Machine Learning

(Course 41204)

Winter 2022

Professor: Mladen Kolar

Prediction of hotel bookings cancellation using Machine Learning

Team members:

Xingyue Fang

Xinyi Gu

Guillermo Trefogli

I. Introduction

Goal:

The main goal of this paper is to predict Hotel bookings cancellation for clients to reduce the loss caused by this problem. We apply machine learning techniques to approach this problem. We perform this analysis using a real business dataset (available at kaggle.com) containing 31 variables and almost 120,000 observations. We build several models and choose the best one to predict the probability of booking cancellations. Hotel business can choose an optimal level of threshold for themselves to maximize the profits according to its own revenues and costs using this model. By doing so, they can improve the administration of bookings and minimize loss. We believe that this analysis is relevant given that this data is common to be held by companies in this industry, which makes feasible its application to solve real world problems in the industry. Finally, Hotel industry companies can update the performance for setting predictive models following the same reasoning applied in this paper.

Background: The Business Problem

Optimizing the management of bookings is a key piece of the management of revenues in the hotel industry. Nuno and others (2017), explain well the process of booking in the context of the model of the business: it consists of allowing customers to ensure, in advance, the best service to satisfy their needs. By this way, booking represents a contract signed by the company and the customers in which, usually, the latter have the right to cancel the service. This characteristic translates in risk of loss for the company: there is the possibility of incurring in costs (mainly, fixed costs) while not receiving any revenues due to cancellation: The hotel owner could only optimize revenues if it operates at its full capacity.

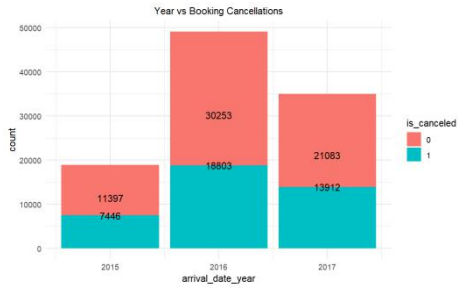
In addition to understand the business problem conceptually, its relevance can be explored for the case st hand as well. For example, we can account for the number of cancellations for the hotels through the time: on average, cancellations have represented the 37% for the total annual number of bookings in the hotels under analysis, for the period under analysis (2015-2017). Undoubtedly, discussing the fraction of unused capacity might be a best proxy for the relevance of booking cancellations, but these number give some initial insights to understand the importance of the subject.

II. Data Summary and Pre-processing

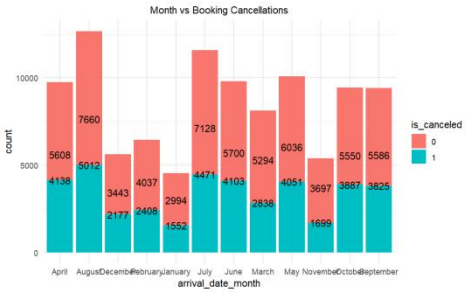
Data Summary

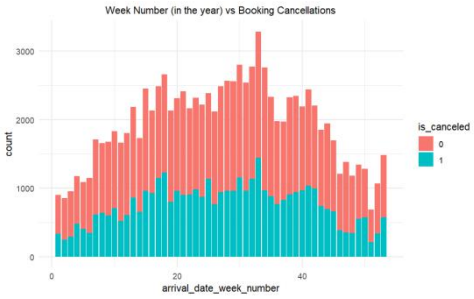
As mentioned, the dataset contains information for a given booking (unit of observation) related to characteristics of the booking (hotel type, date, stays type of night, type of room, parking option), characteristics of customer (number of adults, children, country), records for the customer (repeated guest, previous cancellations), and others. We can start analyzing booking cancellations by exploring its relationship with covariates. For this relationships, we can highlight the following insights and plots:

- Most of the data is for City Hotel and for year 2016. Cancellations are relevant for any type of hotel or year (2015-2017)

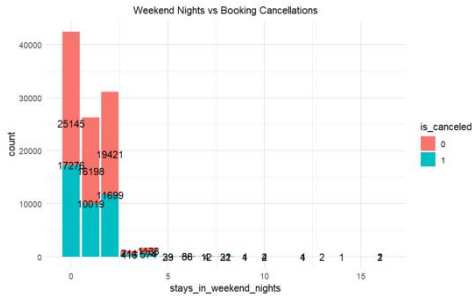


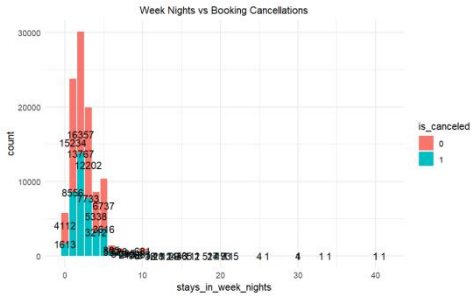
- Cycle in terms of months makes sense for the industry. In terms of number of week in the year, lowest points at the beginning and end of year. Cancellations remains relevant for any month or number of weeks (same with days in month).



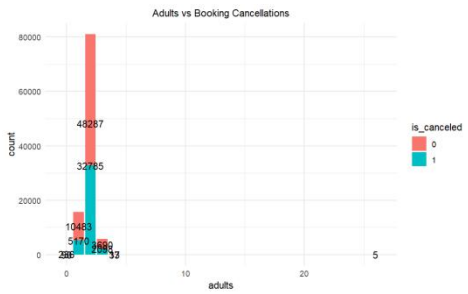


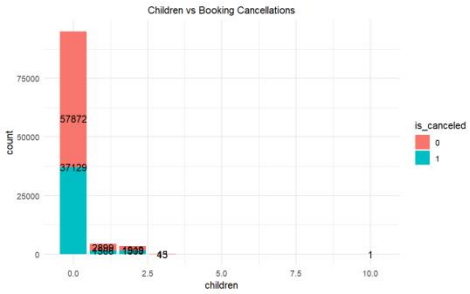
- Bookings and cancellations seem to have same pattern for type of nights: most activity happens for less than 2 weekends nights and less than 5 weeknights.



