HarvardX: PH125.9x - Capstone project 2 (Hotel Booking project)

Gerald Kandulu

May 31, 2020

1.0 Introduction

The aim of this project is to develop a model for predicting hotel booking cancelations. The machine learning algorithm is built from hotel booking demand dataset which contains bookings due to arrive between the 1st of July of 2015 and the 31st of August 2017, including bookings that effectively arrived and bookings that were canceled with all data elements pertaining to hotel or costumer identification deleted. The dataset was extracted from the hotels' Property Management System (PMS) SQL databases (Antonio, Almeida & Nunes, 2019). The dataset has 119, 390 observations with 32 variables for two hotels, the resort hotel and city hotel that are based in Portugal. The dataset was downloaded from Kaggle: https://www.kaggle.com/jessemostipak/hotel-booking-demand.

Considering the large number of features in the dataset and number of observations, the project used feature selection and training sample estimation approaches to ensure that the model developed is easier to interpret by researchers/users, has shortened training time, does not suffer from the curse of dimensionality, and has enhanced generalization by reducing overfitting. Other machine learning projects had also demostrated these benefits after using feature selection and smaller training set size (James et al., 2013 & Irizarry, 2020). The project used the F1 score to measure model performance because the dataset had an uneven class distribution. Also from the hotels' perspective, it is more important to predict the True Positives and True negatives to avoid wasting resources.

The dataset was downloaded from Kaggle and upload to Github, url: https://github.com/geraldlab/kaggle-data/raw/master/hotel-booking-demand.zip so that the data can easily be fetched for peer grading.

2.0 Methods and analysis

The project used the following R packages: caret, data.table, ggcorrplot, gridExtra, and tidyverse.

2.1 Data loading and cleaning

The code below downloads the data from github as indicated in the introductory section.

```r
#url for zip file at Github - geraldlab
url <- "https://github.com/geraldlab/Harvardx-capstone-project-2/raw/master/hotel-booking-demand.zip"

dl <- tempfile()

#download file in binary mode
download.file(url, dl,  mode="wb")

#unzip and read the file
```

```
hotel <- read.csv(unzip(dl, "hotel_bookings.csv"), stringsAsFactors = TRUE)


#clean the environment
rm(dl, url)
```

2.2 Exploratory data analysis

Inspection of the data from the glimpse see appendix 1 below, indicate that the dataset is as expected(119,390 observations and 32 variables). The code below indicate that the dataset has one variable “children” with null (NA) values.

```
#using sapply and is.na() function to identify columns with "NA"" values
names(which(sapply(hotel, function(x) any(is.na(x)))))
```

```
## [1] "children"
```

The glimpse output also indicated that some categorical variables were coerced to integer class. Below code fixes the variable classes, rename variables, and derives new variables to make the exploratory data analysis process efficient. The data dictionary for the data set can be accessed at Kaggle or from the paper published by the data collectors at: https://www.sciencedirect.com/science/article/pii/S2352340918315191.

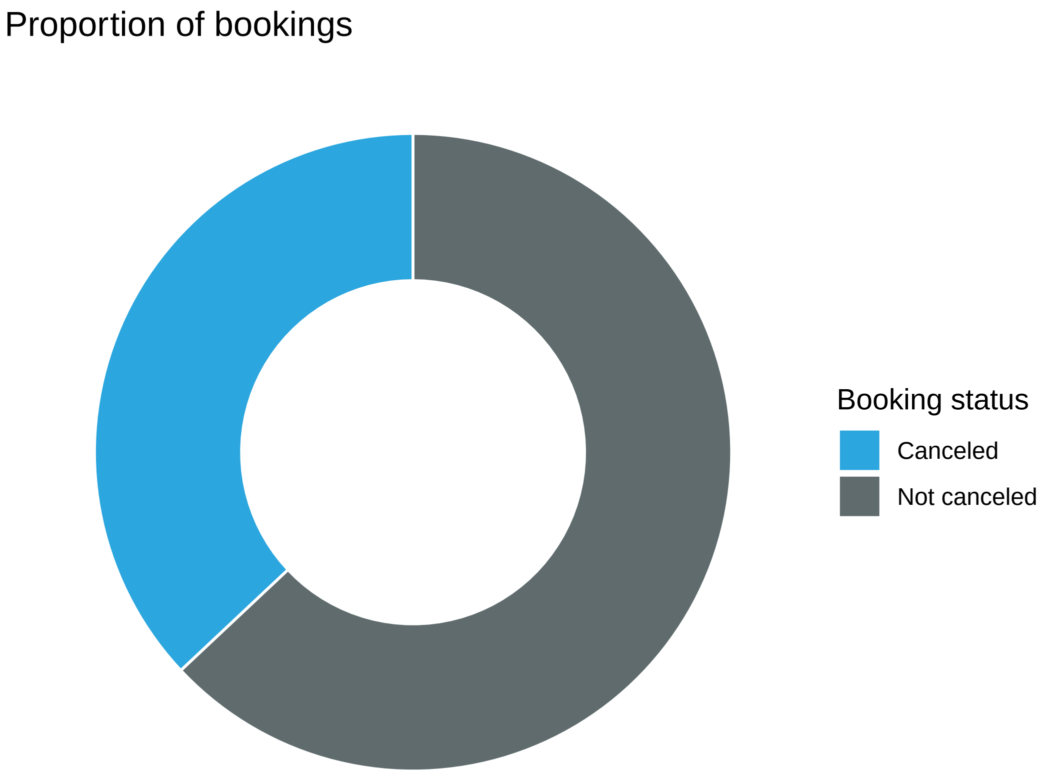
```
#make copy of dataset, copy of dataset run in appendix 1 below
hotel1 <- hotel

#convert the target column is_cancelled to factor Value
#if the booking was canceled (1) or not (0).
hotel$is_canceled <- as.factor(hotel$is_canceled)

#rename the adr column to meaningful name
hotel <- hotel %>% rename(average_daily_rate = adr)

#replace NA in children column
#replace ’NULL’ value with ’Undefined’ in agent and company columns
#add following variable:
#- To indicate the arrival date and month for booking
#     -arrival_date - derived from arrival_date_year, month, and date using lubridate "ymd" function
#     -arrival_month - derived from arrival_date using lubridate "month" function
#- To indicate if booking source is Portugal (1) or not (0)
#     - local_booking - factor (0, 1)

hotel <- hotel %>% mutate_if(is.factor, as.character) %>%
                   mutate(children = ifelse(is.na(children), 0, children) ,
                          agent = ifelse(agent == "NULL", "Undefined", agent),
                          company = ifelse(company == "NULL", "Undefined", company),
                          local_booking = ifelse(country == "PRT", 1, 0),
                          arrival_date = ymd(paste0(arrival_date_year, "/",
                                                    arrival_date_month , "/",
                                                 arrival_date_day_of_month)),
                          arrival_month = month(arrival_date, label = TRUE)) %>%
                   mutate_if(is.character, as.factor)
```

```
#convert int column to factor
hotel$local_booking <- as.factor(hotel$local_booking)
```

Review of the data indicate that the “reservation_status” column promotes leakage of future information hence it was not considered for the exploratory data analysis and training of the model.

Below code plots the proportion of the bookings using the target variable “is_canceled”.

