Stats 506 Final Project

Team Member：Jiaqi Zhu, Yumiao Hui, Mingzhu He

Introduction

The dataset we use is hotel booking data. It has 119390 hotel booking records with 32 variables, including hotel type, booking time, number of guests, number of nights, and everything related to the hotel booking. One important variable we want to mention is 'is_cancelled' which suggests whether the booking was eventually cancelled or not. We think it is closely related to hotel management and operation, thus we will concentrate on it to get some insights. Other variables we are interested in are price (adr), nights stay (stays_in_weekend_nights, stays_in_week_nights), number of guests (adults, children, babies), customer type, gap dates between booking and arrival (lead_time), etc.

Data cleaning

Considering the large number of observations, we have to clean up the dataset first. There are three variables with NA in them, and the number of NA and ratio of NA are shown in the table.

	na_num	na_rate
	<int>	<dbl>
company	112593	94.31
agent	16340	13.69
country	488	0.41
children	4	0.00

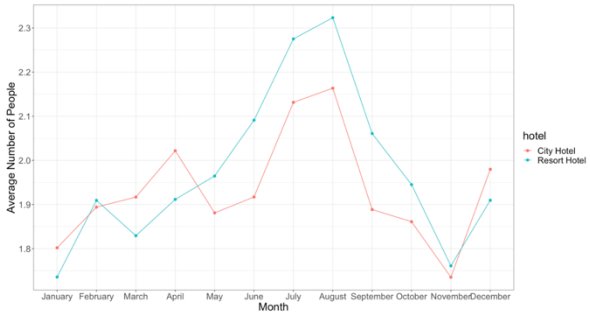
We notice that the NA rate for 'company' is larger than 90%, so we remove the column. For NAs in 'agent', 'country', and 'children', we fill them with 0, 'Unknown', and median, respectively. Also, we notice that for some observations, the total number of guests is 0, which doesn't make sense, so I remove those observations. This also makes the dataset suspicious in its reliability.

Question 1: What seasonality trends do resort hotels and city hotels have? Are they the same?

For the hotel industry, it is important for management to understand its seasonality trends. In the first question, we are interested in seasonality trends in the hotel industry, and try to compare those for city hotels and resort hotels. We consider seasonality trends in three dimensions, which are price, number of guests, and number of nights staying.

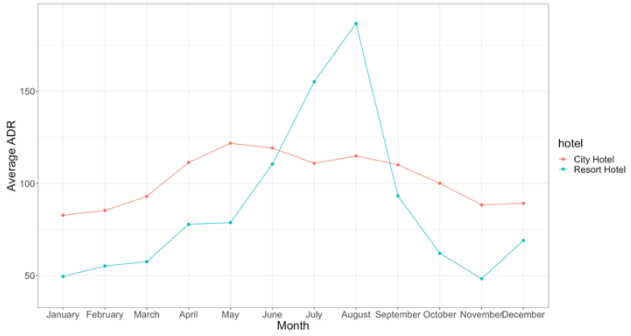
(1) Price

The variable related to price is average daily rate (ADR), which is the sum of all lodging transactions divided by the total number of staying nights. This line chart shows the trend of ADR for resort hotels and city hotels. We can see that, for resort hotels, the high season is from June to August, while November to March is the low season. ADR for the high season is more than three times of that for the low season. For city hotels, however, there is no obvious high season and low season, and prices are quite stable during the year.



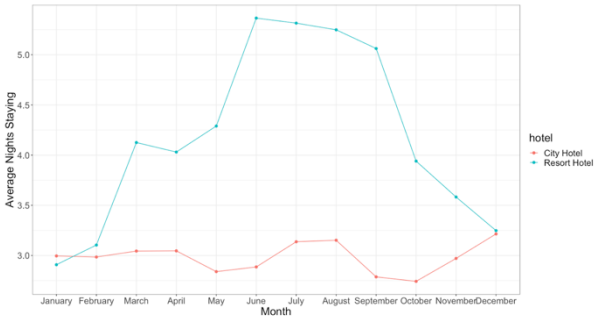
(2) Number of nights staying

For total number of nights staying, I add up weekdays staying and weekends staying together. This line chart shows the trend of total nights of staying for resort hotels and city hotels. We can see that, for resort hotels, customers live more than 5 nights from June to September, while from December to February, customers only live for less than three nights. However, for city hotels, it is stable for the whole year with around three nights of staying.



(3) Number of guests

For the total number of people living in the hotel, I add up adults, children and babies. From this line chart, we can see that resort hotels and city hotels have really similar trends. In July and August, the average number of people is larger than 2, which makes sense, because those months are common for family trips, no matter for city hotels or resort hotels. From the results, we can conclude that resort hotels have distinct seasonal trends, with high season during June to August, and low season from November to March. However, for city hotel, there is no obvious difference during the year.



Question 2：Questions based on cancellation rate on hotel booking data:

Introduction:

Here we focus on the cancellation rate of hotel booking, and thus propose several questions to compare the difference of cancellation rate under different groups. The data we used here is original hotel booking data which is after data cleaning and several variables format changing. Make sure variable **is_canceled** is a double variable which value is 0 or 1 and it is not a factor variable, so we can further based on this to calculate the cancellation rate in different groups. We define cancellation rate here and the formula to calculate the cancellation rate is:

$$Calcellation\ Rate = \frac{Number\ of\ canceled(1)}{Number\ of\ canceled(1) + Number\ of\ not\ canceled(0)}$$

For the several questions just focused on the cancellation rate of hotel booking. We want to figure out whether there is difference between different group conditions. In fact, we want to expect some difference here in the results so we can further to take these conditions as potential predictors to make prediction on the cancellation rate. Here we need to make sure that these condition variables are factors not double variables.