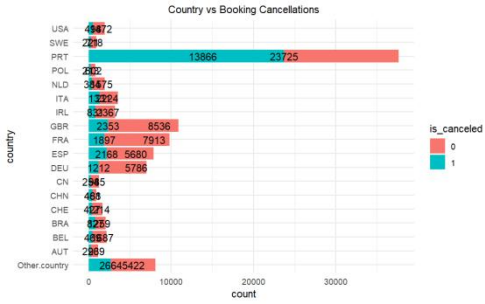


- Bookings and cancellations seem to have same pattern for customer profiles: most of them are two adults and do not have children (same with babies).

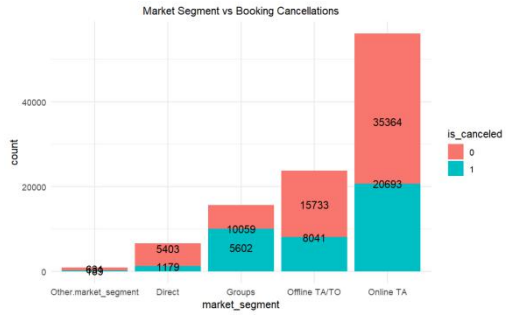


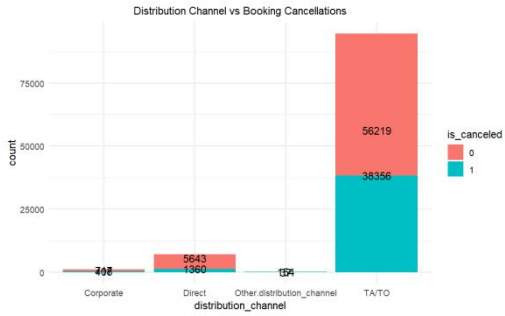


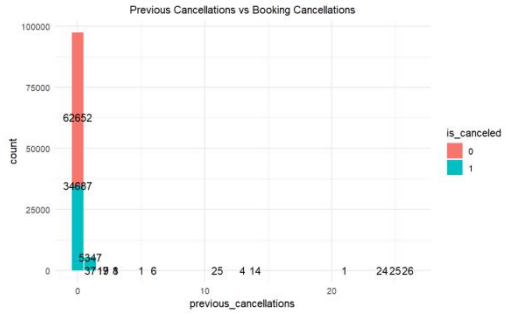
- Bookings and cancellations seem to have same pattern in terms of country, meal, and special request: most customers are from Portugal, get BB meal, and do not have special request.



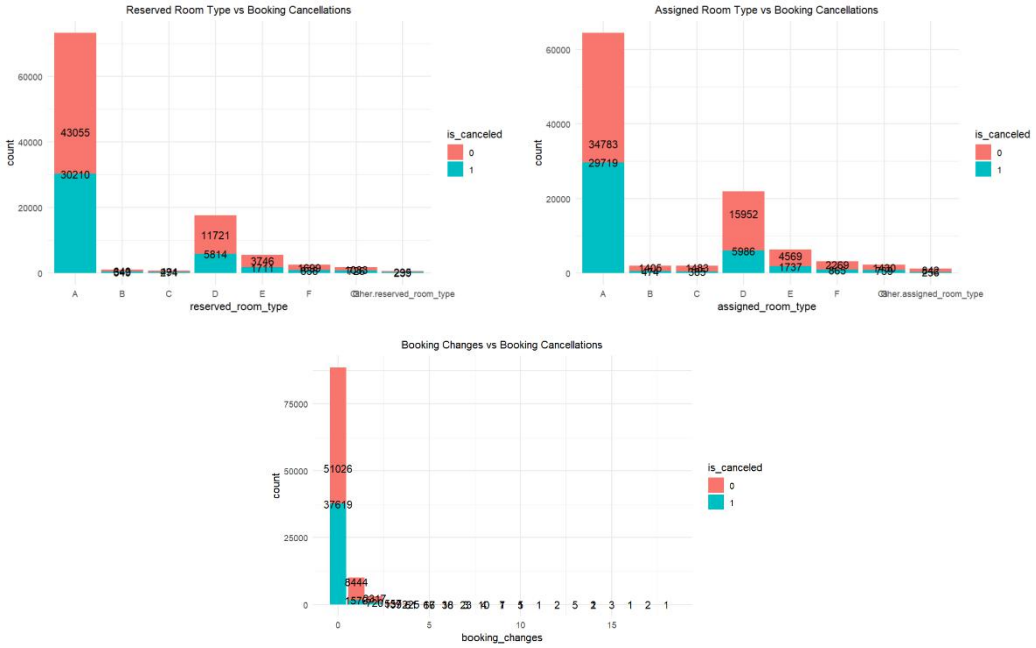
- Most of bookings and cancellations are for market segment Online TA, distribution channel TA/TO, and are new customers.