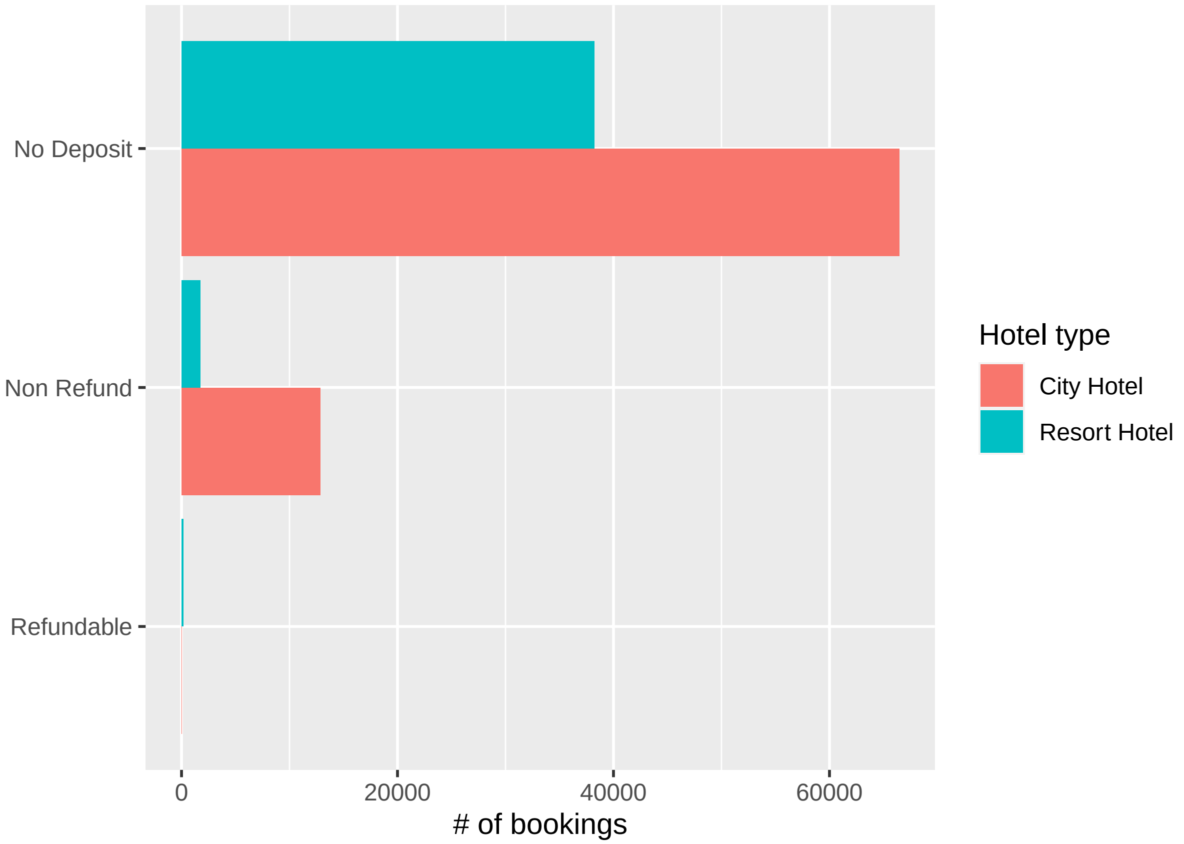
```
#plot pie chart for showing proportion of the target variable
hotel %>% summarise (Canceled = round(mean(is_canceled == 1), 3) * 100,
                     Not_canceled = round(mean(is_canceled == 0), 3) * 100) %>%
          pivot_longer(cols = c("Canceled",  "Not_canceled"),
                       names_to = "variable", values_to = "value") %>%
          mutate(variable = str_replace_all(variable, "_", " "),
                 lab.ypos = cumsum(value) - 0.5*value ) %>%
          arrange(value) %>%
          ggplot(aes(x = 2, y = value, fill = variable, label = value)) +
          geom_bar(stat = "identity", color = "white") +
          #geom_text(aes(y = lab.ypos), color = "white") +
          coord_polar(theta = "y", start = 0) + theme_void() + xlim(0.5, 2.5) +
          scale_fill_manual (values = c("Canceled" = "#2BA6DE", "Not canceled" = "#5F6B6D")) +
          labs(fill = "Booking status")  + ggtitle("Proportion of bookings")
```



From the above plot, about 37 % of the booking for the city and resort hotels were canceled. The next variable that was explored was the distribution of the type of deposit used on the bookings.

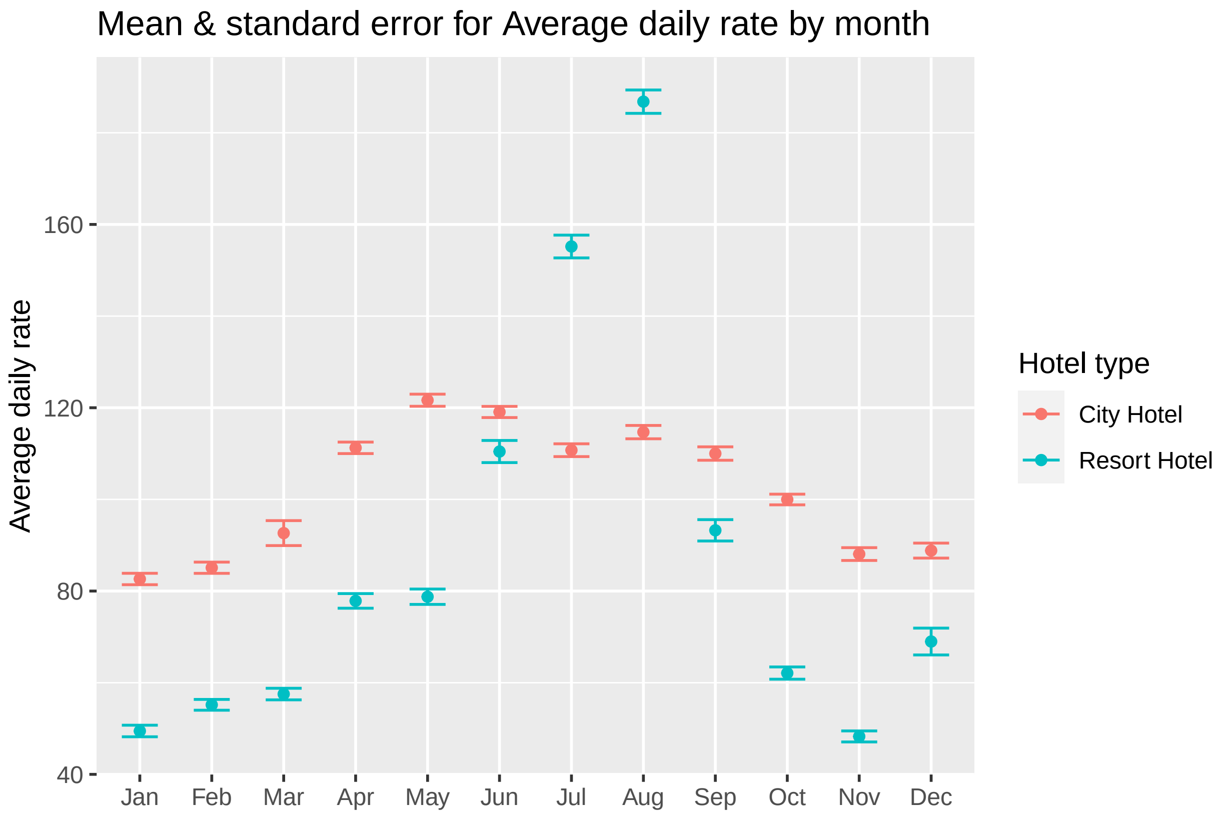
As can be observed from below plot, majority of the bookings had no deposit for both city and resort hotels. Also, overall about 0.14 % of the total bookings paid a refundable deposit.

```
#plot distribution of deposit type
hotel %>% select(hotel, deposit_type) %>%
                group_by( hotel, deposit_type) %>%
                summarise(n = n()) %>%
                ggplot(aes(reorder(deposit_type, n), n, fill = hotel)) +
                geom_bar(stat = "identity", position = position_dodge()) + xlab("") +
                ylab("# of bookings") + labs(fill = "Hotel type") +
                coord_flip()
```



