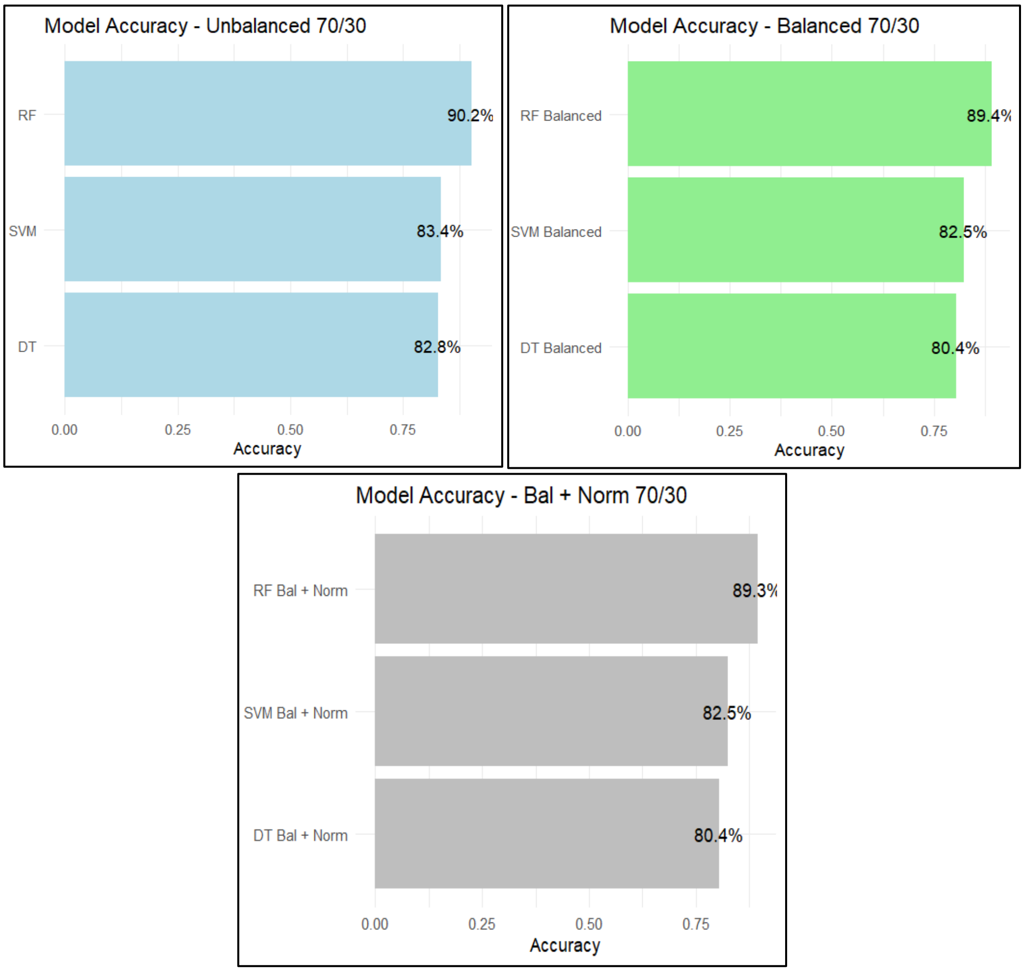

Figure 22. Accuracy Comparison with Unbalanced, Balanced, and Normalized Model

4.0 RESULTS

This section discusses the analysis and recommendations from the model implementation and validation process executed. From the preprocessing steps and model selection, overall the model performance demonstrated surpass those from the related works executed on the same dataset, only being dragged down slightly due to the class balancing procedure that is executed, which is done in the other works. The performance each of the model also aligns with results obtained from the related works, with the Random Forest model proving better accuracy in generating predictions.

4.1 Analysis

The analysis discussed involves the implementation and validation procedures done in the previous section. These include discussion on the class balancing step, data normalization process, the hyperparameter tuning for Balanced models along with parameter considerations, as well as the feature importance identification from RF, comparing it to the DT generated tree structure and plot.

Table 9. Performance Metrics Summary

Model (Tune Param)	**Imbalanced**	**Balanced**	**Balanced + Tuned**	**Balanced + Normalized**
RF (Mtry = 6, Nodesize = 2)				
Accuracy	0.902401	0.893532	**0.8954**	0.893421
F1	0.929016	0.913051	**0.9140**	0.912874
Precision	0.908889	0.900949	**0.9028**	0.901651
Recall	0.950055	0.925483	**0.9266**	0.924379
BalancedAcc	0.877357	0.885138	**0.8872**	0.885288
SVM (Cost = 20)				
Accuracy	0.833962	0.825183	**0.8403**	0.825183
F1	0.881818	0.856596	**0.8683**	0.856596
Precision pos pred	0.845449	0.848934	**0.8649**	0.848934
Recall sensitiv	0.921456	0.864397	**0.8718**	0.864397
BalancedAcc	0.78798	0.814882	**0.8320**	0.814882
DT (CP = 0.001)				
Accuracy	0.827615	0.80429	**0.8537**	0.80429
F1	0.876499	0.845404	**0.8822**	0.845404
Precision	0.845411	0.808429	**0.8589**	0.808429
Recall	0.909962	0.885925	**0.9069**	0.885925
BalancedAcc	0.784338	0.782844	**0.8398**	0.782844

Class Balancing

In the model implementation and validation phase, three variations of each model are done; with the imbalanced, balanced, and balanced + normalized data. From these three variations, overall, the imbalanced dataset still exhibits the highest overall performance versus the other variations on values of Accuracy, F1, Precision and Recall in the Balanced (not tuned) dataset (Table 9). However, from the view of Balanced Accuracy parameter, the balanced dataset serves a better model. As such, although the balanced dataset performs slightly lower than the imbalanced data, the balanced set is still chosen, as the model is more accurately trained and is better in making balanced judgement. Further tuning on the balanced dataset also allows the models to perform better in studying and predicting the data. Analysis on the normalized and tuned models are stated on the next sub headings.