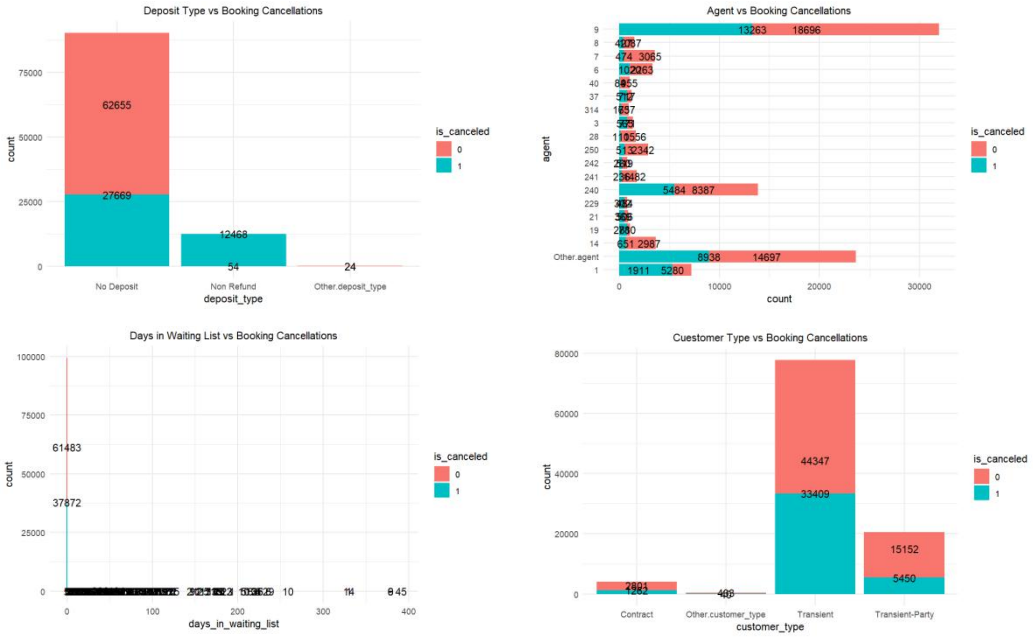




- Most of bookings and cancellations are for reserved rooms A, which is also the more frequent type of room finally assigned, and for which are not changes.



- Most of bookings and cancellations are for customer that do not make deposits, which comes from similar agents (most from 9 followed by Other), are transient (followed by transient party), and do not wait for getting the booking done.



Data Pre-processing

The dataset can be considered as balanced. There are only 4 NA rows, we remove it. As most of the `company` column is NULL, we delete the available from the data set. Also, we delete rows with NULL value as there are not many of them compared with the whole data set. Moreover, we also delete the `reservation_status` column because it has the same meaning as our response `is_canceled`.

We delt with date column, changing them into data format. Then we converted the categorical variables into factor for model building, and convert every column that has less than 700 unique values into a factor.

After our preprocessing, all categorical variables are within 20 levels. At last, we used first 80% as train set and the last 20% as test set.

III. Modeling

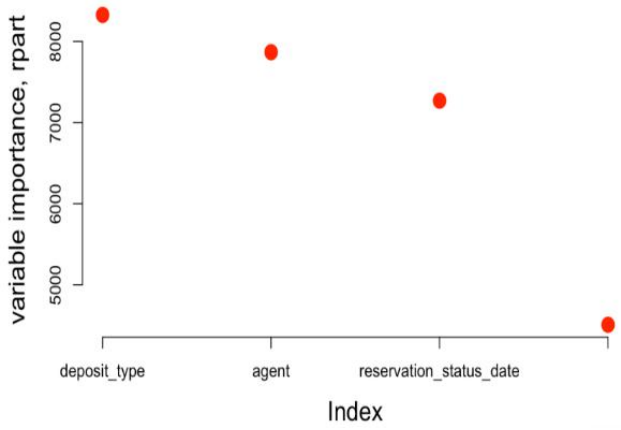
Summary on Approach

Since our goal is to predict consumers will cancel their reservation or not, so it is a binary classification problems. So we will use Decision tree, Random Forest, Boosting, and Neural Networks as our models.

Decision Tree

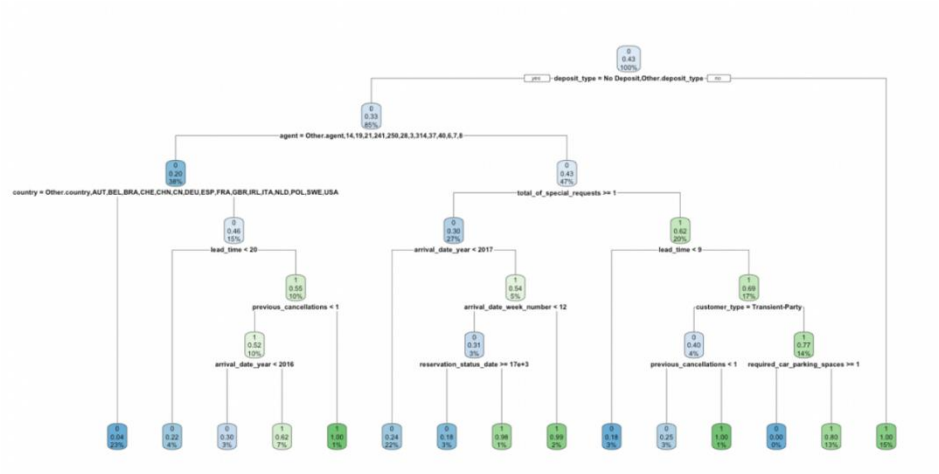
First we tried a decision tree model. Here is the variable importance plot, showing the three most important features are deposit type, agent, and reservation status date.