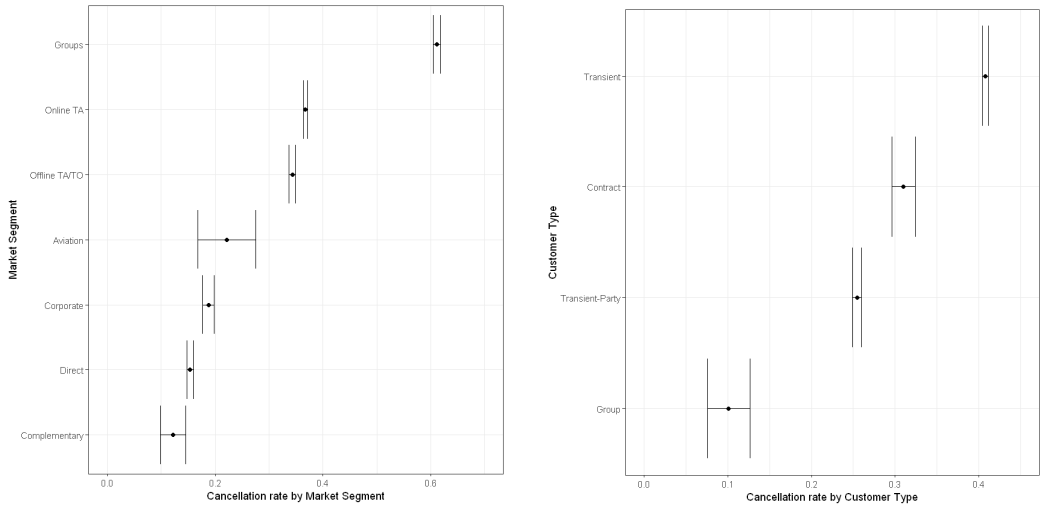
For the potential challenge and problems in the data set, after we conduct some basic summary on these variables, we found that for **deposit_type**, we expect customers who have non-refundable deposit should cancel less than refundable deposit because it means they cannot get their deposit back when cancel the hotel booking. However, after we calculate the cancellation rate based on different deposit type, we found that the result is actually contradictory to our expectation, Non Refund type has a cancellation rate which is high to 0.994, while No Deposit type is 0.284 and Refundable type is 0.222. So we may need to further check or investigate the collected data for deposit type, maybe there are some errors during sample surveys and data collection or entry period.

Q2-a: What is the cancellation rate by customer type? By Market Segment?

For Market Segment type, it means the statistics for what kinds of channels that the customer may go to view and book a hotel room, such as through the direct hotel website or online travel agencies such as Expedia, Booking.com, etc or offline travel agency on the streets. It is more detailed than distribution channel so we choose to analysis this

variable, Here in this dataset, term “TA” means “Travel Agents” and “TO” means “Tour Operators”. Here we just filter the “Undefined” level out from the data because it is meaningless here for further analysis. So actually there are 7 different kinds of channels, as the below left panel of plots shows.

For customer type, we can directly see from the categorical data which has 4 groups: Contract, Group, Transient, Transient-Party. Group means when the booking is associated to a group; Transient means when the booking is not part of a group or contract, and is not associated to other transient booking; Transient-party means when the booking is transient, but is associated to at least other transient booking.



For steps to calculate the cancellation rate for each groups(see codes Part I & II)
For steps to calculate the confidence intervals for the cancellation rate, according to Central limit theorem, first we can calculate sample mean as point estimate, then use Bootstrap to calculate 95% CI applying normal confidence limits.

The formula here are as below(codes are in Basic Part).

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

For $V_{\widehat{F_n}}(\hat{\theta})$ we can calculate it by Monte Carlo method (1000 times), X_{i_*} is from $\widehat{F_n}$

$$\hat{\theta}_{b^*} = \frac{1}{n}\sum_{i=1}^{n} X_{i_*}$$

$$V_{\widehat{F_n}}(\hat{\theta}) \approx V_{boot}(\hat{\theta}) = \frac{1}{1000}\sum_{b=1}^{1000}(\hat{\theta}_{b^*} - \frac{1}{1000}\sum_{b=1}^{1000}\hat{\theta}_{b^*})$$

So, 95% confidence interval with normal limits is:

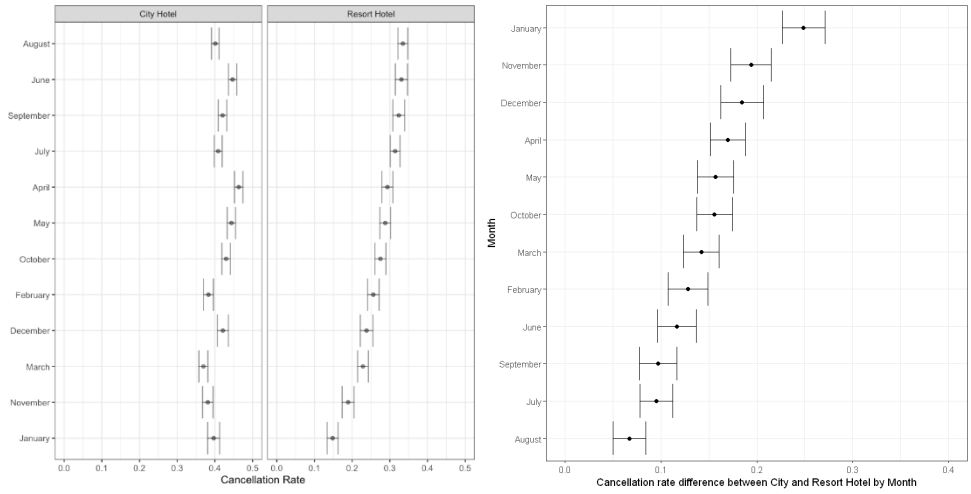
$$(\hat{\theta} - 1.96 * V_{\widehat{F_n}}(\hat{\theta}), \hat{\theta} + 1.96 * V_{\widehat{F_n}}(\hat{\theta}))$$

After calculation we draw plots above (see code part I & II). The result shows that for

different market segments to book hotel rooms, it is most unlikely to cancel the room for the complementary type bookings, it is reasonable because usually it is free of charge, we can see in the data that most of the adr is 0 here and it is like a bonus for customers. Besides, for customer type, a customer when books individually which is called transient here seems more likely to cancel the booking, which is also reasonable because, usually for one person to make plan alone without needs to care about others' minds or need to link with other hotel booking orders for example like make a trip to visit several places in fixed days, which means more flexibility to change or cancel the schedule freely as much as they want.