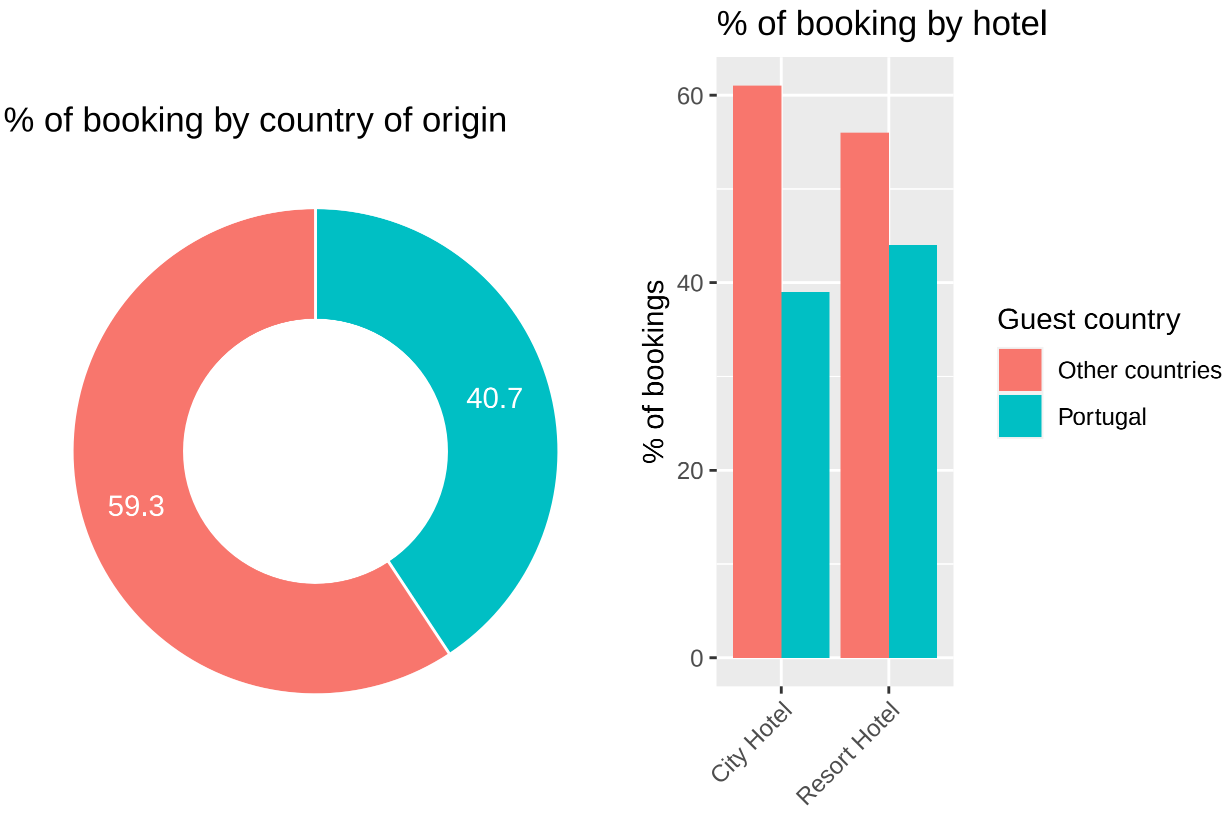
The plot below for mean average daily rate per month by hotel type indicate huge variations by months. Overall, the city hotel has a higher mean average daily rate than the resort hotel except the month of July. This is expected as most people tend to visit resorts hotels in summer months, with July being the busiest month.

```
#plot the mean and standard error for "average daily rate""
hotel %>% select(arrival_month, hotel,  average_daily_rate) %>%
          pivot_longer(cols = c("average_daily_rate"),
                        names_to = "variable", values_to = "value") %>%
          mutate(variable = str_replace_all(variable, "_", " ")) %>%
          group_by(arrival_month, hotel, variable) %>%
          summarise(mu = mean(value),
                    se = sd(value)/sqrt( n())) %>%
          ggplot(aes(arrival_month, mu, ymin= mu - 3*se, ymax = mu + 3*se, color = hotel)) +
          geom_point() + geom_errorbar(aes(width = 0.5)) +
          xlab("") + ylab("Average daily rate") +
          labs(color = "Hotel type") +
          ggtitle("Mean & standard error for Average daily rate by month")
```



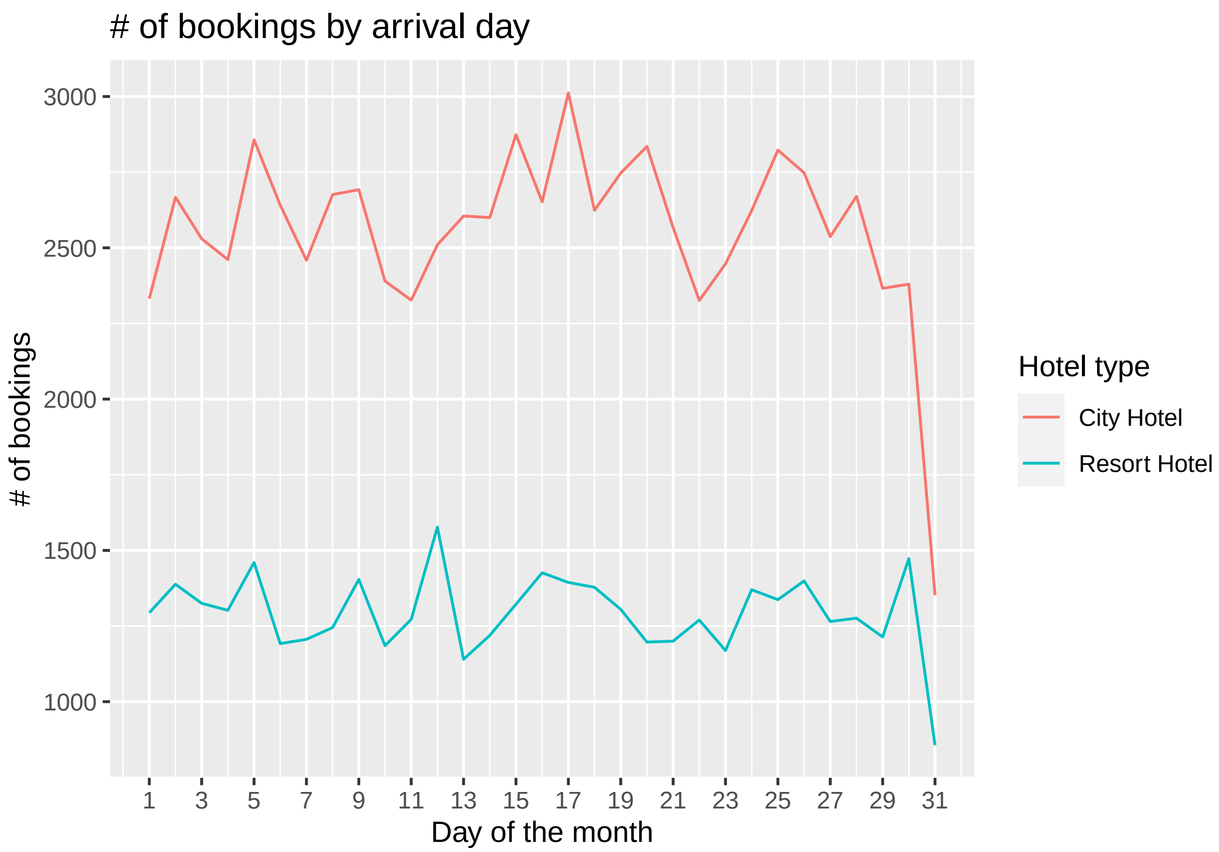
Guests from Portugal account for 40.7 percent of the bookings in the data set. Portuguesse booking are also lower compared from booking from outside Portugal in both hotels. Below code plots both charts.

```
#plot pie chart for overall booking by source
p1 <- hotel %>% summarise (Portugal = round(mean(country == "PRT"), 3) * 100,
                     Other_countries = round(mean(country != "PRT"), 3) * 100) %>%
                pivot_longer(cols = c("Portugal",  "Other_countries"),
                             names_to = "variable", values_to = "value") %>%
                mutate(variable = str_replace_all(variable, "_", " "),
                lab.ypos = cumsum(value) - 0.5*value ) %>%
                arrange(value) %>%
                ggplot(aes(x = 2, y = value, fill = variable, label = value)) +
                geom_bar(stat = "identity", color = "white") +
                geom_text(aes(y = lab.ypos), color = "white") +
                coord_polar(theta = "y", start = 0) + theme_void() + xlim(0.5, 2.5) +
                theme(legend.position = "none") + ggtitle("% of booking by country of origin")

#plot booking by source disaggregated by hotel type
p2 <- hotel %>% group_by(hotel) %>% summarise (Portugal = round(mean(country == "PRT"), 3) * 100,
                                               Other_countries = round(mean(country != "PRT"), 3) * 100
                pivot_longer(cols = c("Portugal",  "Other_countries"),
                             names_to = "variable", values_to = "value") %>%
                mutate(variable = str_replace_all(variable, "_", " ")) %>%
                arrange(value) %>%
                ggplot(aes(hotel, y = value, fill = variable, label = value)) +
                geom_bar(stat = "identity", position = position_dodge()) +
                theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
                xlab("") + ylab("% of bookings") +
                labs(fill = "Guest country") + ggtitle("% of booking by hotel")

#show plots in grid
grid.arrange(p1, p2, ncol = 2)
```