Predict on the test set, we get the accuracy of 0.9463, and the confusion matrix seems to be quite reasonable and does not have obvious issue like unbalanced prediction.




```
Confusion Matrix and Statistics

     tree.pred
Ytest     0     1
    0 12221   385
    1   721  7252

               Accuracy : 0.9463
                 95% CI : (0.9431, 0.9493)
    No Information Rate : 0.6289
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8859

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

            Sensitivity : 0.9443
            Specificity : 0.9496
         Pos Pred Value : 0.9695
         Neg Pred Value : 0.9096
             Prevalence : 0.6289
         Detection Rate : 0.5939
   Detection Prevalence : 0.6126
      Balanced Accuracy : 0.9469
```

From the tree plot (this is not our final tree model, the final model is too big to show, and here is a small tree.), we can see that deposit type, agent, and reservation status date are truly playing important roles in our tree model.



Random Forest

Next, we used random forest model. After using grid search to tune parameters, we found the best parameter and the corresponding RMSE on training set is 0.26. Testing this model on test set, we found the accuracy is 96%.

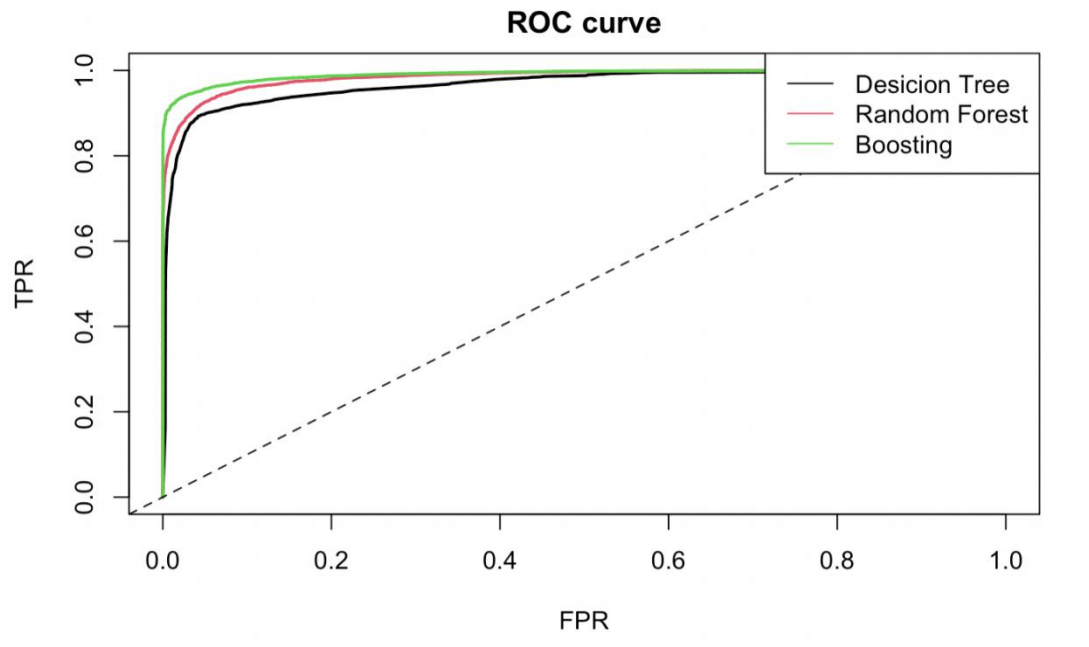
```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 12464    94
         1   737  7284

               Accuracy : 0.9596
                 95% CI : (0.9568, 0.9623)
    No Information Rate : 0.6415
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9139

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

            Sensitivity : 0.9442
            Specificity : 0.9873
         Pos Pred Value : 0.9925
         Neg Pred Value : 0.9081
             Prevalence : 0.6415
         Detection Rate : 0.6057
   Detection Prevalence : 0.6102
      Balanced Accuracy : 0.9657
```

Boosting

Similarly, we tried Boosting model. We ran a grid parameter search on the number of splits (interaction depth), the number of trees and shrinkage. With the best parameters, we fitted a final boosting model and tested it on test set. The following is its result, it has accuracy of 96%.

```
Confusion Matrix and Statistics

     rf.pred
Ytest     0     1
    0 12446   160
    1   612  7361

               Accuracy : 0.9625
                 95% CI : (0.9598, 0.965)
    No Information Rate : 0.6345
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9201

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

            Sensitivity : 0.9531
            Specificity : 0.9787
         Pos Pred Value : 0.9873
         Neg Pred Value : 0.9232
             Prevalence : 0.6345
         Detection Rate : 0.6048
   Detection Prevalence : 0.6126
      Balanced Accuracy : 0.9659
```

To compare three tree models, we draw the ROC curve:



Also, we can compute AUC for three models, and boosting model has the highest AUC of 0.98, while AUC of Random Forest is 0.972 and 0.967.

Neural Networks

Then we tried Neural Networks model with early stopping and grid search. After using grid search on training data, we found the best parameters and the NN model predict completely correct on validation set. However, when we performed it on test dataset, it performed not as well as the models we used before. AUC value is 0.749 and Accuracy equals to 0.712.

Below is the result:

	0 <chr>	1 <chr>	Error <chr>	Rate <chr>
0	8612	3918	0.312690	=3918/12530
1	2001	6048	0.248602	=2001/8049
Totals	10613	9966	0.287623	=5919/20579

```
MSE:  0.5475892
RMSE:  0.7399927
LogLoss:  4.808273
Mean Per-Class Error:  0.2806459
AUC:  0.7491157
AUCPR:  0.5854553
Gini:  0.4982314
```

Let's also see the 10 most important variables:

	variable <chr>	relative_importance <dbl>
1	x.arrival_date_year	1.0000000
2	x.reservation_status_date	0.7823861
3	x.arrival_date_week_number	0.4101236
4	x.deposit_type.Non Refund	0.4063434
5	x.deposit_type.No Deposit	0.3874705
6	x.country.PRT	0.2595074
7	x.previous_cancellations	0.1927064
8	x.required_car_parking_spaces	0.1772559
9	x.agent.9	0.1647516
10	x.is_repeated_guest	0.1611553

IV. Conclusions

We are showing hotel industry business the benefits of applying machine learning tools to get useful information to make better decisions regarding business problems, in this case booking cancellations.