Q2-b: Is there difference between cancellation rate by hotel type in 12 months?

Like previous, first we calculate cancellation rate by city hotel and resort hotel respectively, then we calculate the difference, and then use Bootstrap again to calculate 95% CI for the difference applying normal confidence limits. The result shows that, when in Jan, Feb, the difference is obvious because of low season for tourism, especially affecting the resort hotel, as fewer tourists so the booking orders in resort hotel. But when in Jul, Aug, it is high season for tourism, which means more people will consider about resort hotels, so the difference between 2 types is the smallest in a year circle. (see code part III)



Q2-c: Is there difference between cancellation rate by whether has special requests/has previous cancellations/required car parking spaces/been in waiting list/ is repeated guest or not these 5 control groups? What can you tell from that?

We can apply permutation test (Monte Carlo Sampling, resampling 9999 times) here to calculate and compare the difference on cancellation rate in two groups. (see code part IV). Here we take whether has special requests and whether has previous cancellations as example to analysis and show the summary results below. The other 3 control group results are also in part IV. The result of these 2 groups are as follows:

1.Testing whether has special requests

Approximative Two-Sample Fisher-Pitman Permutation Test
data: score by
treatment (has special requests, no special requests)
Z = -91.472, p-value < 1e-04
alternative hypothesis: true mu is not equal to 0

2. Testing whether has previous cancellations

Approximative Two-Sample Fisher-Pitman Permutation Test
data: score by
treatment (has previous cancellation, no previous cancellation)
Z = 93.574, p-value < 1e-04
alternative hypothesis: true mu is not equal to 0

From the results above we can see that there is much difference in cancellation rate between group has special requests and no special requests, also the group whether has previous cancellations or not. It is reasonable because usually proposing more demands means more probability to live in the room in the hotel rather than cancellation. Besides, if a person has previous cancellation history, then he/she is more likely to cancel the booking than those who doesn't has any previous cancellation records. Also, for other groups, has required car parking spaces/been in waiting list/been repeated guest, after permutation test, they all show the same result that p-value is smaller than 0.05(see code part IV) and thus, we can get conclusion that there are differences in cancellation rate between these control groups. If a guest meets one or many of the following conditions: has proposed special requests or has no previous cancellations or has required car parking or has not been in waiting list or is a repeated guest, then he/she would be more likely not to cancel the booking. Besides, It is also interesting to find that, if I filter the data with customer who have car parking requirements, the cancellation rate now is real 0, with 7409 records from the dataset in total. It is reasonable because requiring car parking slots means that the customer is planning to drive here just in the near future, which is a signal for incoming check-in. So, among these conditions, I personally believe car parking requirements is a very powerful and strong metric to evaluate whether there will be a hotel booking cancellation for or not.

Question 3: Use machine learning methods to predict hotel cancellation. Figure out the test error and training error of each method. Which method is better?

For the hotel agents, the thing they need to be concerned about is whether the customers will cancel their bookings or not. If they can predict the cancellation tendency of the customers, they can avoid the financial loss of cancellation. To be specific, if the

customers have a high tendency to cancel their bookings, the hotel can change their reservation size in advance to cover the potential loss. Hotels made a lot of efforts on minimizing the loss from cancellation. They can build some models using machine learning and apply the best one that fits their customers. I would like to fit some models and see the accuracy of hotel cancellation models.

Question 3 of this project explores several methods to predict if the customers will cancel the reservation or not. I indicate four methods that help to make predictions. They are the Logistic Regression Method, Linear Discriminant Analysis, K-Nearest Neighbor Method, and Lasso Regression Method. For each of these methods, I calculated the training error and test error.

By conducting this problem, I help to figure out some potential models used by the hotels and evaluate the accuracy of each model.

- Set up:

1. Import libraries. Use *read.csv* to import the hotel_bookings.csv file.
2. Interpret each column and get familiar with the dataset.

I recognize that is_canceled represents if the customer canceled the reservation or not. If is_canceled=1, then the customer canceled the reservation; if is_canceled=0, then the customer doesn't cancel the reservation. Other variables are corresponding to principal components that affect customers' decisions.

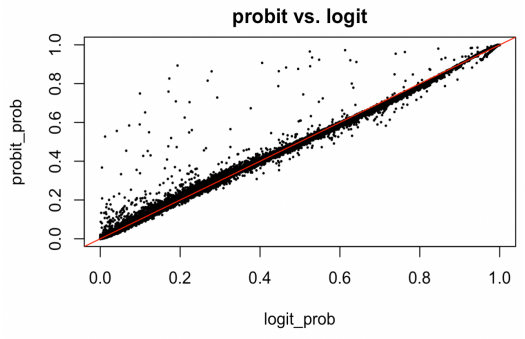
3. Clean the dataset.

Delete the variables that are useless. Also, convert the categorical variables into boolean values using function dimmyVars.

4. Set seed to make sure the test error and training error wouldn't change everytime.

- Model 1: Logistic Regression Method

1. Since the inverse of a logistic function is the logit function, I can use glm and set family = binomial('logit') to get logit values. Generalized linear models for binary data involve a link function logit which relates a linear function of the predictors to a function of the probability p. Logistic regression is based on the logit link function. In this case, we will link logit function with profit function. The graph is shown below.



The probit predictions values almost the same as logit values. That means the probit regression function will estimate as good as the logit regression function.