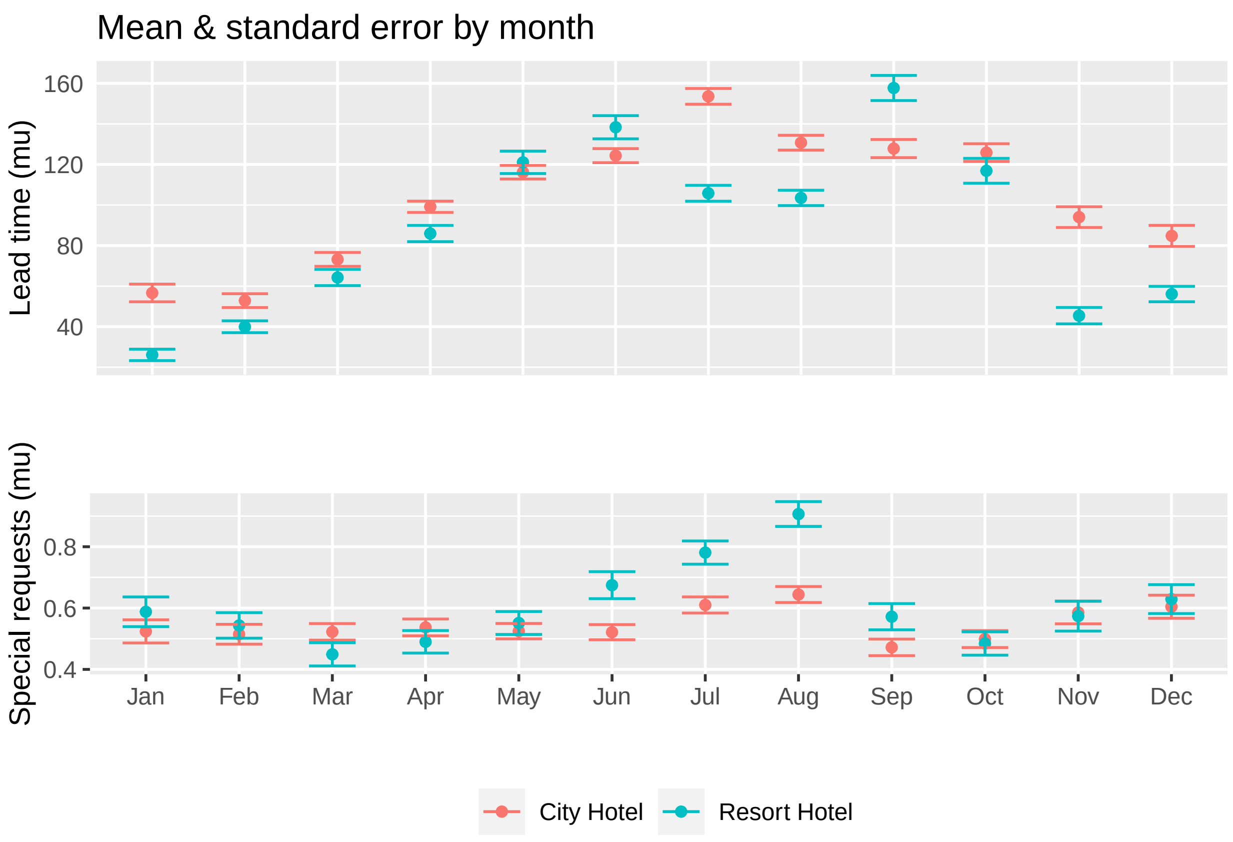
Code below plots the number of bookings by the arrival day of the month. The plot indicate that most booking have the arrival day of 17th and 12th for city hotel and resort hotel respectively.

```
#plot # of bookings by day of the month of the arrival date
hotel %>% group_by(hotel, arrival_date_day_of_month) %>%
          summarise(n = n()) %>%
          ggplot(aes(arrival_date_day_of_month, n, color = hotel)) +
          geom_line() + scale_x_continuous(breaks = seq(1, 31, 2)) +
          ylab("# of bookings") + xlab("Day of the month") +
          labs(color = "Hotel type") + ggtitle("# of bookings by arrival day")
```



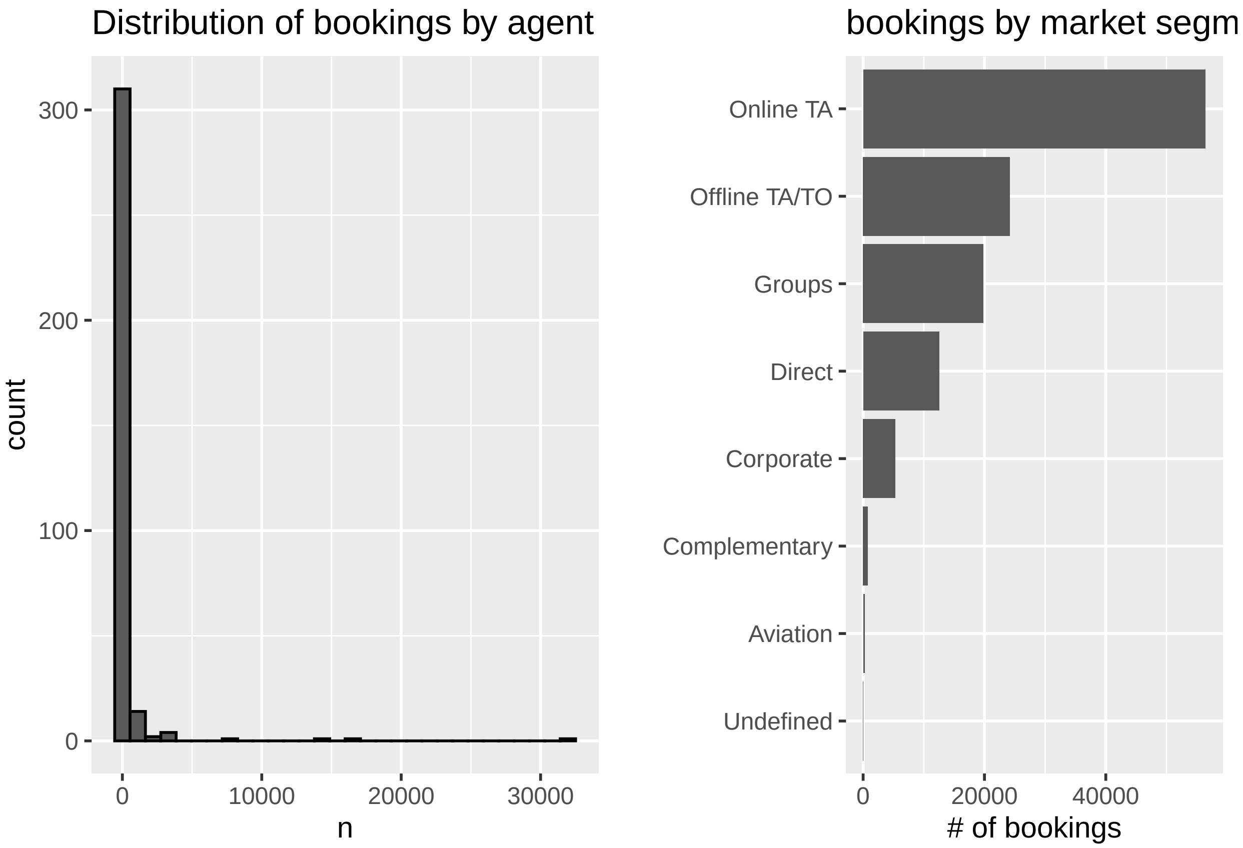
The data also indicate that most of the bookings are recording in the system at least a month (lead_time) before the arrival day. The highest mean lead time for booking at City hotel is 153.5 days in July while Resort hotel is 157.7 days in September. On average, August is the month with most special requests at both hotels. Plot for mean lead time and number of special requests is below.

```
#plot lead time
p1 <- hotel %>% select(arrival_month, hotel, lead_time) %>%
                      pivot_longer(cols = c( "lead_time"),
                              names_to = "variable", values_to = "value") %>%
                      mutate(variable = str_replace_all(variable, "_", " ")) %>%
                      group_by(arrival_month, hotel, variable) %>%
                      summarise(mu = mean(value),
                                se = sd(value)/sqrt( n())) %>%
                      ggplot(aes(arrival_month, mu, ymin= mu - 3*se, ymax = mu + 3*se, color = hotel)) +
                      geom_point() + geom_errorbar(aes(width = 0.5)) +
                      ylab("Lead time (mu)") + xlab("") + theme(legend.position = "none",
                                                        axis.text.x = element_blank(),
                                                        axis.ticks = element_blank()) +
                      ggtitle("Mean & standard error by month")

#plot # of special requests
p2 <- hotel %>% select(arrival_month, hotel, total_of_special_requests) %>%
                    pivot_longer(cols = c("total_of_special_requests"),
                             names_to = "variable", values_to = "value") %>%
                    mutate(variable = str_replace_all(variable, "_", " ")) %>%
                    group_by(arrival_month, hotel, variable) %>%
                    summarise(mu = mean(value),
                              se = sd(value)/sqrt( n())) %>%
                    ggplot(aes(arrival_month, mu, ymin= mu - 3*se, ymax = mu + 3*se, color = hotel)) +
                    geom_point() + geom_errorbar(aes(width = 0.5)) +
                    xlab("") + ylab("Special requests (mu)") + labs(color = "Hotel type")  +
                    labs(color = "") + theme(legend.position = "bottom") + ggtitle("")



#show plots in grid
grid.arrange(p1, p2, nrow = 2)
```