Particularly, we find that **our final model is boosting model**, with **accuracy of 96%** on test set

Recommendations:

Hotel industry companies can choose an optimal level of threshold to maximize the profits according to its own revenues and costs using this model. In other words, **the best model will be the one that allow each company to increase its profits**. The discussion based on level of accuracy of models is an initial step for this analysis.

This strategy will allow you to better approach the **administration of bookings**, for example, by creating set of policies, according to the probability of booking cancellations. By doing so, you will minimize losses or **maximize profits** in your business.

Appendix

---

title: "Prediction of hotel bookings cancellation using Machine Learning"

author: "Xingyue Fang, Xinyi Gu, and Guillermo Trefogli"

date: "18/03/2022"

output:

  pdf_document:

    number_sections: yes

  html_document:

    df_print: paged

urlcolor: blue

```{r}

df_hotel = na.omit(df_hotel)

df_hotel = df_hotel[,-which(names(df_hotel)=='company')]

df_hotel = df_hotel[-which(df_hotel$country=='NULL'),]

df_hotel = df_hotel[-which(df_hotel$agent=='NULL'),]

df_hotel = df_hotel[,-30]

dim(df_hotel)

```

Deal with date column.

```{r}

df_hotel <- mutate_at(df_hotel, 'reservation_status_date', function(x) as.Date(x, format = "%Y-%m-%d"))

```

Then we convert the categorical variables into factor for model building.

```{r}

df_hotel = mutate_if(df_hotel, is.character, as.factor)

df_hotel$is_canceled = as.factor(df_hotel$is_canceled)

str(df_hotel)

```

Convert every column that has less than 700 unique values into a factor.

```{r}

combinerarecategories<-function(data_frame,mincount){

for (i in 1:ncol(data_frame)) {

a<-data_frame[,i]

replace <- names(which(table(a) < mincount))

levels(a)[levels(a) %in% replace] <-

paste("Other", colnames(data_frame)[i], sep=".")

data_frame[,i]<-a

}

return(data_frame)

}
```

```

```{r}
df_hotel<-combinerarecategories(df_hotel,700)
str(df_hotel)
dim(df_hotel)
```

Now, all categorical variables are within 20 levels.

As the hotel booking information contains booking date, we sort the dataset according to `reservation_status_date` and use first 80% as train set and the last 20% as test set.

```{r}
ntrain = round(0.8*nrow(df_hotel))
tr = sample(1:nrow(df_hotel), ntrain)
df.train = df_hotel[tr,]
df.test = df_hotel[-tr,]
```

```{r}
Xtrain = df.train[,-2]
Ytrain = df.train[,2]
Xtest = df.test[,-2]
Ytest = df.test[,2]
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#hotel type
hotel_explo <- df_hotel %>%
  group_by(hotel, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

hotel_explo

df_hotel %>%
  ggplot(aes(hotel, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Hotel Type vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#year
year_explo <- df_hotel %>%
  group_by(arrival_date_year, is_canceled) %>%

```
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

year_explo

df_hotel %>%
  ggplot(aes(arrival_date_year, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Year vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
# month
month_explo <- df_hotel %>%
  group_by(arrival_date_month, is_canceled) %>%
  summarise(n = n())

month_explo

df_hotel %>%
  ggplot(aes(arrival_date_month, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Month vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))

```
```{r, warning=FALSE, echo=FALSE, message=FALSE}
# Week
week_explo <- df_hotel %>%
  group_by(arrival_date_week_number, is_canceled) %>%
  summarise(n = n())

week_explo

df_hotel %>%
  ggplot(aes(arrival_date_week_number, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
```

```
  labs(subtitle = "Week Number (in the year) vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5))

```
```{r, warning=FALSE, echo=FALSE, message=FALSE}
# Day of month
day_explo <- df_hotel %>%
  group_by(arrival_date_day_of_month, is_canceled) %>%
  summarise(n = n())

day_explo

df_hotel %>%
  ggplot(aes(arrival_date_day_of_month, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Day of Month vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5))

```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
# Weekend night
wknd_night_explo <- df_hotel %>%
  group_by(stays_in_weekend_nights, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

wknd_night_explo

df_hotel %>%
  ggplot(aes(stays_in_weekend_nights, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Weekend Nights vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))

```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
# Week night
wk_night_explo <- df_hotel %>%
```

```
  group_by(stays_in_week_nights, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

wk_night_explo

df_hotel %>%
  ggplot(aes(stays_in_week_nights, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Week Nights vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))

```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#adults
adults_explo <- df_hotel %>%
  group_by(adults, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

adults_explo

df_hotel %>%
  ggplot(aes(adults, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Adults vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#children
children_explo <- df_hotel %>%
  group_by(arrival_date_year, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))
```

```
children_explo

df_hotel %>%
  ggplot(aes(children, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Children vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#babies
babies_explo <- df_hotel %>%
  group_by(arrival_date_year, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

babies_explo

df_hotel %>%
  ggplot(aes(babies, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Babies vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#meal
meal_explo <- df_hotel %>%
  group_by(meal, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

meal_explo

df_hotel %>%
  ggplot(aes(meal, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
```