2. Logistic regression specifically estimates the probability of an observation as a particular class label. I will define a probability threshold for assigning class labels based on the probabilities returned by the glm fit. If the probability is larger than 50% identify as cancel. Fit a logistic regression to predict calcellation given all other features in the dataset using the glm function. Predict data and find the test error and training error of logistic regression method.

	train.error	test.error
logistic	0.1902944	0.1920141
lda	NA	NA
knn	NA	NA
lasso	NA	NA

The training error for the logistic regression method is 0.1902944, the test error is 0.1920141..

- Model 2: Linear Discriminant Analysis (LDA)

1. There is a function called lda in the package. We can select the is_cancelled column of the hotel dataset and use lda to fit it. Then use the predict function to find the training error and test error of linear discriminant analysis.

	train.error	test.error
logistic	0.1902944	0.1920141
lda	0.2113183	0.2122725
knn	NA	NA
lasso	NA	NA

The training error for the linear discriminant analysis is 0.2113183, the test error is 0.2122725.

- Model 3: K-Nearest Neighbor Method

Since in the knn method, I need to use a for loop to figure out the error. R doesn't have enough capacity to loop large dataset. In this project, I have about 120 thousands rows. Thus, I selected about 6 thousands rows from the original data.

1. Before fitting the data into models, I need to make sure the model does not have high bias or overfitting. I can find the right balance by the method called "Cross Validation for Model Selection". I use 10-fold cross validation to select the best number of neighbors best.kfold out of six values of k in 1, 10, 20, 30, 40, 50. For each model out of 6, divide data into 10 (roughly) equal sized chunks. Choose a test set (1 of the 10 chunks). The rest are training sets. Fit the model on training sets. Compare prediction vs true outcomes on the test set only. Repeat 10 times using each chunk as the test set exactly once. Save the test error for this model, averaged over 10 test sets. Choose the model that minimizes the test error.

train.error <dbl>	val.error <dbl>	neighbors <dbl>
0.003762688	0.3110254	1
0.255750400	0.3229222	10
0.278572921	0.3164722	20
0.288405178	0.3156654	30
0.300297911	0.3176831	40
0.304486092	0.3203016	50

The best k-fold neighbors value is 1 since the validation error is the least. However, when best k-fold = 1, it is easy to overfit the data. In order to correct it, I choose the neighbor with the second small value of validation error, which is 30. Note, KNN cannot tell which variables are important.

2. Use the best k-fold I calculated (30) to find the test error and training error of knn method.

	train.error	test.error
logistic	0.1902944	0.1920141
lda	0.2113183	0.2122725
knn	0.2860310	0.3370000
lasso	NA	NA

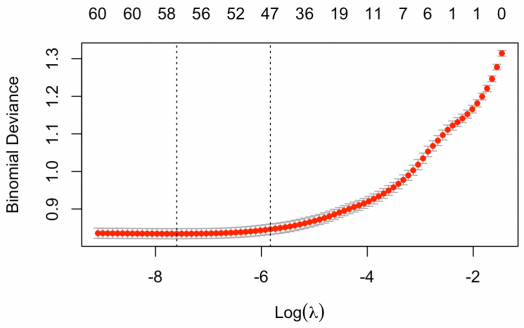
The training error for the knn method is 0.2860310, the test error is 0.337.

- Model 4: Lasso Regression Method

One way to control overfitting in logistic regression is through regularization. Since the Lasso Regression predict is too complex (we need to find the optimal value of lambda), if I run the full dataset, RStudio will crash. Thus I will use about 6000 rows instead of 120 thousands rows.

1. I will run k-fold cross validation and select the best regularization parameter for the logistic regression with Lasso penalty. The best lambda value is the one that can minimize the main square error (MSE) value. The best lambda value is 0.0005009646 in cross validation.

The plot of cross validation is shown below.



2. Find the test error and training error of lasso regression method.

	train.error	test.error
logistic	0.1902944	0.1920141
lda	0.2113183	0.2122725
knn	0.2860310	0.3370000
lasso	0.2062084	0.2020000

The training error for the lasso regression method is 0.2062084, the test error is 0.202.

By comparing the test error of those 4 models, we can figure out that Logistic Regression Method is the best method in this dataset. Logistic Regression Method performs well in linear and is less likely to overfitting. This is a good prediction if the dataset is not based on normal assumptions. Since the rows are not normally distributed, Logistic Regression works. Also, logistic regression can outperform LDA if normal assumptions are not met.

Overall, the accuracy of the hotel cancellation prediction is about 80% for all models. Using machine learning models to help make predictions in this dataset is okay but not perfect.

CODE

```
library(tidyverse)
library(ggplot2)
data = read.csv("hotel_bookings.csv")
```

Data Cleaning

```
# Replace 'NULL' with 'NA'
datanew = mutate_all(data, ~na_if(., "NULL"))

# Calculate number of NA's and NA rate for each column
f1 = function(x){sum(is.na(x))}
f2 = function(x){round(sum(is.na(x))/length(x)*100, 2)}
df = data.frame(#colnames = colnames(datanew),
                        na_num = apply(datanew, 2, f1),
                        na_rate = apply(datanew, 2, f2))
df = df %>% arrange(desc(na_num))
df_na = df[1:4, ]
# Remove column comany
# Replace NA with 0, 'Unknown' and median, seperately
datanew = datanew %>% select(-company) %>%
                              mutate(agent = as.integer(ifelse(is.na(agent), 0, agent)),
                                     country = ifelse(is.na(country), 'Unknown', country),
                                     children = ifelse(is.na(children), median(children, na.rm
= TRUE), children))

# Remove rows with no customer
datanew = datanew %>% filter(adults != 0 | children != 0 | babies != 0)
```