The plots for distribution by agent and market segments are ploted by below code. The distribution of bookings by agent is right skewed. Also, Online TA (Travel agents), and Offline TA/TO (Travel agents/ Tour operators) having the most bookings.

```
#plot of distribution agent
p1 <- hotel %>% group_by(agent) %>%summarise(n = n())%>%
                ggplot(aes(n)) +
                geom_histogram(bins = 30, color = "black") +
                ggtitle("Distribution of bookings by agent")

#plot distribution of market segment
p2 <- hotel %>% select(hotel, market_segment) %>%
                group_by( hotel, market_segment) %>%
                summarise(n = n()) %>%
                ggplot(aes(reorder(market_segment, n), n)) +
                geom_bar(stat = "identity") + xlab("") + ylab("# of bookings") +
                coord_flip() + ggtitle("bookings by market segment")


#show plots in grid
grid.arrange(p1, p2, ncol = 2)
```



2.3 Modeling machine learning algorithm

As observed in the data exploration section, the data set has quite a large number of dimensions. To avoid the curse of dimensionality and ensure a shortened training time, the following sections discusses the feature selection and training set size estimation methods used by the project.

2.3.1 Feature selection

From the dataset, we notice that some features have a potential to leak information of the target variable (is_canceled) for the learning model. These features ("reservation_status" and "reservation_status_date") were removed from the model in the code below. Removing features that promote leakage of future information is an approach that has been recommended by other researchers(Abbott, 2014). Redundant features ("country", "arrival_date", and "arrival_month") were also removed from the data set.

```
#Remove redundant features and those that promote information leakage
hotel <- hotel %>% select(-c(reservation_status, reservation_status_date,
                             country, arrival_date, arrival_month))
```

Other researchers have also suggested using the correlation of the target variable to the other features and correlation among features as good approaches for reducing dimensionality (Blessie & Karthikeyan, 2012).

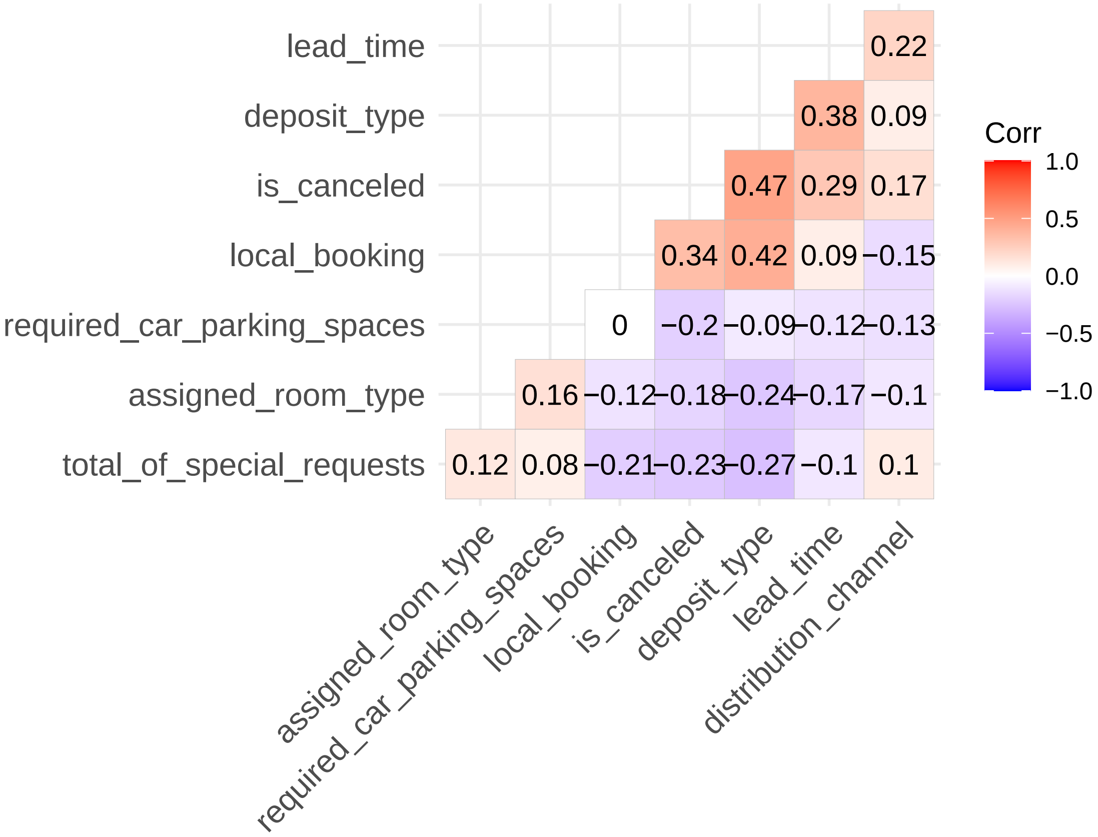
Code below extract list of features that have a +/- 0.15 correlation to the target variable ("is_canceled").

```r
#convert all factors to numeric
tt <- hotel %>% mutate_if(is.factor, as.numeric)

#calculate correlation coefficient between target variable and the other variables
cor_is_canceled <-  data.frame ( sapply(tt, cor, y=tt$is_canceled))

#raname table
colnames(cor_is_canceled) <- "corr_coeff"

#select only those with corr_coeff < -0.15 | corr_coeff > 0.15
cor_is_canceled <- cor_is_canceled %>%  rownames_to_column(., var = "feature") %>%
                                        filter(corr_coeff < -0.15 | corr_coeff > 0.15) %>%
                                        arrange(desc(corr_coeff))


#subset the tibble with only features that are strongly correlated to the target feature
tt <- tt %>%select(cor_is_canceled$feature)

#print the table
cor_is_canceled  %>% knitr::kable()
```

feature	corr_coeff
is_canceled	1.0000000
deposit_type	0.4686338
local_booking	0.3361218
lead_time	0.2931234
distribution_channel	0.1676003
assigned_room_type	-0.1760277
required_car_parking_spaces	-0.1954978
total_of_special_requests	-0.2346578

The above table indicate 7 features has a correlation of at least +/- 0.15 to the target variable. Below code investigates how correlated these 7 feature are:

```r
#calculate correlation between the variable and plot the matrix
 cor(tt) %>% ggcorrplot(., hc.order = TRUE, type = "lower",
                        lab = TRUE)
```



The above correlation matrix indicate that the features that had a correlation of at least +/- 0.15 to the target variable are not strong correlated to each other.