```
  labs(subtitle = "Meal vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#country
country_explo <- df_hotel %>%
  group_by(country, is_canceled) %>%
  summarise(n = n())

country_explo

df_hotel %>%
  ggplot(aes(country, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Country vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..)) +
  coord_flip()
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#market segment
sgmt_explo <- df_hotel %>%
  group_by(market_segment, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

sgmt_explo

df_hotel %>%
  ggplot(aes(market_segment, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Market Segment vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#distribution channel
```

```
chn_explo <- df_hotel %>%
  group_by(distribution_channel, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

chn_explo

df_hotel %>%
  ggplot(aes(distribution_channel, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Distribution Channel vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```
```{r, warning=FALSE, echo=FALSE, message=FALSE}
#repeated guest
rp_explo <- df_hotel %>%
  group_by(is_repeated_guest, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

rp_explo

df_hotel %>%
  ggplot(aes(is_repeated_guest, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Repeated Guest vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))

df_hotel %>%
  filter(is_canceled == 1) %>%
  ggplot(aes(is_repeated_guest)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Repeated Guest Among Those Who Cancelled Bookings") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))

```
```
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#previous cancellations
prv_explo <- df_hotel %>%
  group_by(previous_cancellations, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

prv_explo

df_hotel %>%
  ggplot(aes(previous_cancellations, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Previous Cancellations vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#room reserved
roomr_explo <- df_hotel %>%
  group_by(reserved_room_type, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

roomr_explo

df_hotel %>%
  ggplot(aes(reserved_room_type, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Reserved Room Type vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
# room assigned
rooma_explo <- df_hotel %>%
```

```
  group_by(assigned_room_type, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

rooma_explo

df_hotel %>%
  ggplot(aes(assigned_room_type, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Assigned Room Type vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#changes in booking
changes_explo <- df_hotel %>%
  group_by(booking_changes, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

changes_explo

df_hotel %>%
  ggplot(aes(booking_changes, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Booking Changes vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#deposit type
depost_explo <- df_hotel %>%
  group_by(deposit_type, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))
```

```
depost_explo

df_hotel %>%
  ggplot(aes(deposit_type, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Deposit Type vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#agent
agent_explo <- df_hotel %>%
  group_by(agent, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

agent_explo

df_hotel %>%
  ggplot(aes(agent, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Agent vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..)) +
  coord_flip()
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#days in waiting list
waitlist_explo <- df_hotel %>%
  group_by(days_in_waiting_list, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

waitlist_explo

df_hotel %>%
  ggplot(aes(days_in_waiting_list, fill = is_canceled)) +
```

```
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Days in Waiting List vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#customer type
custom_explo <- df_hotel %>%
  group_by(customer_type, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

custom_explo

df_hotel %>%
  ggplot(aes(customer_type, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Cuestomer Type vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
#parking
parking_explo <- df_hotel %>%
  group_by(required_car_parking_spaces, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

parking_explo

df_hotel %>%
  ggplot(aes(required_car_parking_spaces, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Parking vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```
```{r, warning=FALSE, echo=FALSE, message=FALSE}
#special requests
sprequest_explo <- df_hotel %>%
  group_by(total_of_special_requests, is_canceled) %>%
  summarise(n = n()) %>%
  mutate(percent = n/sum(n)*100,
         percent = paste(round(percent, 0), "%"))

sprequest_explo

df_hotel %>%
  ggplot(aes(total_of_special_requests, fill = is_canceled)) +
  geom_bar() +
  theme_minimal() +
  labs(subtitle = "Number of Special Requests vs Booking Cancellations") +
  theme(plot.subtitle = element_text(hjust = 0.5)) +
  geom_text(stat = "count", aes(label = ..count..))
```

```{r, warning=FALSE, echo=FALSE, message=FALSE}
# relevance of booking in time
first_explo <- df_hotel %>%
  group_by(reservation_status_date, is_canceled) %>%
  summarise(n = n())

table_cancel <- df_hotel %>%
  group_by(arrival_date_year,is_canceled) %>%
  summarise(cases = n())

table_cancel %>%
  group_by(arrival_date_year) %>%
  mutate(percent = cases/sum(cases)*100,
         percent = paste(round(percent, 0), "%"))

first_explo %>%
  ggplot(aes(reservation_status_date, n, color = is_canceled)) +
  geom_line() +
  theme_bw() +
  labs(subtitle = "Evolution of Booking Cancelattions in Time") +
  theme(plot.subtitle = element_text(hjust = 0.5))
```

Model building
```

Neural Network

```{r}
library(h2o)
h2o.init(nthreads=-1, max_mem_size="8G")
```

```{r}
dftrain = as.h2o(data.frame(x=Xtrain,y=Ytrain), destination_frame = "xor.train")
splits <- h2o.splitFrame(dftrain, c(0.8), seed=123)
train  <- h2o.assign(splits[[1]], "train.hex") # 80% as train set
valid  <- h2o.assign(splits[[2]], "valid.hex") # 20% as validation set
test = as.h2o(data.frame(x=Xtest,y=Ytest), destination_frame = "xor.train")

response <- "y"
predictors <- setdiff(names(dftrain), response)
```

First try a model:

```{r,cache=TRUE}
m1 <- h2o.deeplearning(
  model_id="dl_model_first",
  training_frame=train,
  validation_frame=valid,      ## validation data are used for scoring and early stopping
  x=predictors,
  y=response,
  activation="Tanh",      ## default
  hidden=c(100,100),
  epochs=100,
  seed = 123
)
summary(m1)
plot(m1)
```