Question 1

```
# Select needed variables and calculate interested values
data1 = datanew %>%
                    select(hotel, is_canceled, lead_time, month = arrival_date_month,
                           wknd_nights = stays_in_weekend_nights,
                           week_nights = stays_in_week_nights, adults,
                           children, babies, adr) %>%
                    mutate(total_person = adults + children + babies,
                             total_nights = wknd_nights + week_nights)
```

```
# Calculated monthly everage values
data1_cal = data1 %>% group_by(hotel, month) %>%
                                    summarise(cancel_rate = sum(is_canceled)/n(),
                                              avg_lead_time = mean(lead_time),
                                              avg_nights_wknd = mean(wknd_nights),
                                              avg_nights_week = mean(week_nights),
                                              avg_adults = mean(adults),
                                              avg_children = mean(children),
                                              avg_babies = mean(babies),
                                              avg_person = mean(total_person),
                                              avg_total_nights = mean(total_nights),
                                              avg_adr = mean(adr),
                                              .groups = 'drop_last')

# Line chart for ADR
data1_cal %>% ggplot(aes(x = factor(month, levels = month.name), y = avg_adr,
                                    group = hotel, color = hotel)) + geom_line() +
                  geom_point(aes(color = hotel), size = 2) + theme_bw() +
                  ylim(40, 190) + theme(text = element_text(size=20)) +
                  xlab('Month') + ylab('Average ADR')

# Line chart for number of nights staying
data1_cal %>% ggplot(aes(x = factor(month, levels = month.name), y = avg_total_nights,
                                    group = hotel, color = hotel)) + geom_line() +
                  geom_point(aes(color = hotel), size = 2) + theme_bw() +
                  #ylim(1.65, 2.05) + theme(text = element_text(size=20)) +
                  xlab('Month') + ylab('Average Nights Staying')

# Line chart for number of guests
data1_cal %>% ggplot(aes(x = factor(month, levels = month.name), y = avg_person,
                                    group = hotel, color = hotel)) + geom_line() +
                  geom_point(aes(color = hotel), size = 2) + theme_bw() +
                  theme(text = element_text(size=20)) +
                  xlab('Month') + ylab('Average Number of People')
```

Question 2

Basic

```
library(tidyverse)
library(dplyr)
```

```
library(tidyr)
library(ggplot2)
hotel_booking = read_delim("data_cleaned.csv",delim=",")
#Make sure is_canceled is double, and several conditions are factors not double
hotel_booking = hotel_booking%>%
  mutate(
          hotel=as.factor(hotel),
          arrival_date_year=as.factor(arrival_date_year),
          arrival_date_month=as.factor(arrival_date_month),
          arrival_date_day_of_month = as.factor(arrival_date_day_of_month),
          deposit_type=as.factor(deposit_type),
          agent=as.factor(agent),
          meal=as.factor(meal),
          country=as.factor(country),
          market_segment=as.factor(market_segment),
          distribution_channel=as.factor(distribution_channel),
          is_repeated_guest=as.factor(is_repeated_guest),
          reserved_room_type=as.factor(reserved_room_type),
          assigned_room_type=as.factor(assigned_room_type),
          customer_type=as.factor(customer_type),
          reservation_status=as.factor(reservation_status),
          )
#calculate bootstrap variance
boot_var_mean = function(X, B){
    # Bootstrap variance estimate for the mean estimator
    # X is our data
    # B is the number of Monte Carlo Samples
    n = length(X)
    mhat.boot = numeric(B)
    for(j in 1:B){
        X.boot = sample(X, n, replace = TRUE) # sample bootstrap data from F_n
        mhat.boot[j] = mean(X.boot) # Median Bootstrap samples
    }
    var.boot = var(mhat.boot) # Calculate bootstrap variance estimator
    mean.boot = mean(mhat.boot)
    return(list(mean.boot = mean.boot, var.boot=var.boot, mhat.boot = mhat.boot))
}
#calculate point estimate for mean
boot_mean = function(X){
    #bootstrap_res = boot_var_mean(X, B = 1000)
    #return (bootstrap_res$mean.boot)
```

```
        return(mean(X))
}
#calculate variance estimator
boot_var = function(X){
        bootstrap_res = boot_var_mean(X, B = 1000)
        return (bootstrap_res$var.boot)
}
alpha = qnorm(.975)
```

Code Part I

```
#check cancellation rate for different deposit type
deposit = hotel_booking %>% filter(!is.na(deposit_type)) %>%
group_by(deposit_type) %>% summarize(deposit_mean = mean(is_canceled)) %>%
print
# A tibble: 3 x 2
   deposit_type deposit_mean
   <fct>                <dbl>
1 No Deposit           0.284
2 Non Refund           0.994
3 Refundable           0.222
```