Data Normalization

Table 9 also shows that results of Balanced + Normalized dataset is similar as the non-normalized data for the SVM and DT models. The reason behind this may be due to the deep tree structure in the decision tree model; thus, there is no significant difference whether the data is normalized or not. For the case of SVM model, the input variables may not be correlated to each other, thus the SVM algorithm is unable to find the exactly accurate hyperplane to separate the data. The linearity of the data may also affect the SVM and DT algorithm in executing the prediction with accurate results, resulting in minimal impact of data normalization. The normalized model version of the RF is also dropping slightly in Accuracy, F1, and Recall values, despite slight increase in Precision and Balanced accuracy. The reasoning behind this may be due to the ability of the RF model to detect non-linear relationships in the data without the normalizing procedures.

Tuning Parameters

From results in Table 9, it can also be seen that the tuning attempts on the three balanced models allow improvements on the model performances. Tuning on the RF model is initially done with only the Ntree and Mtry parameters, which are the two most common and impactful parameters in RF modeling in R, representing the number of trees in the data (ntree) and the number of features that are considered in each node split (mtry). However, as modifications on ntree and mtry show no significant changes, the tuning parameters are expanded to nodesize and maxnodes which determine the node specifications. Despite experimentation on multiple parameters, the tuning showed only a small improvement in performance compared to the default model, with the difference in performance only at 0.1-0.3% versus the initial model. Analyzed from the dataset point of view, this may be due to the data already fitting well with the default model generated, therefore there is little to improve from the parameter tunings. Another reason may be due to the data is not noisy, thus the model are not impacted a lot from hyperparameter tuning attempts.

Tuning for the DT model is done using the complexity parameter (cp) value that determines how the model should divide its nodes in classifying the data. From the before-after comparison tree structure in Figure 19 (pg.20), it can be seen that tuning allows the tree design to become more complex, with more splits and consideration of different variables. The amount of split at each node also varies more, compared to the initial one that split only to two branches. Analyzing the DT Tuned model further, the split of each node in the model is displayed in Figure 23.

```
> best_model
n= 20999

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 20999 8315 1 (0.60402876 0.39597124)
   2) lead_time< 151.5 16381 4702 1 (0.71296014 0.28703986)
     4) no_of_special_requests>=0.5 7604 1185 1 (0.84416097 0.15583903)
       8) date>=17369.5 7557 1140 1 (0.84914649 0.15085351) *
       9) date< 17369.5 47    2 2 (0.04255319 0.95744681) *
     5) no_of_special_requests< 0.5 8777 3517 1 (0.59929361 0.40070639)
      10) market_segment_type=1,2,3,4 4199  781 1 (0.81400333 0.18599667)
        20) lead_time< 90.5 3232  368 1 (0.88613861 0.11386139) *
        21) lead_time>=90.5 967  413 1 (0.57290589 0.42709411)
          42) arrival_month=1,2,3,6,8,9,11,12 543  136 1 (0.74953959 0.25046041) *
          43) arrival_month=4,5,7,10 424  147 2 (0.34669811 0.65330189) *
      11) market_segment_type=5 4578 1842 2 (0.40235911 0.59764089)
        22) lead_time< 13.5 1122  327 1 (0.70855615 0.29144385) *
        23) lead_time>=13.5 3456 1047 2 (0.30295139 0.69704861)
          46) date< 17562.5 319  125 1 (0.60815047 0.39184953)
            92) lead_time< 81.5 221   45 1 (0.79638009 0.20361991) *
            93) lead_time>=81.5 98   18 2 (0.18367347 0.81632653) *
          47) date>=17562.5 3137  853 2 (0.27191584 0.72808416)
            94) required_car_parking_space>=0.5 43    0 1 (1.00000000 0.00000000) *
            95) required_car_parking_space< 0.5 3094  810 2 (0.26179703 0.73820297) *
   3) lead_time>=151.5 4618 1005 2 (0.21762668 0.78237332)
     6) avg_price_per_room< 100.04 2428  925 2 (0.38097199 0.61902801)
      12) no_of_special_requests>=0.5 661  235 1 (0.64447806 0.35552194)
        24) no_of_weekend_nights>=0.5 448  101 1 (0.77455357 0.22544643) *
        25) no_of_weekend_nights< 0.5 213   79 2 (0.37089202 0.62910798)
          50) lead_time< 180.5 63    9 1 (0.85714286 0.14285714) *
          51) lead_time>=180.5 150   25 2 (0.16666667 0.83333333) *
      13) no_of_special_requests< 0.5 1767  499 2 (0.28239955 0.71760045)
        26) no_of_adults< 1.5 400  131 1 (0.67250000 0.32750000)
          52) market_segment_type=2,4 319   57 1 (0.82131661 0.17868339) *
          53) market_segment_type=5 81    7 2 (0.08641975 0.91358025) *
        27) no_of_adults>=1.5 1367  230 2 (0.16825165 0.83174835) *
     7) avg_price_per_room>=100.04 2190   80 2 (0.03652968 0.96347032) *
```

Figure 23. DT Tuned Model Structure Text

From the figure, which shows are more readable version from Figure 19 tree structure, shows that the variable Lead time is the most important split point in the data, which exist as the first data split node, and also appears on the lower nodes. The important variables is followed with the number of special requests and market segment, with time related attributes such as arrival month and date also considered on the lower split nodes.