Early stopping

```{r,cache=TRUE}
m2 <- h2o.deeplearning(
  model_id="dl_model_faster",
  training_frame=train,
  validation_frame=valid,
  x=predictors,
  y=response,
  hidden=c(16,16,16,16),                  ## small network, runs faster
  epochs=1000000,                         ## hopefully converges earlier...
  stopping_rounds=2,
  stopping_metric="logloss",                ## could be "MSE","logloss","r2"
```

```
  stopping_tolerance=0.01,
  seed = 123
)
summary(m2)
plot(m2)
```

Tuning

```{r,cache=TRUE}
m3 <- h2o.deeplearning(
  model_id="dl_model_tuned",
  training_frame=train,
  validation_frame=valid,
  x=predictors,
  y=response,
  hidden=c(32,32,32,32),          ## more hidden layers -> more complex interactions
  epochs=1000,                    ## to keep it short enough
  score_duty_cycle=0.025,         ## don't score more than 2.5% of the wall time
  l1=1e-5,                        ## add some L1/L2 regularization
  l2=1e-5,
  max_w2=10,                      ## helps stability for Rectifier
  stopping_rounds=2,
  stopping_metric="logloss",          ## could be "MSE","logloss","r2"
  stopping_tolerance=0.01,
  seed = 123
)
summary(m3)
plot(m3)
```

Among m1, m2, m3We can see that m2 has the smallest logloss on validation set. Let's see their performance on test set.

```{r}
h2o.performance(m1, newdata=test)
h2o.performance(m2, newdata=test)
h2o.performance(m3, newdata=test)
```

Tuning with Grid Search

```{r, cache=TRUE}
hyper_params <- list(
  hidden=list(c(32,32,32),c(64,64)),
  input_dropout_ratio=c(0,0.05),
  l1=c(0,1e-5,1e-3),
  l2=c(0,1e-5,1e-3),
  max_w2=c(5,10)
```

```
)

grid <- h2o.grid(
   algorithm="deeplearning",
   grid_id="dl_grid",
   training_frame=train,
   validation_frame=valid,
   x=predictors,
   y=response,
   epochs=10,
   stopping_metric="logloss",
   stopping_tolerance=1e-2,              ## stop when MSE does not improve by >=1% for 2 scoring events
   stopping_rounds=2,
   score_duty_cycle=0.025,                ## don't score more than 2.5% of the wall time
   hyper_params=hyper_params,
   seed = 123
)
```
```{r}
grid <- h2o.getGrid("dl_grid",sort_by="logloss",decreasing=FALSE)
grid
```

```{r}
grid@summary_table[1,]
```

```{r}
best_model <- h2o.getModel(grid@model_ids[[1]])
best_model
```

```{r}
h2o.performance(best_model, newdata=test)
```

Tuning with Random Search
```{r}
hyper_params <- list(
   activation=c("Rectifier","Tanh","RectifierWithDropout","TanhWithDropout"),
   hidden=list(c(20,20),c(50,50),c(30,30,30),c(25,25,25,25),c(64,64,64,64)),
   input_dropout_ratio=c(0,0.05),
   l1=seq(0,1e-4,1e-6),
   l2=seq(0,1e-4,1e-6),
```

```
  max_w2=c(5,10,15)
)

## Stop once the top 5 models are within 1% of each other
##
##         - the windowed average varies less than 1%
##
search_criteria = list(
  strategy = "RandomDiscrete",
  max_runtime_secs = 360,
  max_models = 100,
  seed=1,
  stopping_rounds=5,
  stopping_tolerance=1e-2
  )

dl_random_grid <- h2o.grid(
  algorithm="deeplearning",
  grid_id = "dl_grid_random",
  training_frame=train,
  validation_frame=valid,
  x=predictors,
  y=response,
  epochs=10,
  stopping_metric="logloss",
  stopping_tolerance=1e-2,              ## stop when MSE does not improve by >=1% for 2 scoring events
  stopping_rounds=2,
  score_duty_cycle=0.025,                ## don't score more than 2.5% of the wall time
  hyper_params = hyper_params,
  search_criteria = search_criteria,
  seed = 123
)
```
```{r}
grid <- h2o.getGrid("dl_grid_random", sort_by="logloss", decreasing=FALSE)
grid
```

```{r}
grid@summary_table[1,]
```

```{r}
best_model <- h2o.getModel(grid@model_ids[[1]]) ## model with lowest RMSE
```

best_model
```

```{r}
h2o.performance(best_model, newdata=test)
```

\newpage

Three Tree models
```{r}
train = cbind(Xtrain,Ytrain)
colnames(train)[colnames(train) == 'Ytrain'] <- 'y'
test = cbind(Xtest,Ytest)
colnames(test)[colnames(test) == 'Ytest'] <- 'y'
```

Decision Tree
```{r}
big.tree = rpart(y~., data = train,
                 control=rpart.control(minsplit=5,
                                       cp=0.0001,
                                       xval=5)
)

nbig = length(unique(big.tree$where))
cat('size of big tree: ',nbig,'\n')

cptable = printcp(big.tree)
bestcp = cptable[ which.min(cptable[,"xerror"]), "CP" ]      # this is the optimal cp parameter
best.tree = prune(big.tree,cp=bestcp)
```
```{r}
#check feature importance
tvimp = best.tree$variable.importance

plot(tvimp[1:4],axes=F,pch=16,col='red',ylab="variable importance, rpart",cex=2,cex.lab=1.5)
axis(1,labels=names(tvimp[1:4]),at=1:length(tvimp[1:4]))
axis(2)
```
```{r}
# too big to see
temp.tree = prune(big.tree,cp=0.01)
```
```

```
rpart.plot(temp.tree)
```

```{r}
# Predict on test set
tree.pred <- predict(best.tree, newdata = Xtest, type = "class")
xtab.tree <- table(Ytest, tree.pred)
confusionMatrix(xtab.tree)
```