2.3.2 Estimating training set size

The data with the features selected in section 2.3.1 is split into training and validation sets with 80:20 split.

```
tt <- hotel %>%select(cor_is_canceled$feature)
#set
set.seed(1)

#create the 80: 20 index
test_index <- createDataPartition(y = tt$is_canceled, times = 1, p = 0.2, list = FALSE)

#training and text set
df_train <- tt[-test_index,]
df_validation <- tt[test_index,]

#view the dimensions of the training set.
dim(df_train)
```

```
## [1] 95511     8
```

Splitting the data into 80:20 for training and validation sets results in a training set (df_train) of 95,512 observations. Training a model on a training set of this magnitude is computationally expensive and some of the classifacation algorithms may results in unresponsiveness on a standard computer. Inaddition, other researchers observed that larger training samples results in overfitting of the learning model and a learning classifier learning curves generally follows an inverse power law (Beleites et al, 2013, & Irizarry, 2020).

As describe by Figueroa et al. (2012) that a learning curve is a collection of data points (xj, yj) that describe how the performance of a classifier (yj) is related to training sample sizes (xj) with j = 1 to m as the total number of instances. They observed that a learning curve has 3 major phases. In the first phase, performance increases rapidly with an increase in training set size; second phase being charactirised with less rapid improvements and then in the final phase the classifier reaches its efficiency threshold, i.e. no (or only marginal) improvement in performance is observed with increasing training set size.

The project used the F1 score to measure model performance and not "Accurancy" for the following reasons. As observed in the data exploration process above, the target variable('is_canceled') is imbalanced (Not canceled = 63%, canceled = 37 %). Irizarry (2020) indicated that F1 score is usually more useful than accuracy, especially when there is an uneven class distribution. Also from the hotels' perspective, it is more important to predict the True Positives and True negatives because this will help them plan their operations better and minimise loss of resources.

For classification problems a model is considered to be performing well if the F1 score is closer to 1. With this in mind, a base model was developed by randomly sampling the samples using the proportions of the target variable on the training set. Code below calculates the F1 score of the base model.

```r
#proportions of is_canceled
p <- mean(df_train$is_canceled == 1)

#predict the class using the propability of the target features.
naive_pred <- as.factor(sample(c(1, 0), nrow(df_train), prob=c(p, 1 - p), replace=TRUE))

# Use confusion matrix to calculate base model accurancy
naive_pred <- confusionMatrix(naive_pred, df_train$is_canceled)

#display the F1 score
round(naive_pred$byClass["F1"], 4)
```

```
##     F1
## 0.6323
```

The base model F1 score is 0.6291.

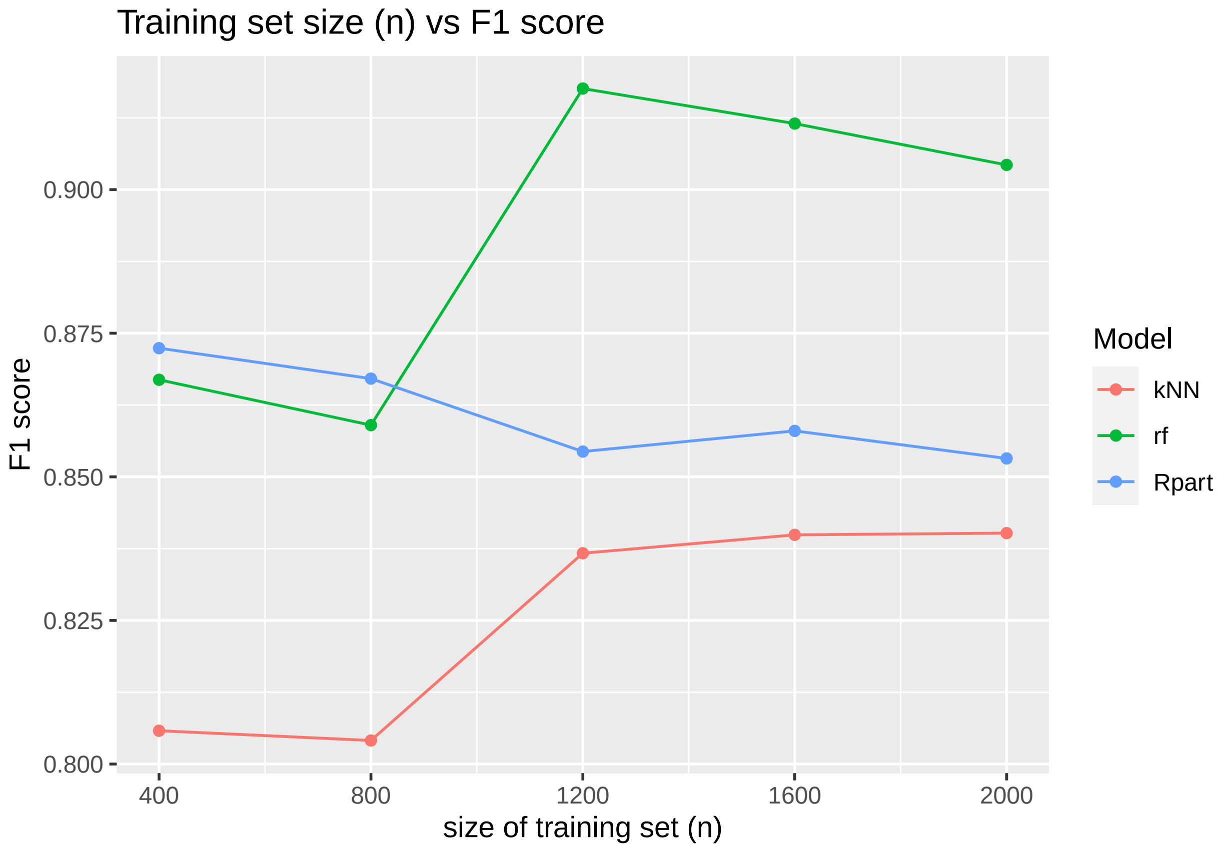
The following classication algorithms: k-nearest neighbors (kNN), classification tree (rpart), and random forest(rf) were used on the training data with 10-fold cross validation. The algorithm with highest F1 score was used on the validation set to validate the learning model. The use of cross validation on training set and use of the kNN, rpart, and rf machine learning algorithms were also identified by Irizarry (2020) as appropiate for classification problems.

Using progressive sampling with maximum sample size n = 2000, a learning curve with increments of 400 is implemented below. A maximum sample size of 2000 was selected as various researchers have indicated that sample sizes of at least 1000 observations usually results in statistically signicant results (Figueroa et al., 2012), and are not computationally expensive.

```r
#! warning this section of code takes abit longer (3 - 7 minutes)
#minimum, maximum, and increments for estimating  training set size
estimate_size  <- seq(400, 2000, 400)


# set tuning parameters
control <- trainControl(method="cv", number=10)
```

```
metric <- "Accuracy"

#function to train the models with 10-fold cross validation
train_model <- function(n){
  set.seed(1)
  #sample n row from training
  index <- sample(nrow(df_train), n)

  #subset
  tt1 <- df_train[index,]

  #train model
  # kNN
  set.seed(1)
  fit.knn <- train(is_canceled~., data=tt1, method="knn", metric=metric, trControl=control)
  pred.kNN <-predict(fit.knn, tt1)

  # classification tree (rpart)
  set.seed(1)
  fit.rpart <- train(is_canceled~., data=tt1, method="rpart", metric=metric, trControl=control)
  pred.rpart <-predict(fit.rpart, tt1)

  # # Random Forest
  set.seed(1)
  fit.rf <- train(is_canceled~., data=tt1, method="rf", metric=metric, trControl=control)
  pred.rf <-predict(fit.rf, tt1)


  # # return the f1 scores
  # "KNN", "Rpart", "rf"
  return(tibble(n = nrow(tt1),
                kNN = confusionMatrix(pred.kNN, tt1$is_canceled)$byClass[["F1"]],
                Rpart = confusionMatrix(pred.rpart, tt1$is_canceled)$byClass[["F1"]],
                rf = confusionMatrix(pred.rf, tt1$is_canceled)$byClass[["F1"]])
  )

}

#call function to train the model using sapply function and store F1 scores
models_accurancies <- data.frame(sapply(estimate_size, train_model))

#convert row names to column names and vice-versa
models_accurancies <- models_accurancies %>% rownames_to_column() %>%
                                             gather(variable, value, -rowname) %>%
                                             spread(rowname, value) %>%  select(-variable)


#convert all columns to numeric
i <- seq(1, ncol(models_accurancies), 1)
models_accurancies[ , i] <- apply(models_accurancies[ , i], 2,
                                  function(x) as.numeric(as.character(x)))

#convert tibble to tidy format
```

```
models_accurancies <- models_accurancies %>% select("n", "kNN", "Rpart",  "rf") %>%
                      pivot_longer(cols = c( "kNN", "Rpart",  "rf"),
                                   names_to = "Model", values_to = "F1_score") %>%
                      mutate(F1_score = round(F1_score, 4))

#plot F1 score vs size of training set
models_accurancies %>%  ggplot(aes(n, F1_score, color = Model)) + geom_line() + geom_point() +
                        xlab("size of training set (n)") + ylab("F1 score") +
                        scale_x_continuous(breaks= estimate_size) +
                        ggtitle("Training set size (n) vs F1 score")
```