Lastly, for tuning efforts on the SVM model, the *epsilon* and *cost* are used. These two parameters are utilized as they are among the most important parameters to modify how and SVM model behave in digesting the data. The tuning of *epsilon* and *cost* allows control on the error margin in the model, which may affect the robustness and the accuracy of the model. Although, an extreme value assigned to these parameters may allow the model to be overfitting. Thus, in determining the optimal tuning parameters, the *cost* = 20 tuned model is finally chosen as the best SVM model despite not having the highest test set accuracy value. This is due to the train set accuracy value rising in an increasing pace compared to the test set, which may indicate increasing chance of overfitting.

Feature Importance

Apart from the preprocessing, implementation and validation attempts, another important part of the model generation is from the feature extraction that may be done utilizing the model results. In the Random Forest model training process, one parameter that the model calculates when training and generating prediction is the MeanDecreaseGini of each variable from the Gini impurity of a node. A node with a high value of Gini impurity shows that the variable helps to separate the class on the higher level of separation. Thus, the higher the Gini value, the more important the attribute is. From Figure 24, it can be concluded that the top 5 attributes that impactsthe prediction model is lead time, average room price, date, no of special requests, and arrival month. With this importance information, a more specific model may also be trained just by selecting these important features as input variables.

```
> mod_rf_bal$importance
                                     MeanDecreaseGini
no_of_adults                               202.543774
no_of_children                              65.472455
no_of_weekend_nights                       278.803082
no_of_week_nights                          378.526887
type_of_meal_plan                          199.894349
required_car_parking_space                  65.872712
room_type_reserved                         146.402620
lead_time                                 2635.213513
arrival_year                               126.474110
arrival_month                              632.941662
arrival_date                               562.220308
date                                      1059.463337
market_segment_type                        517.649587
repeated_guest                              21.042867
no_of_previous_cancellations                 2.389999
no_of_previous_bookings_not_canceled        17.476854
avg_price_per_room                        1139.856844
no_of_special_requests                    1019.534640
```

Figure 24. Importance – RF Model

As the RF model is an ensemble model derived from the DT model, the Gini values stated may also resonate with the significance of variables in the DT model. Comparing results of RF gini value and the DT tuned model structure, it can be seen that both models have several similarities in building the most optimal model. In both model, the lead_time variable plays the most important role, followed with the number of requests. The time related attributes such as arrival_month and date also have high Gini scores, which resonates with their presence in the DT tree structure. However, in the RF tuned model, the average room price actually has higher gini value than its position in the DT tree, where room prices are determined for classification on the lower nodes. This shows that the difference in attribute importance of the two models might be one of the reasons why the two models resulted in different accuracy and performance levels.

4.2 Recommendations

From the results of the model implementation and validation processes, several recommendations can be given for further study in this area. Firstly, as was discussed in the related works section, suitable feature extraction and selection are one of the key methods to increase a model's performance. The models generated in this study has not utilized this feature selection method in the tuning and validation procedures. The utilization of these selection features might be able to further increase the Random Forest model performance, as well as improving the DT and SVM models. The feature selection may be experimented from the Gini values generated from the RF model and the generated DT plot, apart from other extraction methods.

Second, the machine learning methods utilized in this study are mainly from traditional models, not including more complex boosting models such as XGB, GB and NN. The utilization of these algorithms may explore hidden patterns that are not identified from usage of the SVM and tree-based models DT and RF.

Third, the validation process done in this study makes use of only the train and test split of the dataset. Utilization of *k-fold* cross validation is feasible to see how the model work with different training and testing subset of the data. This method would also allow the model to be trained on more data samples, increasing the chances of better performance in Accuracy, F1, Precision, and Recall values.

5.0 CONCLUSION

The study initially started the model training by comparing the performance of imbalanced and balanced dataset models. Although accuracy and performance metrics are slightly higher in the imbalanced dataset, the balanced dataset is preferable as it allows for higher Balanced Accuracy, proving the robustness of the model in making predictions. The difference in performance is also not significant enough for the imbalanced set to be chosen over the balanced one.

Next, implementation of the models show that the RF model performs best in all training variations of balanced and balanced + normalized data. Model tuning attempts are also done on the three initial models, which results in significant improvements on the DT and SVM models. The DT model performance improve by a significant 5%, with the SVM tuning following at 1.5% improvement. Tuning attempts on the RF model resulted in very small improvement in model performance, however the initial superiority of the model allow the model to still stay as the best performing out of the three. The final RF model with highest performance is chosen as the most optimal model to predict hotel booking canceclations, where the hyperparameter tuning include values of Ntree 500, Mtry 6, and Nodesize 2. The model final performance is at 89.54% Accuracy, 91.4% F1, 90.28% Prec, 92.66% Recall, and 88.72% Balanced Accuracy.

Exploration on the dataset also allowed insights on the patterns on the data. The correlation matrix showed that the variable with the highest correlation with the target variable *booking_canceled* is *lead_time*, followed with *no_of_special_requests*, *arrival_year*, *market_segment_type*, and *avg_price_per_room*. This correlation measures aligns with the attributes that are then identified as most important from the RF and DT model training.