Code Part II

```
#calculate cancellation rate for different market segment type
market =    hotel_booking %>% filter(market_segment != "Undefined") %>%
group_by(market_segment) %>% summarize(market_mean =
boot_mean(is_canceled),market_var = boot_var(is_canceled),market_lwr    =
market_mean - alpha*sqrt(market_var), market_upr = market_mean +
alpha*sqrt(market_var) ) %>% print
# A tibble: 7 x 5
   market_segment market_mean market_var market_lwr market_upr
   <fct>                <dbl>      <dbl>      <dbl>      <dbl>
1 Aviation             0.221 0.000752          0.168      0.275
2 Complementary        0.122 0.000145          0.0987     0.146
3 Corporate            0.188 0.0000278         0.177      0.198
4 Direct               0.154 0.0000108         0.147      0.160
5 Groups               0.611 0.0000121         0.604      0.618
6 Offline TA/TO        0.343 0.00000883        0.337      0.349
7 Online TA            0.368 0.00000419        0.364      0.372
# put the plot from highest to lowest
div_ord_m = {market %>%
        arrange(market_mean)}$market_segment %>% as.character()
#plot cancellation rate by different market segment
filter(market) %>%
```

```
mutate(across(all_of(c("market_mean", "market_lwr", "market_upr")))) %>%
mutate(market_segment    = factor(as.character(market_segment), levels =
div_ord_m)) %>%
ggplot(aes(x = market_mean, y = market_segment)) + geom_point() +
geom_errorbarh( aes(xmin = market_lwr, xmax = market_upr)) +
                    theme_bw() + xlab("Cancellation rate by Market Segment") +
                    ylab("Market Segment") +
                    xlim(c(0,0.7))
#calculate cancellation rate for different customer type
customer =    hotel_booking %>% filter(!is.na(customer_type)) %>%
group_by(customer_type) %>% summarize(customer_mean =
boot_mean(is_canceled),customer_var = boot_var(is_canceled),customer_lwr    =
customer_mean - alpha*sqrt(customer_var), customer_upr = customer_mean +
alpha*sqrt(customer_var) ) %>% print
# A tibble: 4 x 5
   customer_type      customer_mean customer_var customer_lwr customer_upr
   <fct>                      <dbl>        <dbl>        <dbl>        <dbl>
1 Contract                   0.310     0.0000530        0.296        0.324
2 Group                      0.101     0.000157         0.0765       0.126
3 Transient                  0.408     0.00000278       0.405        0.411
4 Transient-Party            0.255     0.00000739       0.249        0.260
# put the plot from highest to lowest
div_ord_c = {customer %>%
      arrange(customer_mean)}$customer_type %>% as.character()
#plot cancellation rate by different customer type
filter(customer) %>%
mutate(across(all_of(c("customer_mean", "customer_lwr", "customer_upr")))) %>%
mutate(customer_type    = factor(as.character(customer_type), levels = div_ord_c)) %>%
ggplot(aes(x = customer_mean, y = customer_type)) + geom_point() +
geom_errorbarh( aes(xmin = customer_lwr, xmax = customer_upr)) +
                    theme_bw() + xlab("Cancellation rate by Customer Type") +
                    ylab("Customer Type") +
                    xlim(c(0,0.45))
```

Code Part III

```
#calculate cancellation rate for resort hotel by months
resort_by_month = hotel_booking %>% filter(hotel == "Resort Hotel") %>%
group_by(arrival_date_month) %>% summarize( resort_boot_mean =
boot_mean(is_canceled),resort_boot_var = boot_var(is_canceled)) %>% print
# A tibble: 12 x 3
    arrival_date_month resort_boot_mean resort_boot_var
    <fct>                        <dbl>             <dbl>
```

```
 1 April                                0.293          0.0000592
 2 August                               0.334          0.0000483
 3 December                             0.239          0.0000740
 4 February                             0.256          0.0000604
 5 January                              0.148          0.0000576
 6 July                                 0.314          0.0000507
 7 June                                 0.331          0.0000691
 8 March                                0.229          0.0000520
 9 May                                  0.288          0.0000575
10 November                             0.189          0.0000647
11 October                              0.275          0.0000580
12 September                            0.324          0.0000707
#calculate cancellation rate for city hotel by months
city_by_month = hotel_booking %>% filter(hotel == "City Hotel") %>%
group_by(arrival_date_month) %>% summarize( city_boot_mean =
boot_mean(is_canceled),city_boot_var = boot_var(is_canceled)) %>% print
# A tibble: 12 x 3
    arrival_date_month city_boot_mean city_boot_var
    <fct>                       <dbl>          <dbl>
 1 April                        0.463        0.0000339
 2 August                       0.401        0.0000263
 3 December                     0.422        0.0000633
 4 February                     0.384        0.0000504
 5 January                      0.397        0.0000661
 6 July                         0.409        0.0000292
 7 June                         0.447        0.0000327
 8 March                        0.371        0.0000357
 9 May                          0.444        0.0000317
10 November                     0.383        0.0000521
11 October                      0.430        0.0000322
12 September                    0.421        0.0000329
#calculate difference bewtween 2 type hotel by months
dif_by_month = resort_by_month %>% left_join(city_by_month, by =
"arrival_date_month") %>% mutate(dif_mean = city_boot_mean -
resort_boot_mean,dif_var=city_boot_var+resort_boot_var, dif_lwr = dif_mean -
alpha*sqrt(dif_var), dif_upr = dif_mean + alpha*sqrt(dif_var)) %>% print
# A tibble: 12 x 9
    arrival_date_month resort_boot_mean resort_boot_var city_boot_mean
    <fct>                        <dbl>           <dbl>           <dbl>
 1 April                         0.294       0.0000541           0.463
 2 August                        0.335       0.0000480           0.402
```

```
 3 December                0.238          0.0000700          0.423
 4 February                0.256          0.0000640          0.384
 5 January                 0.148          0.0000614          0.397
 6 July                    0.314          0.0000464          0.409
 7 June                    0.331          0.0000743          0.447
 8 March                   0.229          0.0000532          0.371
 9 May                     0.288          0.0000659          0.445
10 November                0.189          0.0000621          0.383
11 October                 0.275          0.0000557          0.431
12 September               0.324          0.0000652          0.420
# ... with 5 more variables: city_boot_var <dbl>, dif_mean <dbl>,
#      dif_var <dbl>, dif_lwr <dbl>, dif_upr <dbl>
# put the plot from highest to lowest
div_ord = {dif_by_month %>%
      arrange(dif_mean)}$arrival_date_month %>% as.character()
print(div_ord)
#plot cancellation rate by city and resort hotel by different months
filter(dif_by_month) %>%
mutate(across(all_of(c("dif_mean", "dif_lwr", "dif_upr")))) %>%
mutate(arrival_date_month = factor(as.character(arrival_date_month), levels =
div_ord)) %>%
ggplot(aes(x = dif_mean, y = arrival_date_month)) + geom_point() +
geom_errorbarh( aes(xmin = dif_lwr, xmax = dif_upr)) +
                    theme_bw() + xlab("Cancellation rate difference between City and
Resort Hotel by Month") +
                    ylab("Month") +
                    xlim(c(0,0.4))
```