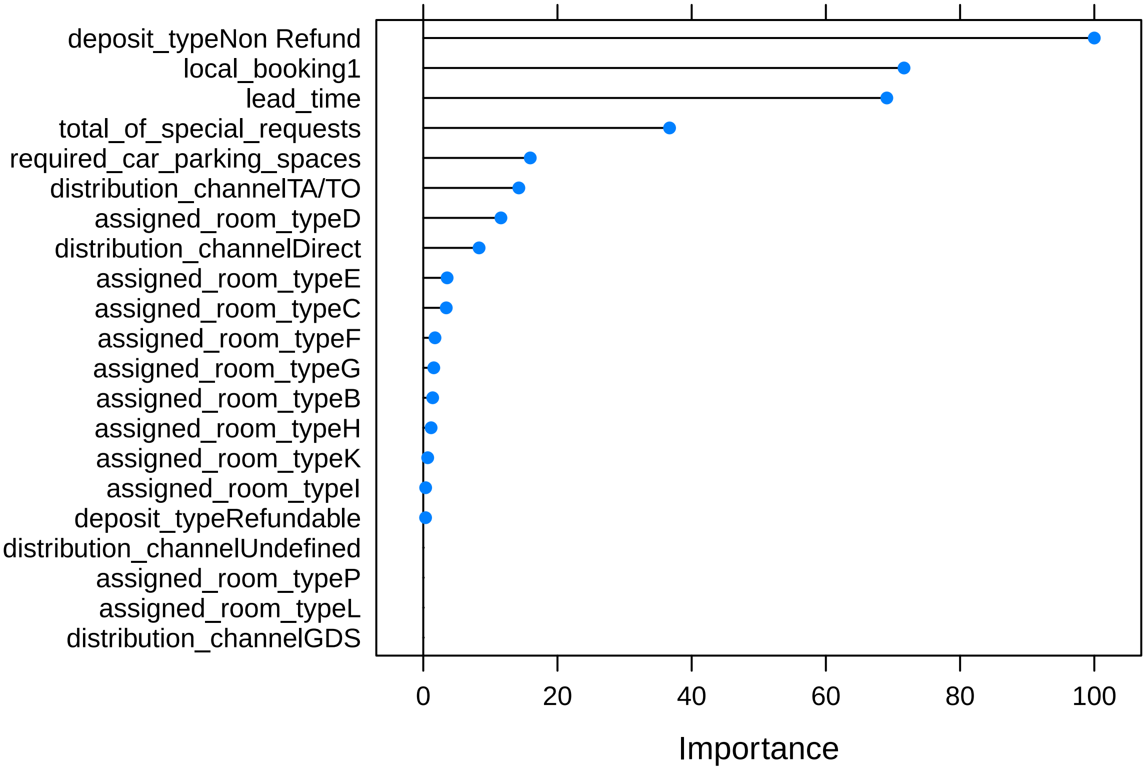


From the F1 score and size of training set plot above, the highest accurancy is achieved at 0.9176 by random forest (rf) with a training data set size of 1,200. The rf model also performed better than the base model that has F1 score of 0.6291.

```
#algorithm with highest accurancy
models_accurancies %>% filter(F1_score == max(F1_score)) %>%
                       select(Model, F1_score, n)  %>% knitr::kable()
```

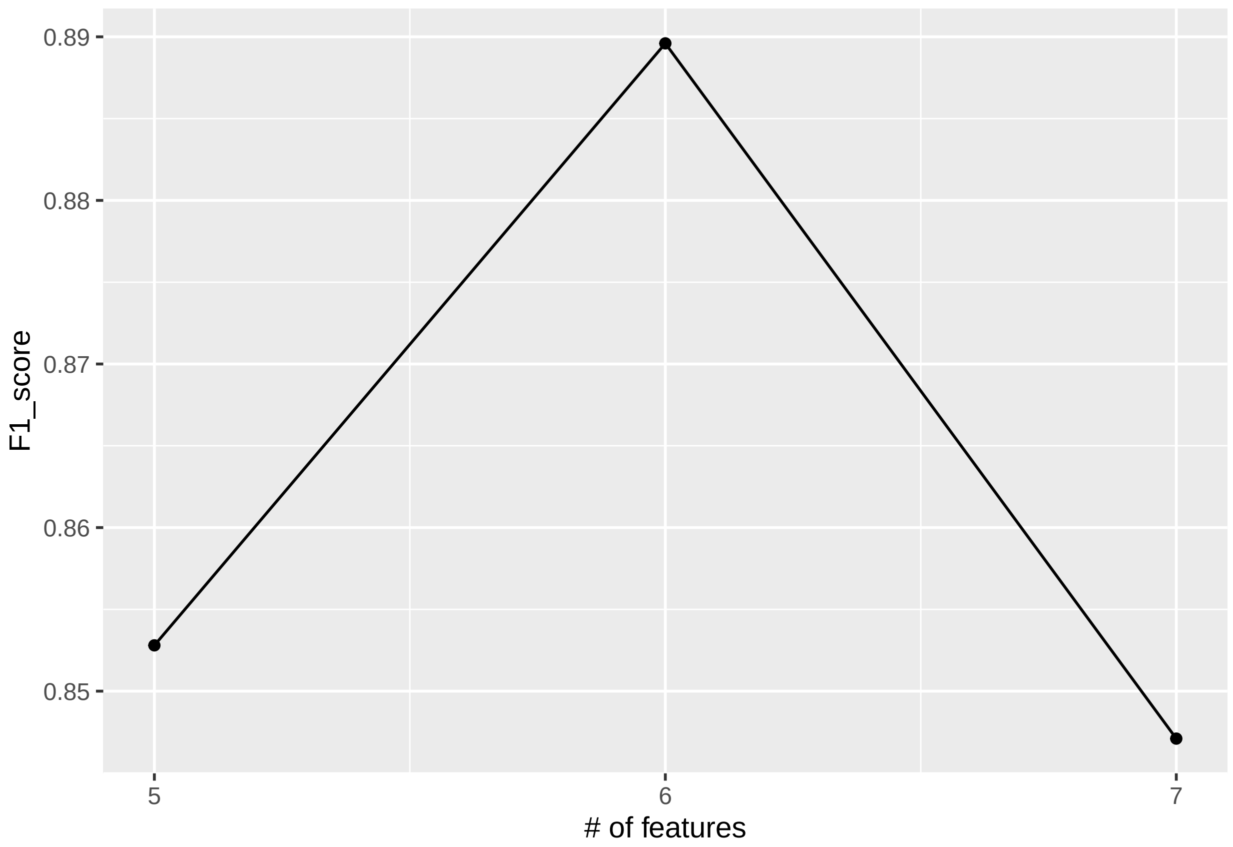
Model	F1_score	n
rf	0.9176	1200

```
#get n with max F1 score
n <- models_accurancies %>% filter(F1_score == max(F1_score)) %>% pull(n)

#sample n rows from training set
index <- sample(nrow(df_train), n)

#subset the training dataset
tt <- df_train[index,]

#train the final model
set.seed(1)
fit.rf <- train(is_canceled~., data=tt, method="rf", metric=metric, trControl=control)

pred.rf <-predict(fit.rf, tt)


#add the model result to table
final_model <- data.frame(model = "rf",
                          F1_score = round(confusionMatrix(pred.rf, tt$is_canceled)$byClass[["F1"]], 4)
                          number_of_feature = ncol(tt) - 1)

#plot the var importance
plot(varImp(fit.rf))
```