The model generation in this study provided a robust model with a high reliability for hotel booking outcome predictions. However further improvements are still feasible on many areas of the study. Utilization of feature selection may allow the models to perform better from only the most important variables. Usage of more thorough validation procedures such as cross validation on the hyperparameter tuning procedures may also evaluate the model performances better due to training done on multiple combinations of the data. Further exploration on the RF model hyperparameters should also be able to increase the model's performance further.

REFERENCES

Antonio, N., Almeida, A. de, & Nunes, L. (2017). Predicting hotel booking cancellations to decrease uncertainty and increase revenue. *Tourism & Management Studies*, *13*(2), 25–39. https://doi.org/10.18089/TMS.2017.13203

Antonio, N., de Almeida, A., & Nunes, L. (2019a). Big Data in Hotel Revenue Management: Exploring Cancellation Drivers to Gain Insights Into Booking Cancellation Behavior. *Cornell Hospitality Quarterly*, *60*(4), 298–319. https://doi.org/10.1177/1938965519851466

Antonio, N., de Almeida, A., & Nunes, L. (2019b). Hotel booking demand datasets. *Data in Brief*, *22*, 41–49. https://doi.org/10.1016/J.DIB.2018.11.126

Bonaccorso, G. (2018). *Machine Learning Algorithms: Popular algorithms for data science and machine learning* (2nd ed.).

Chen, S., Ngai, E. W. T., Ku, Y., Xu, Z., Gou, X., & Zhang, C. (2023). Prediction of hotel booking cancellations: Integration of machine learning and probability model based on interpretable feature interaction. *Decision Support Systems*, *170*, 113959. https://doi.org/10.1016/J.DSS.2023.113959

Lee, C. S., Yeng, P., Cheang, S., & Moslehpour, M. (2022). *Predictive Analytics in Business Analytics: Decision Tree*.

Li, W., Ma, X., Chen, Y., Dai, B., Chen, R., Tang, C., Luo, Y., & Zhang, K. (2021). Random Fuzzy Granular Decision Tree. *Mathematical Problems in Engineering*, *2021*. https://doi.org/10.1155/2021/5578682

Mahesh, B. (2018). Machine Learning Algorithms-A Review. *International Journal of Science and Research*. https://doi.org/10.21275/ART20203995

Masiero, L., Viglia, G., & Nieto-Garcia, M. (2020). Strategic consumer behavior in online hotel booking. *Annals of Tourism Research*, *83*, 102947. https://doi.org/10.1016/J.ANNALS.2020.102947

Sánchez, E. C., Sánchez-Medina, A. J., & Pellejero, M. (2020). Identifying critical hotel cancellations using artificial intelligence. *Tourism Management Perspectives*, *35*, 100718. https://doi.org/10.1016/J.TMP.2020.100718

Satu, M. S., Ahammed, K., Abedin, M., & Abedin, M. Z. (2021). *Performance Analysis of Machine Learning Techniques to Predict Hotel booking Cancellations in Hospitality Industry*. https://doi.org/10.1109/ICCIT51783.2020.9392648

Sekeroglu, A. G. (2023). *Modeling-Tuning-ExplainableAI-EDA-PandasProfiling | Kaggle*. https://www.kaggle.com/code/galipsekeroglu/modeling-tuning-explainableai-eda-pandasprofiling

Singh Kushwah, J., Kumar, A., Patel, S., Soni, R., Gawande, A., & Gupta, S. (2022). Comparative study of regressor and classifier with decision tree using modern tools. *Materials Today: Proceedings*, *56*, 3571–3576. https://doi.org/10.1016/J.MATPR.2021.11.635

Slimbensalah. (2023). *RMarkdown - Booking Classification EDA + Modeling | Kaggle*. https://www.kaggle.com/code/slimbensalah/rmarkdown-booking-classification-eda-modeling/report

Timamopoulos, C. (2020). *Anomaly Detection: Predicting hotel booking cancellations*. https://repository.ihu.edu.gr//xmlui/handle/11544/29631

Vinitnantharat, N. (2023). *Hotel reservation classification | Kaggle*. https://www.kaggle.com/code/nasvirat/hotel-reservation-classification

WTTC. (2023). *Economic Impact Records*. https://wttc.org/research/economic-impact

Yuxuan, C., Tuan-Chun, H., Zhuofan, J., Melvin, T. C. W., Goyal, V., & Yijun, Z. (2023). Hotel Booking Cancellation Analytics on Imbalanced Data. *Tourism Analytics Before and After COVID-19*, 97–117. https://doi.org/10.1007/978-981-19-9369-5_7

Appendix A – Random Forest Confusion Matrix

Random Forest – Imbalanced

```
> confusionMatrix(rf_predict, test_set$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE  6943  696
     TRUE    365 2867

               Accuracy : 0.9024
                 95% CI : (0.8967, 0.9079)
    No Information Rate : 0.6722
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7731

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9501
            Specificity : 0.8047
         Pos Pred Value : 0.9089
         Neg Pred Value : 0.8871
             Prevalence : 0.6722
         Detection Rate : 0.6387
   Detection Prevalence : 0.7027
      Balanced Accuracy : 0.8774

       'Positive' Class : FALSE
```

Random Forest - Balanced

```
> confusionMatrix(rf_predict_bal, test_set_balanced$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 5030  553
         2  405 3010

               Accuracy : 0.8935
                 95% CI : (0.887, 0.8998)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7758

 Mcnemar's Test P-Value : 2.041e-06

            Sensitivity : 0.9255
            Specificity : 0.8448
         Pos Pred Value : 0.9009
         Neg Pred Value : 0.8814
             Prevalence : 0.6040
         Detection Rate : 0.5590
   Detection Prevalence : 0.6205
      Balanced Accuracy : 0.8851

       'Positive' Class : 1
```