Code Part IV

```
library(coin)
#divide data into 2 groups by different conditions, following are the same,5 groups in
total,
#here means no previous cancellations
pre1=hotel_booking[which(hotel_booking$previous_cancellations== 0), ]$is_canceled
#has previous cancellations
pre2=hotel_booking[which(hotel_booking$previous_cancellations!= 0), ]$is_canceled
#no special requests
spe1 = hotel_booking[which(hotel_booking$total_of_special_requests==
0), ]$is_canceled
#has special requests
spe2 = hotel_booking[which(hotel_booking$total_of_special_requests!=
0), ]$is_canceled
```

```
#no parking requirements
ca1 = hotel_booking[which(hotel_booking$required_car_parking_spaces ==
0), ]$is_canceled
#has parking requirements
ca2 = hotel_booking[which(hotel_booking$required_car_parking_spaces !=
0), ]$is_canceled
#not in waiting list
w1=hotel_booking[which(hotel_booking$days_in_waiting_list == 0), ]$is_canceled
#has been in waiting list
w2=hotel_booking[which(hotel_booking$days_in_waiting_list != 0), ]$is_canceled
#not repeated guest
r1=hotel_booking[which(hotel_booking$is_repeated_guest == 0), ]$is_canceled
#is repeated guest
r2=hotel_booking[which(hotel_booking$is_repeated_guest != 0), ]$is_canceled
score = c(pre1,pre2)
treatment <- factor(c(rep("no previous cancellation",length(pre1)),rep("has previous
cancellation",length(pre2))))
mydata <- data.frame(treatment, score)
oneway_test(score~treatment, data=mydata,distribution=approximate(nresample=9999))
Approximative Two-Sample Fisher-Pitman Permutation Test

data:    score by
        treatment (has previous cancellation, no previous cancellation)
Z = 93.574, p-value < 1e-04
alternative hypothesis: true mu is not equal to 0
score = c(spe1,spe2)
treatment <- factor(c(rep("no special requests",length(spe1)),rep("has special
requests",length(spe2))))
mydata <- data.frame(treatment, score)
oneway_test(score~treatment, data=mydata,distribution=approximate(nresample=9999))
Approximative Two-Sample Fisher-Pitman Permutation Test

data:    score by
        treatment (has special requests, no special requests)
Z = -91.472, p-value < 1e-04
alternative hypothesis: true mu is not equal to 0
score = c(ca1,ca2)
treatment <- factor(c(rep("no car parking demand",length(ca1)),rep("has car parking
demand",length(ca2))))
mydata <- data.frame(treatment, score)
oneway_test(score~treatment, data=mydata,distribution=approximate(nresample=9999))
```

```
Approximative Two-Sample Fisher-Pitman Permutation Test

data:  score by
       treatment (has car parking demand, no car parking demand)
Z = -68.227, p-value < 1e-04
alternative hypothesis: true mu is not equal to 0
score = c(w1,w2)
treatment <- factor(c(rep("not in waiting list",length(w1)),rep("in waiting
list",length(w2))))
mydata <- data.frame(treatment, score)
oneway_test(score~treatment, data=mydata,distribution=approximate(nresample=9999))
Approximative Two-Sample Fisher-Pitman Permutation Test

data:  score by
       treatment (in waiting list, not in waiting list)
Z = 34.254, p-value < 1e-04
alternative hypothesis: true mu is not equal to 0
score = c(r1,r2)
treatment <- factor(c(rep("not repeated guests",length(r1)),rep("is repeated
guests",length(r2))))
mydata <- data.frame(treatment, score)
oneway_test(score~treatment, data=mydata,distribution=approximate(nresample=9999))
Approximative Two-Sample Fisher-Pitman Permutation Test

data:  score by
       treatment (is repeated guests, not repeated guests)
Z = -28.915, p-value < 1e-04
alternative hypothesis: true mu is not equal to 0
```

Question 3

Set Up:

```
# install libraries
library(plyr)
library(class)
library(rpart)
library(tidyverse)
```

```
library(caret)
library(glmnet)

# import data
cancel <- read.csv("hotel_bookings.csv")
cancel
# explore columns
names(cancel)
# missing value?
sum(is.na(cancel))

# use cleaned dataset
cancel <- read.csv("data_cleaned.csv")

cancel <-cancel %>%
    mutate(arrival_date_month =as.character(
                match(substr(arrival_date_month,1,3),month.abb)),
           arrival_date_year = as.character(
                arrival_date_year),
          required_car_parking_spaces = as.character(required_car_parking_spaces)) %>%
    select(-reservation_status_date,
           -country,
           -reservation_status,
           -adr,
           -arrival_date_week_number,
           -arrival_date_day_of_month,
            -arrival_date_year)

# convert the the categorical variables into boolean values
dummy <- dummyVars(" ~ .", data = cancel, fullRank = TRUE)
cancel <- data.frame(predict(dummy, newdata =cancel))

# build up function to calculate error rate
calc_error_rate <- function(predicted.value, true.value){
return(mean(true.value!=predicted.value))
}

# built a matrix to add train error and test error in 3 methods
records = matrix(NA, nrow=4, ncol=2)
colnames(records) <- c("train.error","test.error")
rownames(records) <- c("logistic","lda", "knn", "lasso")
```

```
# set train data and test data
set.seed(1)
n <- nrow(cancel)
test.indices = sample(n, 0.2*n)
cancel.train = cancel[-test.indices,]
cancel.test = cancel[test.indices,]

# divided into 10 folds
nfold = 10
folds = seq.int(nrow(cancel.train)) %>% ## sequential obs ids
cut(breaks = nfold, labels=FALSE) %>% ## sequential fold ids
sample ## random fold ids

# assigned values to variables
XTrain <- select(cancel.train, -is_canceled)
YTrain <- cancel.train$is_canceled
XTest <- select(cancel.test, -is_canceled)
YTest <- cancel.test$is_canceled
```