Random Forest
```{r}
train$y = as.factor(train$y)
```

```{r}
hyper_grid <- expand.grid(
  mtry            = seq(3, 5, by = 2),
  node_size    = c(25, 50),
  OOB_RMSE     = 0
)
for(i in 1:nrow(hyper_grid)) {
  # train model
  model <- ranger(
    formula               = y~.,
    data                  = train,
    num.trees             = 500,
    mtry                  = hyper_grid$mtry[i],
    min.node.size     = hyper_grid$node_size[i],
    seed                  = 123
  )
    # add OOB error to grid
  hyper_grid$OOB_RMSE[i] <- sqrt(model$prediction.error)
}

(oo = hyper_grid %>%
  dplyr::arrange(OOB_RMSE) %>%
  head(10))
```

```{r}
#Final model for Random forest
rf.fit.final <- ranger(
    formula               = y~.,
```

```
      data                  = train,
      num.trees             = 500,
      mtry                  = oo[1,]$mtry,
      min.node.size     = oo[1,]$node_size,
      probability = TRUE
      )
#Predict on test set
rf.prob = predict(rf.fit.final, data = Xtest)$predictions
rf.pred = round(rf.prob[,2])
xtab.rf <- table(Ytest, rf.pred)
confusionMatrix(xtab.rf)
```

Boosting Model

```{r, eval=FALSE}
# create hyperparameter grid
hyper_grid <- expand.grid(
  shrinkage = c(0.1),         ## controls the learning rate
  interaction.depth = c(5, 8), ## tree depth
  n.minobsinnode = c(30,50), ##    minimum number of observations required in each terminal node
  bag.fraction = c(.5),    ##    percent of training data to sample for each tree
  optimal_trees = 0,                      ## a place to dump results
  min_RMSE = 0                            ## a place to dump results
)
```

Then perform the grid search, also use cross-validation.

```{r}
train.xgb = Matrix::sparse.model.matrix(y~ ., data = train)[,-1]
test.xgb = Matrix::sparse.model.matrix(y~ ., data = test)[,-1]
```

```{r,eval=FALSE}
for(i in 1:nrow(hyper_grid)) {
  # create parameter list
  params <- list(
    eta = hyper_grid$shrinkage[i],
    max_depth = hyper_grid$interaction.depth[i],
    min_child_weight = hyper_grid$n.minobsinnode[i],
    subsample = hyper_grid$bag.fraction[i]
  )

  # reproducibility
  set.seed(123)
```

```
  # train model using Cross Validation
  xgb.tune <- xgb.cv(
    params = params,
    data = train.xgb,
    label = Ytrain,
    nrounds = 2000,
    nfold = 5,
    objective = "reg:squarederror",         # for regression models
    verbose = 0,                            # silent,
    early_stopping_rounds = 10              # stop if no improvement for 10 consecutive trees
  )

  # add min training error and trees to grid
  hyper_grid$optimal_trees[i]<-which.min(xgb.tune$evaluation_log$test_rmse_mean)
  hyper_grid$min_RMSE[i] <- min(xgb.tune$evaluation_log$test_rmse_mean)
}
```

Arrange grid by RMSE

```{r,eval=FALSE}
(oo = hyper_grid %>%dplyr::arrange(min_RMSE) %>%head(5))
```

Extract best parameters

```{r,echo=FALSE}
# parameter list
params <- list(
  eta = 0.1,
  max_depth = 5,
  min_child_weight = 30,
  subsample = 0.5
)
```

Train Final Model for Boosting

```{r}
xgb.fit.final <- xgboost(
  params = params,
  data = train.xgb,
  label = as.factor(train$y),
  nrounds = 300,
  objective = "reg:squarederror",
  verbose = 0
)
```

```{r}
```

test.xgb = Matrix::sparse.model.matrix(y~ ., data = test)[,-1]
```

```{r}
test$y = as.factor(test$y)
#Predict on test set
xgb.pred = predict(xgb.fit.final, newdata = test.xgb)
xgb.pred = as.factor(round(xgb.pred-1))

# confusion matrix of test set
confusionMatrix(Ytest, xgb.pred)
```

ROC curves for 3 tree models
```{r}
tree.prob = predict(best.tree, newdata = Xtest, type = "prob")[,2]
rf.prob = predict(rf.fit.final, data = Xtest)$predictions[,2]
xgb.prob <- predict(xgb.fit.final, newdata=test.xgb)

yhat = tibble("Desicion Tree" = tree.prob,"Random Forest" = rf.prob,"Boosting"= xgb.prob)
for(i in 1:ncol(yhat)) {
   pred = prediction(yhat[,i], Ytest)
   perf = performance(pred, measure = "tpr", x.measure = "fpr")
   if (i == 1) {
   plot(perf, col = 1, lwd = 2,
   main= 'ROC curve', xlab='FPR', ylab='TPR', cex.lab=1)
      }
      else
      {
   plot(perf, add = T, col = i, lwd = 2)
      }
}
abline(0,1,lty=2)
legend("topright",legend=colnames(yhat),col=1:ncol(yhat),lty=rep(1,ncol(yhat)))
```
Compute AUC
```{r}
for(i in 1:ncol(yhat)) {
   pred = prediction(yhat[,i], Ytest)
   perf <- performance(pred, measure = "auc")
   print(paste0("AUC of ", colnames(yhat)[i], ": ", perf@y.values[[1]]))
}
```
```