Random Forest – Balanced– Validation with Training Set

```
> rf_trained_bal = predict(mod_rf_bal, newdata = train_set_balanced)
> confusionMatrix(rf_trained_bal, train_set_balanced$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction     1     2
         1 12537   208
         2   147  8107

               Accuracy : 0.9831
                 95% CI : (0.9813, 0.9848)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9646

 Mcnemar's Test P-Value : 0.00145

            Sensitivity : 0.9884
            Specificity : 0.9750
         Pos Pred Value : 0.9837
         Neg Pred Value : 0.9822
             Prevalence : 0.6040
         Detection Rate : 0.5970
   Detection Prevalence : 0.6069
      Balanced Accuracy : 0.9817

       'Positive' Class : 1
```

Random Forest – Balanced Tuned

```
> confusionMatrix(rftuned_predict_bal, test_set_balanced
$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction    1    2
         1 5036  542
         2  399 3021

               Accuracy : 0.8954
                 95% CI : (0.8889, 0.9017)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7799

 Mcnemar's Test P-Value : 3.673e-06

            Sensitivity : 0.9266
            Specificity : 0.8479
         Pos Pred Value : 0.9028
         Neg Pred Value : 0.8833
             Prevalence : 0.6040
         Detection Rate : 0.5597
   Detection Prevalence : 0.6199
      Balanced Accuracy : 0.8872

       'Positive' Class : 1
```

Random Forest – Balanced Tuned – Validation with Training Set

CROSS VAL WITH TRAIN DATA - **Random Forest - still overfit**

```
> rftuned_train_bal = predict(mod_rftuned_bal, newdata = train_set_balanced)
> confusionMatrix(rftuned_train_bal, train_set_balanced$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction     1     2
         1 12633   142
         2    51  8173

               Accuracy : 0.9908
                 95% CI : (0.9894, 0.9921)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9808

 Mcnemar's Test P-Value : 9.274e-11

            Sensitivity : 0.9960
            Specificity : 0.9829
         Pos Pred Value : 0.9889
         Neg Pred Value : 0.9938
             Prevalence : 0.6040
         Detection Rate : 0.6016
   Detection Prevalence : 0.6084
      Balanced Accuracy : 0.9895

       'Positive' Class : 1
```

Appendix B – Decision Tree Confusion Matrix

Decision Tree – Unbalanced

```
> confusionMatrix(dt_predict, test_set$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE  6650 1216
     TRUE    658 2347

               Accuracy : 0.8276
                 95% CI : (0.8204, 0.8347)
    No Information Rate : 0.6722
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5924

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9100
            Specificity : 0.6587
         Pos Pred Value : 0.8454
         Neg Pred Value : 0.7810
             Prevalence : 0.6722
         Detection Rate : 0.6117
   Detection Prevalence : 0.7236
      Balanced Accuracy : 0.7843

       'Positive' Class : FALSE
```

Decision Tree - Balanced

```
> confusionMatrix(dt_predict_bal, test_set_balanced$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 4815 1141
         2  620 2422

               Accuracy : 0.8043
                 95% CI : (0.7959, 0.8124)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5803

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8859
            Specificity : 0.6798
         Pos Pred Value : 0.8084
         Neg Pred Value : 0.7962
             Prevalence : 0.6040
         Detection Rate : 0.5351
   Detection Prevalence : 0.6619
      Balanced Accuracy : 0.7828

       'Positive' Class : 1
```

Decision Tree – Balanced– Validation with Training Set

```
> dt_trained_bal = predict(mod_dt_bal, newdata = train_set_balanced, type = "class")
> confusionMatrix(dt_trained_bal, train_set_balanced$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction     1     2
         1 11327  2659
         2  1357  5656

               Accuracy : 0.8088
                 95% CI : (0.8034, 0.8141)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5891

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8930
            Specificity : 0.6802
         Pos Pred Value : 0.8099
         Neg Pred Value : 0.8065
             Prevalence : 0.6040
         Detection Rate : 0.5394
   Detection Prevalence : 0.6660
      Balanced Accuracy : 0.7866

       'Positive' Class : 1
```

Decision Tree – Balanced Tuned

```
> confusionMatrix(dttuned_predict_bal, test_set_balanced$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 4819  951
         2  616 2612

               Accuracy : 0.8259
                 95% CI : (0.8179, 0.8336)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.63

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8867
            Specificity : 0.7331
         Pos Pred Value : 0.8352
         Neg Pred Value : 0.8092
             Prevalence : 0.6040
         Detection Rate : 0.5356
   Detection Prevalence : 0.6413
      Balanced Accuracy : 0.8099

       'Positive' Class : 1
```

Decision Tree – Balanced Tuned – Validation with Training Set

```
CROSS VAL WITH TRAIN DATA - DT not overfit
> dttuned_train_bal = predict(mod_dttuned_bal,
"class")
> confusionMatrix(dttuned_train_bal, train_set_balanced$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction     1     2
         1 11365  2183
         2  1319  6132

               Accuracy : 0.8332
                 95% CI : (0.8281, 0.8382)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.645

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8960
            Specificity : 0.7375
         Pos Pred Value : 0.8389
         Neg Pred Value : 0.8230
             Prevalence : 0.6040
         Detection Rate : 0.5412
   Detection Prevalence : 0.6452
      Balanced Accuracy : 0.8167

       'Positive' Class : 1
```