The code below optimizes the random forest model using information from importance variable importance plot above.

```
#! warning code takes a bit longer 3 - 5 minutes
#columns with low var Importance less than 30
varImpo_colsL <- c("required_car_parking_spaces", "distribution_channel")

#columns with higher varImp >30
varImpo_colsH <- c("is_canceled", "deposit_type", "local_booking",
                   "lead_time", "total_of_special_requests")

#train final model to optiomal # of features by loading features in desc order of varImp
for (c in varImpo_colsL){
  varImpo_colsH <- c(varImpo_colsH, c)

  #subset training set
  tt1 <- tt %>% select(all_of(varImpo_colsH))


  #train the final model
  set.seed(1)
  fit.rf1 <- train(is_canceled~., data=tt1, method="rf", metric=metric, trControl=control)
  pred.rf1 <-predict(fit.rf1, tt1)


  #add the model result to table
  final_model <- bind_rows(final_model,
                           data.frame(model = "rf",
                                      F1_score = round(confusionMatrix(pred.rf1, tt1$is_canceled)$byClas
                                      number_of_feature = ncol(tt1) - 1)
                          )
}

#plot number of features vs F1_score
final_model %>% ggplot(aes(number_of_feature, F1_score)) +
                geom_line() + geom_point() + scale_x_continuous(breaks=seq(5, 7, 1)) +
                xlab("# of features")
```



The final model with trimmed features (6 features) has an F1 score of 0.8896 which is more than the initial final model with 7 features. The final model with 6 feature will be used below to validate the model on the validation set.

```r
#retrain model with 6 features to be used on validation set
varImpo_colsH <- c("is_canceled", "deposit_type", "local_booking", "lead_time",
                   "total_of_special_requests", "required_car_parking_spaces",
                   "distribution_channel")

#train model to final optiomal # of features
tt1 <- tt %>% select(all_of(varImpo_colsH))

#train the final model
set.seed(1)
fit.final_rf.model <- train(is_canceled~., data=tt1, method="rf", metric=metric, trControl=control)
```

3.0 Results

Using the final optimized random forest model with 6 features, the code below calculates the F1 score on the validation set.

```r
#predict target variables on validation set
prediction <-predict(fit.final_rf.model, df_validation)

#get confusion matrix
conf_matrix <- confusionMatrix( prediction, df_validation$is_canceled)
```

```r
#display F1 score on validation set
round(conf_matrix$byClass["F1"], 4)
```

```
##     F1
## 0.8215
```

The F1 score on the validation set is: 0.8215 and other information of the confusion matrix are indicated in Appendex 2.

4.0 Conclusion

This project used feature selection and training sample estimation methods to ensure that the model developed had a shortened training time, avoid the curse of dimensionality, and enhance generalization by reducing overfitting. The training and validation sets were split using an 80:20 split.This resulted in a training set with 95,512 observations and 31 features. The use of feature selection and sample estimation resulted in a training set with 6 features and 1,200 observations of training set data. Reducing number of features and size of training set size was done to ensure that the final model has shortened training time, avoids curse of dimensionality, and overfitting. The final machine learning model was trained on the reduced train set using 10 cross validation. Three classifacation learning algorithms: k-nearest neighbors (kNN), classification tree (rpart), and random forest(rf), were considered in the project. Tests on the training set indicated that random forest(rf) performed better than the other two models (kNN, and rpart).

We believe the work in this project has contributed to the body of machine learning for classification problems by demonstrating methods that can be used to select features, and estimate size of training set. Also, the project has demostrated that random forest performed better on the dataset than rpart and kNN. Use of other approaches and/ or sophisticated algorithms have the potential of improving the learning model. Inaddition, future work could consider using other sophisticated machine learning algorithms to select optimal features, training set size, and improve the model performance.

5.0 Reference

Abbott, D. (2014), "Applied Predictive Analytics: Principles and Techniques for the Professional Data Analyst" Wiley, Indianapolis, IN, USA

Antonio, N, Almeida, A. & Nunes, L (2019) "Hotel Booking Demand Datasets", url: https://www.sciencedirect.com/science/article/pii/S2352340918315191

Blessie, E. C. and Karthikeyan, E. (2012) 'Sigmis: A Feature Selection Algorithm Using Correlation Based Method', Journal of Algorithms & Computational Technology, pp. 385–394. doi: 10.1260/1748-3018.6.3.385.

Beleites, C. Neugebauer, U. Bocklitz, T. Kraff, C. & Popp, J. (2013) 'Sample size planning for classification models', Analytica Chimica Acta, 760, pp 25 - 33, https://doi.org/10.1016/j.aca.2012.11.007

Figueroa, R. L., Zeng-Treitler, Q., Kandula, S., & Ngo, L. H. (2012) 'Predicting sample size required for classification performance', BMC medical informatics and decision making, 12 (8), https://doi.org/10.1186/1472-6947-12-8

Irizarry, R.A. (2020), "Introduction to Data Science Data Analysis and Prediction Algorithms with R", (url: https://rafalab.github.io/dsbook/)

James, G., Witten, D., Hastie, T & Tibshirani, R. 2013, An Introduction to Statistical Learning with Applications in R, 7th edition, Springer

6.0 Appendix

1.0 Glimpse dataset

```
#glimple the data
glimpse(hotel1)
```

```
## Rows: 119,390
## Columns: 32
## $ hotel                          <fct> Resort Hotel, Resort Hotel, Resort H...
## $ is_canceled                    <int> 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, ...
## $ lead_time                      <int> 342, 737, 7, 13, 14, 14, 0, 9, 85, 7...
## $ arrival_date_year              <int> 2015, 2015, 2015, 2015, 2015, 2015, ...
## $ arrival_date_month             <fct> July, July, July, July, July, July, ...
## $ arrival_date_week_number       <int> 27, 27, 27, 27, 27, 27, 27, 27, 27, ...
## $ arrival_date_day_of_month      <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ stays_in_weekend_nights        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ stays_in_week_nights           <int> 0, 0, 1, 1, 2, 2, 2, 2, 3, 3, 4, 4, ...
## $ adults                         <int> 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, ...
## $ children                       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ babies                         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ meal                           <fct> BB, BB, BB, BB, BB, BB, BB, FB, BB, ...
## $ country                        <fct> PRT, PRT, GBR, GBR, GBR, GBR, PRT, P...
## $ market_segment                 <fct> Direct, Direct, Direct, Corporate, O...
## $ distribution_channel           <fct> Direct, Direct, Direct, Corporate, T...
## $ is_repeated_guest              <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ previous_cancellations         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ previous_bookings_not_canceled <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ reserved_room_type             <fct> C, C, A, A, A, A, C, C, A, D, E, D, ...
## $ assigned_room_type             <fct> C, C, C, A, A, A, C, C, A, D, E, D, ...
## $ booking_changes                <int> 3, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ deposit_type                   <fct> No Deposit, No Deposit, No Deposit, ...
## $ agent                          <fct> NULL, NULL, NULL, 304, 240, 240, NUL...
## $ company                        <fct> NULL, NULL, NULL, NULL, NULL, NULL, ...
## $ days_in_waiting_list           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ customer_type                  <fct> Transient, Transient, Transient, Tra...
## $ adr                            <dbl> 0.00, 0.00, 75.00, 75.00, 98.00, 98....
## $ required_car_parking_spaces    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ total_of_special_requests      <int> 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, ...
## $ reservation_status             <fct> Check-Out, Check-Out, Check-Out, Che...
## $ reservation_status_date        <fct> 2015-07-01, 2015-07-01, 2015-07-02, ...
```

2.0 Confusion matrix for final model on validation set

```
#dislay the confusion matrix by class values
conf_matrix$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.8796727            0.5545506            0.7704632
##       Neg Pred Value            Precision               Recall
##            0.7305630            0.7704632            0.8796727
##                   F1           Prevalence       Detection Rate
##            0.8214541            0.6295909            0.5538339
## Detection Prevalence    Balanced Accuracy
##            0.7188324            0.7171117
```