 # Method 1:
 # Logistic Regression Method

```
logit <- glm(is_canceled ~ ., data= cancel.train, family=binomial('logit'))
probit <- glm(is_canceled ~ ., data=cancel.train, family = binomial(link="probit"))
logit_prob <- predict(logit, type="response")
probit_prob <- predict(probit, type="response")
plot(logit_prob, probit_prob, pch=19, cex=0.2, main="probit vs. logit")
abline(a=0, b=1, col="red")

# use logistic regression to perform classification
regression_cancel <- glm(is_canceled~ ., data=cancel.train, family="binomial")

# train error for logistic regression
pred.Train <- predict(regression_cancel, cancel.train, type="response")
train.error1 <- calc_error_rate(ifelse(pred.Train>1/2, 1, 0), YTrain)
train.error1
```

```
# test error for logistic regression
pred.Test <- predict(regression_cancel, cancel.test, type="response")
test.error1 <- calc_error_rate(ifelse(pred.Test>1/2, 1, 0), YTest)
test.error1

# fill in record
records[1,] <- c(train.error1, test.error1)
kbl(records, booktabs = T) %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
full_width=FALSE)
```

 # Method 2:
 # Linear Discriminant Analysis (LDA)

```
# train error for Linear Discriminant Analysis
library(MASS)
lda.fit <- lda(is_canceled~., data=cancel.train)
lda_fitted <- lda.fit$posterior[,2]
pred.Train <- predict(lda.fit, cancel.train)$posterior[,2]
train.error2 <- calc_error_rate(ifelse(pred.Train > 1/2, 1, 0), YTrain)
train.error2

# test error for Linear Discriminant Analysis
pred.Test <- predict(lda.fit, cancel.test)$posterior[,2]
test.error2 <- calc_error_rate(ifelse(pred.Test > 1/2, 1, 0), YTest)
test.error2

# fill in record
records[2,] <- c(train.error2, test.error2)
kbl(records, booktabs = T) %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
full_width=FALSE)
```

 # Method 3:
 # K-Nearest Neighbor Method (KNN)

```
# since the dataset is too large for R to run for loop, I selected 6 thousands rows out of 120
thousands rows in this part.

certainRow <- c(1)
```

```
for (i in c(1:nrow(cancel))){
  if (i%%20==0) certainRow = c(certainRow,i)
}
cancel_tmp <- cancel[c(certainRow),]

# set train data and test data
set.seed(1)
test.indices = sample(1:nrow(cancel_tmp), 1000)
cancel_tmp.train = cancel_tmp[-test.indices,]
cancel_tmp.test = cancel_tmp[test.indices,]

# divided into 10 folds
nfold = 10
folds = seq.int(nrow(cancel_tmp.train)) %>% ## sequential obs ids
cut(breaks = nfold, labels=FALSE) %>% ## sequential fold ids
sample ## random fold ids

# built up the function do.chunk to use in the for loop below
do.chunk <- function(chunkid, folddef, Xdat, Ydat, k){
train = (folddef!=chunkid)
Xtr = Xdat[train,]
Ytr = Ydat[train]
Xvl = Xdat[!train,]
Yvl = Ydat[!train]
# get classifications for current training chunks
predYtr = knn(train = Xtr, test = Xtr, cl = Ytr, k = k)
# get classifications for current test chunk
predYvl = knn(train = Xtr, test = Xvl, cl = Ytr, k = k)
data.frame(train.error = calc_error_rate(predYtr, Ytr),
val.error = calc_error_rate(predYvl, Yvl))
}

# assigned values to variables to simplify the elements in the for loop
error <- NULL
kvec <- c(1, seq(10, 50, length.out=5))
XTrain <- subset(cancel_tmp.train, select = -is_canceled)
YTrain <- cancel_tmp.train$is_canceled
XTest <- subset(cancel_tmp.test, select = -is_canceled)
YTest <- cancel_tmp.test$is_canceled

# figuring out train.error and val.error for different neighbors
```

```
for (k in kvec){
  repeatChunk <- ldply(1:nfold, do.chunk,
                       folddef = folds,
                       Xdat = XTrain,
                       Ydat = YTrain, k=k) %>%
  summarise_all(funs(mean)) %>%
  mutate(neighbors = k)
  error <- rbind(error,repeatChunk)}
error

# find the best kfold based on the smallest value of val.error
# selecting neighbors
best.kfold <- error$neighbors[which.min(error$val.error)]
best.kfold

# train error for knn
pred.YTrain <- knn(train=XTrain, test=XTrain, cl=YTrain, k=30)
train.error3 <- calc_error_rate(pred.YTrain, YTrain)
train.error3

# test error for knn
pred.YTest <- knn(train=XTrain, test=XTest, cl=YTrain, k=30)
test.error3 <- calc_error_rate(pred.YTest, YTest)
test.error3

# fill in record
records[3,] <- c(train.error3, test.error3)
kbl(records, booktabs = T) %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
full_width=FALSE)
```

 # Method 4:
 # Lasso Regression Method

```
x = model.matrix(is_canceled~., cancel_tmp.train)[,-1]
y = ifelse(cancel_tmp.train$is_canceled == "1", 1, 0)
cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial")
best_lambda <- cv.lasso$lambda.min
best_lambda
coef(cv.lasso, best_lambda)
```

```
plot(cv.lasso)

lasso.model <- cv.glmnet(x, y, alpha = 1, family = "binomial")
lasso.model

# train error for lasso regression
pred.Train <- lasso.model %>% predict(newx = x)
train.error4 <- calc_error_rate(ifelse(pred.Train>1/2, 1, 0),
                                cancel_tmp.train$is_canceled)
train.error4

# test error for lasso regression
newx <- model.matrix(is_canceled~., cancel_tmp.test)[,-1]
pred.Test <- lasso.model %>% predict(newx = newx)
test.error4 <- calc_error_rate(ifelse(pred.Test>1/2, 1, 0),
                               cancel_tmp.test$is_canceled)
test.error4

# fill in record
records[4,] <- c(train.error4, test.error4)
kbl(records, booktabs = T) %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
full_width=FALSE)
```