Appendix C – Support Vector Machine Confusion Matrix

Support Vector Machine – Unbalanced

```
> confusionMatrix(svm_predict, test_set$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE  6734 1231
     TRUE    574 2332

               Accuracy : 0.834
                 95% CI : (0.8268, 0.8409)
    No Information Rate : 0.6722
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6045

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9215
            Specificity : 0.6545
         Pos Pred Value : 0.8454
         Neg Pred Value : 0.8025
             Prevalence : 0.6722
         Detection Rate : 0.6194
   Detection Prevalence : 0.7327
      Balanced Accuracy : 0.7880

       'Positive' Class : FALSE
```

Support Vector Machine - Balanced

```
> confusionMatrix(svm_predict_bal, test_set_balanced$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 4698  836
         2  737 2727

               Accuracy : 0.8252
                 95% CI : (0.8172, 0.833)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.6328

 Mcnemar's Test P-Value : 0.01348

            Sensitivity : 0.8644
            Specificity : 0.7654
         Pos Pred Value : 0.8489
         Neg Pred Value : 0.7872
             Prevalence : 0.6040
         Detection Rate : 0.5221
   Detection Prevalence : 0.6150
      Balanced Accuracy : 0.8149

       'Positive' Class : 1
```

Support Vector Machine – Balanced– Validation with Training Set

```
> svm_trained_bal = predict(mod_svm_bal, newdata = train_set_balanced)
> confusionMatrix(svm_trained_bal, train_set_balanced$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction     1     2
         1 11081  1873
         2  1603  6442

               Accuracy : 0.8345
                 95% CI : (0.8294, 0.8395)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.652

 Mcnemar's Test P-Value : 5.052e-06

            Sensitivity : 0.8736
            Specificity : 0.7747
         Pos Pred Value : 0.8554
         Neg Pred Value : 0.8007
             Prevalence : 0.6040
         Detection Rate : 0.5277
   Detection Prevalence : 0.6169
      Balanced Accuracy : 0.8242

       'Positive' Class : 1
```

Support Vector Machine – Balanced Tuned

```
> mod_svmtuned_bal <- svm(booking_canceled~., data = train_set_balanced, epsilon = 0,
cost = 4)
> svmtuned_predict_bal = predict(mod_svmtuned_bal, newdata = test_set_balanced)
>
> confusionMatrix(svmtuned_predict_bal, test_set_balanced$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction    1    2
         1 4685  774
         2  750 2789

               Accuracy : 0.8306
                 95% CI : (0.8227, 0.8383)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6455

 Mcnemar's Test P-Value : 0.5558

            Sensitivity : 0.8620
            Specificity : 0.7828
         Pos Pred Value : 0.8582
         Neg Pred Value : 0.7881
             Prevalence : 0.6040
         Detection Rate : 0.5207
   Detection Prevalence : 0.6067
      Balanced Accuracy : 0.8224

       'Positive' Class : 1
```

Support Vector Machine – Balanced Tuned – Validation with Training Set

```
CROSS VAL WITH TRAIN DATA - not overfit
> svmtuned_train_bal = predict(mod_svmtuned_bal, newdata = train_set_balanced)
> confusionMatrix(svmtuned_train_bal, train_set_balanced$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction     1     2
         1 11101  1675
         2  1583  6640

               Accuracy : 0.8448
                 95% CI : (0.8399, 0.8497)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.675

 Mcnemar's Test P-Value : 0.1109

            Sensitivity : 0.8752
            Specificity : 0.7986
         Pos Pred Value : 0.8689
         Neg Pred Value : 0.8075
             Prevalence : 0.6040
         Detection Rate : 0.5286
   Detection Prevalence : 0.6084
      Balanced Accuracy : 0.8369

       'Positive' Class : 1
```

Appendix D – Random Forest Hyperparameter Tuning

NTREE + MTRY, IMPROVE

```
> mod_rftuned_bal = randomForest(booking_canceled ~ ., data =
train_set_balanced,
+                                ntree = 600,
+                                mtry = 6)
> rftuned_predict_bal = predict(mod_rftuned_bal, newdata =
test_set_balanced)
>
> confusionMatrix(rftuned_predict_bal, test_set_balanced
$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction    1    2
         1 5017  533
         2  418 3030

               Accuracy : 0.8943
                 95% CI : (0.8878, 0.9006)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7778

 Mcnemar's Test P-Value : 0.0002184

            Sensitivity : 0.9231
            Specificity : 0.8504
         Pos Pred Value : 0.9040
         Neg Pred Value : 0.8788
             Prevalence : 0.6040
         Detection Rate : 0.5576
   Detection Prevalence : 0.6168
      Balanced Accuracy : 0.8867

       'Positive' Class : 1
```

NTREE TUNING - NEGATIVE

```
> mod_rftuned_bal = randomForest(booking_canceled ~ ., data =
train_set_balanced,
+                                ntree = 600,
+                                mtry = 4)
> rftuned_predict_bal = predict(mod_rftuned_bal, newdata =
test_set_balanced)
>
> confusionMatrix(rftuned_predict_bal, test_set_balanced
$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction    1    2
         1 5019  549
         2  416 3014

               Accuracy : 0.8928
                 95% CI : (0.8862, 0.8991)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7744

 Mcnemar's Test P-Value : 2.145e-05

            Sensitivity : 0.9235
            Specificity : 0.8459
         Pos Pred Value : 0.9014
         Neg Pred Value : 0.8787
             Prevalence : 0.6040
         Detection Rate : 0.5578
   Detection Prevalence : 0.6188
      Balanced Accuracy : 0.8847

       'Positive' Class : 1
```

MTRY TUNING - IMPROVE

```
> mod_rftuned_bal = randomForest(booking_canceled ~ ., data =
train_set_balanced,
+                                ntree = 500,
+                                mtry = 6)
> rftuned_predict_bal = predict(mod_rftuned_bal, newdata =
test_set_balanced)
>
> confusionMatrix(rftuned_predict_bal, test_set_balanced
$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction    1    2
         1 5027  544
         2  408 3019

               Accuracy : 0.8942
                 95% CI : (0.8877, 0.9005)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7774

 Mcnemar's Test P-Value : 1.212e-05

            Sensitivity : 0.9249
            Specificity : 0.8473
         Pos Pred Value : 0.9024
         Neg Pred Value : 0.8809
             Prevalence : 0.6040
         Detection Rate : 0.5587
   Detection Prevalence : 0.6191
      Balanced Accuracy : 0.8861

       'Positive' Class : 1
```

+NODESIZE TUNING - IMPROVE - CHOSEN AS FINAL MODEL

```
> mod_rftuned_bal = randomForest(booking_canceled ~ ., data =
train_set_balanced,
+                                ntree = 500,
+                                mtry = 6,
+                                importance = TRUE,
+                                nodesize = 2
+                                )
> rftuned_predict_bal = predict(mod_rftuned_bal, newdata =
test_set_balanced)|
>
> confusionMatrix(rftuned_predict_bal, test_set_balanced
$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction    1    2
         1 5036  542
         2  399 3021

               Accuracy : 0.8954
                 95% CI : (0.8889, 0.9017)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7799

 Mcnemar's Test P-Value : 3.673e-06

            Sensitivity : 0.9266
            Specificity : 0.8479
         Pos Pred Value : 0.9028
         Neg Pred Value : 0.8833
             Prevalence : 0.6040
         Detection Rate : 0.5597
   Detection Prevalence : 0.6199
      Balanced Accuracy : 0.8872

       'Positive' Class : 1
```

+MAXNODES TUNING - DROPPING, NOT USED.

5K MAXNODES

```
> mod_rftuned_bal = randomForest(booking_canceled ~ ., data =
train_set_balanced,
+                                ntree = 500,
+                                mtry = 6,
+                                importance = TRUE,
+                                nodesize = 2,
+                                maxnodes = 5000
+                                )
> rftuned_predict_bal = predict(mod_rftuned_bal, newdata =
test_set_balanced)
>
> confusionMatrix(rftuned_predict_bal, test_set_balanced
$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction    1    2
         1 5027  542
         2  408 3021

               Accuracy : 0.8944
                 95% CI : (0.8879, 0.9007)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7779

 Mcnemar's Test P-Value : 1.595e-05

            Sensitivity : 0.9249
            Specificity : 0.8479
         Pos Pred Value : 0.9027
         Neg Pred Value : 0.8810
             Prevalence : 0.6040
         Detection Rate : 0.5587
   Detection Prevalence : 0.6189
      Balanced Accuracy : 0.8864

       'Positive' Class : 1
```

10K MAX NODES

```
> mod_rftuned_bal = randomForest(booking_canceled ~ ., data =
train_set_balanced,
+                                ntree = 500,
+                                mtry = 6,
+                                importance = TRUE,
+                                nodesize = 2,
+                                maxnodes = 10000
+                                )
> rftuned_predict_bal = predict(mod_rftuned_bal, newdata =
test_set_balanced)
>
> confusionMatrix(rftuned_predict_bal, test_set_balanced
$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction    1    2
         1 5026  539
         2  409 3024

               Accuracy : 0.8946
                 95% CI : (0.8881, 0.9009)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7784

 Mcnemar's Test P-Value : 2.793e-05

            Sensitivity : 0.9247
            Specificity : 0.8487
         Pos Pred Value : 0.9031
         Neg Pred Value : 0.8809
             Prevalence : 0.6040
         Detection Rate : 0.5586
   Detection Prevalence : 0.6185
      Balanced Accuracy : 0.8867

       'Positive' Class : 1
```

Appendix E – SVM Hyperparameter Tuning

```
> mod_svmtuned_bal <- svm(booking_canceled~., data = train_set_balanced, epsilon = 0,
cost = 2)
> svmtuned_predict_bal = predict(mod_svmtuned_bal, newdata = test_set_balanced)
>
> confusionMatrix(svmtuned_predict_bal, test_set_balanced$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction    1    2
         1 4688  788
         2  747 2775

               Accuracy : 0.8294
                 95% CI : (0.8215, 0.8371)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6427

 Mcnemar's Test P-Value : 0.3073

            Sensitivity : 0.8626
            Specificity : 0.7788
         Pos Pred Value : 0.8561
         Neg Pred Value : 0.7879
             Prevalence : 0.6040
         Detection Rate : 0.5210
   Detection Prevalence : 0.6086
      Balanced Accuracy : 0.8207

       'Positive' Class : 1

> mod_svmtuned_bal <- svm(booking_canceled~., data = train_set_balanced, epsilon = 0,
cost = 4)
> svmtuned_predict_bal = predict(mod_svmtuned_bal, newdata = test_set_balanced)
>
> confusionMatrix(svmtuned_predict_bal, test_set_balanced$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction    1    2
         1 4685  774
         2  750 2789

               Accuracy : 0.8306
                 95% CI : (0.8227, 0.8383)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6455

 Mcnemar's Test P-Value : 0.5558

            Sensitivity : 0.8620
            Specificity : 0.7828
         Pos Pred Value : 0.8582
         Neg Pred Value : 0.7881
             Prevalence : 0.6040
         Detection Rate : 0.5207
   Detection Prevalence : 0.6067
      Balanced Accuracy : 0.8224

       'Positive' Class : 1

> mod_svmtuned_bal <- svm(booking_canceled~., data = train_set_balanced, epsilon = 1,
cost = 4)
> svmtuned_predict_bal = predict(mod_svmtuned_bal, newdata = test_set_balanced)
>
> confusionMatrix(svmtuned_predict_bal, test_set_balanced$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction    1    2
         1 4685  774
         2  750 2789

               Accuracy : 0.8306
                 95% CI : (0.8227, 0.8383)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6455

 Mcnemar's Test P-Value : 0.5558

            Sensitivity : 0.8620
            Specificity : 0.7828
         Pos Pred Value : 0.8582
         Neg Pred Value : 0.7881
             Prevalence : 0.6040
         Detection Rate : 0.5207
   Detection Prevalence : 0.6067
      Balanced Accuracy : 0.8224

       'Positive' Class : 1
```

```
> mod_svmtuned_bal <- svm(booking_canceled~., data = train_set_balanced, epsilon = 2
cost = 4)
> svmtuned_predict_bal = predict(mod_svmtuned_bal, newdata = test_set_balanced)
>
> confusionMatrix(svmtuned_predict_bal, test_set_balanced$booking_canceled)
Confusion Matrix and Statistics
Reference
Prediction    1    2
         1 4685  774
         2  750 2789

               Accuracy : 0.8306
                 95% CI : (0.8227, 0.8383)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6455

 Mcnemar's Test P-Value : 0.5558

            Sensitivity : 0.8620
            Specificity : 0.7828
         Pos Pred Value : 0.8582
         Neg Pred Value : 0.7881
             Prevalence : 0.6040
         Detection Rate : 0.5207
   Detection Prevalence : 0.6067
      Balanced Accuracy : 0.8224

       'Positive' Class : 1
```

Cost = 8

```
> mod_svmtuned_bal <- svm(booking_canceled~., data = train_set_balanced, epsilon = 0, cost = 8)
> svmtuned_predict_bal = predict(mod_svmtuned_bal, newdata = test_set_balanced)
>
> confusionMatrix(svmtuned_predict_bal, test_set_balanced$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 4725  766
         2  710 2797

               Accuracy : 0.836
                 95% CI : (0.8281, 0.8436)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6562

 Mcnemar's Test P-Value : 0.1523

            Sensitivity : 0.8694
            Specificity : 0.7850
         Pos Pred Value : 0.8605
         Neg Pred Value : 0.7975
             Prevalence : 0.6040
         Detection Rate : 0.5251
   Detection Prevalence : 0.6102
      Balanced Accuracy : 0.8272

       'Positive' Class : 1
```

Cost = 20

```
> mod_svmtuned_bal <- svm(booking_canceled~., data = train_set_balanced, epsilon = 0, cost = 20)
> svmtuned_predict_bal = predict(mod_svmtuned_bal, newdata = test_set_balanced)
>
> confusionMatrix(svmtuned_predict_bal, test_set_balanced$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 4738  740
         2  697 2823

               Accuracy : 0.8403
                 95% CI : (0.8326, 0.8478)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6655

 Mcnemar's Test P-Value : 0.2679

            Sensitivity : 0.8718
            Specificity : 0.7923
         Pos Pred Value : 0.8649
         Neg Pred Value : 0.8020
             Prevalence : 0.6040
         Detection Rate : 0.5266
   Detection Prevalence : 0.6088
      Balanced Accuracy : 0.8320

       'Positive' Class : 1
```

cost = 40

```
> mod_svmtuned_bal <- svm(booking_canceled~., data = train_set_balanced, epsilon = 0, cost = 40)
> svmtuned_predict_bal = predict(mod_svmtuned_bal, newdata = test_set_balanced)
>
> confusionMatrix(svmtuned_predict_bal, test_set_balanced$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 4762  728
         2  673 2835

               Accuracy : 0.8443
                 95% CI : (0.8366, 0.8517)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6736

 Mcnemar's Test P-Value : 0.1491

            Sensitivity : 0.8762
            Specificity : 0.7957
         Pos Pred Value : 0.8674
         Neg Pred Value : 0.8082
             Prevalence : 0.6040
         Detection Rate : 0.5292
   Detection Prevalence : 0.6101
      Balanced Accuracy : 0.8359

       'Positive' Class : 1
```

Validation with training set (cost = 20)

```
> svmtuned_train_bal = predict(mod_svmtuned_bal, newdata = train_set_balanced)
> confusionMatrix(svmtuned_train_bal, train_set_balanced$booking_canceled)
Confusion Matrix and Statistics

          Reference
Prediction     1     2
         1 11277  1555
         2  1407  6760

               Accuracy : 0.8589
                 95% CI : (0.8542, 0.8636)
    No Information Rate : 0.604
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7042

 Mcnemar's Test P-Value : 0.006913

            Sensitivity : 0.8891
            Specificity : 0.8130
         Pos Pred Value : 0.8788
         Neg Pred Value : 0.8277
             Prevalence : 0.6040
         Detection Rate : 0.5370
   Detection Prevalence : 0.6111
      Balanced Accuracy : 0.8510

       'Positive' Class : 1
```