```
install.packages('quanteda')
install.packages('readr')
install.packages('readxl')
install.packages('ggplot2')
install.packages('ggmap')
install.packages('countrycode')
install.packages('lubridate')
install.packages('MASS')
install.packages('arules')
install.packages('arulesViz')
install.packages('jtools')
library(quanteda); library(readr);library(readxl)
library(ggplot2);library(ggmap);library(countrycode)
library(lubridate);library(MASS)
library(arules); library(arulesViz)
library(jtools)
#####################################################################
#Data input
#####################################################################
setwd("/Users/tonyj/Desktop/IST 687/Final project")
getwd()
Resort <- read_excel("H1-Resort.xlsx")
City <- read_excel("H2-City.xlsx")
#####################################################################
#Exploratory work
#####################################################################
#Descriptive Statistics
summary(City)
summary(Resort)
#####################################################################
#Numeric Variables
#####################################################################
#ADR
hist(City$ADR)
hist(Resort$ADR)
#####################################################################
#Categorical variables
#####################################################################
#Data Cleaning
############################
getSeason <- function(DATES) {
  WS <- as.Date("2015-12-15","2016-12-15","2017-12-15", format = "%Y-%m-%d") # Winter
  SE <- as.Date("2015-3-15","2016-3-15","2017-3-15",  format = "%Y-%m-%d") # Spring
  SS <- as.Date("2015-6-15","2016-6-15","2017-6-15",  format = "%Y-%m-%d") # Summer
```

```
  FE <- as.Date("2015-9-15","2016-9-15","2017-9-15",  format = "%Y-%m-%d") # Fall

  # Convert dates from any year to 2015-2017 dates
  d <- as.Date(strftime(DATES, format="2015-%m-%d"))

  ifelse (d >= WS | d < SE, 4,
          ifelse (d >= SE & d < SS, 1,
                  ifelse (d >= SS & d < FE, 2, 3)))
}
my.dates <- City$`Arrival Date`
City$Season<-data.frame(getSeason(my.dates))
my.dates2 <- Resort$`Arrival Date`
Resort$Season<-data.frame(getSeason(my.dates2))
City$ArrivalMonth <- month(City$`Arrival Date`)
Resort$ArrivalMonth <- month(Resort$`Arrival Date`)
City<-City[City$ADR!=5400,]
Resort<-Resort[Resort$ReservationStatus!='`',]
City$ArrivalMonth<-as.factor(City$ArrivalMonth)
Resort$ArrivalMonth<-as.factor(Resort$ArrivalMonth)
colSums(is.na(City))#--checking NAs
colSums(is.na(Resort))
City <- na.omit(City)
Resort <- na.omit(Resort)
City$CountryName <- countrycode(City$Country, origin = "iso3c", destination = "country.name")
Resort$CountryName <- countrycode(Resort$Country, origin = "iso3c", destination =
"country.name")
##########################################
#Agent: ID of the travel agency that made the booking
table(sort(City$Agent))
table(sort(Resort$Agent))
#ArrivalDateMonth
table(sort(City$`Arrival Date`))
table(sort(Resort$`Arrival Date`))
#AssignedRoomType:Code for the type of room assigned to the booking. Sometimes the
assigned room type differs from the reserved room type due to hotel operation reasons
(e.g.overbooking) or by customer request. Code is presented instead of designation for
anonymity reasons
table(sort(City$AssignedRoomType))
table(sort(Resort$AssignedRoomType))
sort(table(City$AssignedRoomType),decreasing = TRUE) #auto ranking
sort(table(Resort$AssignedRoomType),decreasing = TRUE)

#Company:ID of the company/entity that made the booking or responsible for paying the
booking. ID is presented instead of designation for anonymity reasons
```

```
table(sort(City$Company))
table(sort(Resort$Company))
#Country: Country of origin. Categories are represented in the ISO 3155-3:2013 format
table(sort(City$Country))
table(sort(Resort$Country))
#CustomerType: Type of booking, assuming one of four categories:
#Contract - when the booking has an allotment or other type of contract associated to it;
#Group - when the booking is associated to a group;
#Transient - when the booking is not part of a group or contract, and is not associated to other transient booking;
#Transient-party - when the booking is transient, but is associated to at least other transient booking
table(sort(City$CustomerType))
table(sort(Resort$CustomerType))
sort(table(City$CustomerType),decreasing = TRUE)
sort(table(Resort$CustomerType),decreasing = TRUE)
#DepositType: Indication on if the customer made a deposit to guarantee the booking. This variable can assume three categories:
#No Deposit - no deposit was made;
#Non Refund - a deposit was made in the value of the total stay cost;
#Refundable - a deposit was made with a value under the total cost of stay
table(sort(City$DepositType))
table(sort(Resort$DepositType))
#DistributionChannel:  Booking distribution channel. The term "TA" means "Travel Agents" and "TO" means "Tour Operators"
table(sort(City$DistributionChannel))
table(sort(Resort$DistributionChannel))
sort(table(City$DistributionChannel),decreasing = TRUE)
sort(table(Resort$DistributionChannel),decreasing = TRUE)
#IsCanceled:Value indicating if the booking was canceled (1) or not (0)
table(sort(City$IsCanceled))
table(sort(Resort$IsCanceled))
#IsRepeatedGuest:Value indicating if the booking name was from a repeated guest (1) or not (0)
table(sort(City$IsRepeatedGuest))
table(sort(Resort$IsRepeatedGuest))
#MarketSegment:Market segment designation. In categories, the term "TA" means "Travel Agents" and "TO" means "Tour Operators"
table(sort(City$MarketSegment))
table(sort(Resort$MarketSegment))
sort(table(City$MarketSegment),decreasing = TRUE)
sort(table(Resort$MarketSegment),decreasing = TRUE)
#Meal: Type of meal booked. Categories are presented in standard hospitality meal packages:
#Undefined/SC - no meal package;
```

```
#BB - Bed & Breakfast;
#HB - Half board (breakfast and one other meal - usually dinner);
#FB - Full board (breakfast, lunch and dinner)
table(sort(City$Meal))
table(sort(Resort$Meal))
sort(table(City$Meal),decreasing = TRUE)
sort(table(Resort$Meal),decreasing = TRUE)
#ReservationStatus:Reservation last status, assuming one of three categories:
#Canceled - booking was canceled by the customer;
#Check-Out - customer has checked in but already departed;
#No-Show - customer did not check-in and did inform the hotel of the reason why
table(sort(City$ReservationStatus))
table(sort(Resort$ReservationStatus))
sort(table(City$ReservationStatus),decreasing = TRUE)
sort(table(Resort$ReservationStatus),decreasing = TRUE)
#ReservedRoomType: Code of room type reserved. Code is presented instead of designation
for anonymity reasons
table(sort(City$ReservedRoomType))
table(sort(Resort$ReservedRoomType))
sort(table(City$ReservedRoomType),decreasing = TRUE)
sort(table(Resort$ReservedRoomType),decreasing = TRUE)
##################################################################
#BoxPlotsofADR
##################################################################
#Agent
#ggplot(City, aes(x=Agent, y=ADR)) + geom_boxplot()
#ggplot(Resort, aes(x=Agent, y=ADR)) + geom_boxplot()
#ArrivalDate
#ggplot(City, aes(x=`Arrival Date`, y=ADR)) + geom_boxplot()
#ggplot(Resort, aes(x=`Arrival Date`, y=ADR)) + geom_boxplot()

##AssignedRoomType
ggplot(City, aes(x=AssignedRoomType, y=ADR,fill=AssignedRoomType)) +
geom_boxplot()+ggtitle('City_AssignedRoomType')
ggplot(Resort, aes(x=AssignedRoomType, y=ADR,fill=AssignedRoomType)) +
geom_boxplot()+ggtitle('Resort_AssignedRoomType')
#Company
#ggplot(City, aes(x=Company, y=ADR)) + geom_boxplot()
#ggplot(Resort, aes(x=Company, y=ADR)) + geom_boxplot()
##CustomerType
ggplot(City, aes(x=CustomerType, y=ADR,fill=CustomerType)) +
geom_boxplot()+ggtitle('City_CustomerType')
ggplot(Resort, aes(x=CustomerType, y=ADR,fill=CustomerType)) +
geom_boxplot()+ggtitle('Resort_CustomerType')
```

```
##DepositType
ggplot(City, aes(x=DepositType, y=ADR,fill=DepositType)) + geom_boxplot()
ggplot(Resort, aes(x=DepositType, y=ADR,fill=DepositType)) + geom_boxplot()
##DistributionChannel
ggplot(City, aes(x=DistributionChannel, y=ADR,fill=DistributionChannel)) + geom_boxplot()
ggplot(Resort, aes(x=DistributionChannel, y=ADR,fill=DistributionChannel)) + geom_boxplot()
##IsCanceled(hist or table)
ggplot(City, aes(x=IsCanceled, y=ADR)) + geom_boxplot()
ggplot(Resort, aes(x=IsCanceled, y=ADR)) + geom_boxplot()
##IsRepeatedGuest(hist or table)
ggplot(City, aes(x=IsRepeatedGuest, y=ADR)) + geom_boxplot()
ggplot(Resort, aes(x=IsRepeatedGuest, y=ADR)) + geom_boxplot()
##MarketSegment
ggplot(City, aes(x=MarketSegment, y=ADR,fill=MarketSegment)) + geom_boxplot()
ggplot(Resort, aes(x=MarketSegment, y=ADR,fill=MarketSegment)) + geom_boxplot()
##Meal
ggplot(City, aes(x=Meal, y=ADR,fill=Meal)) + geom_boxplot()
ggplot(Resort, aes(x=Meal, y=ADR,fill=Meal)) + geom_boxplot()
##ReservationStatus
ggplot(City, aes(x=ReservationStatus, y=ADR,fill=ReservationStatus)) + geom_boxplot()
ggplot(Resort, aes(x=ReservationStatus, y=ADR,fill=ReservationStatus)) + geom_boxplot()
##ReservedRoomType
ggplot(City, aes(x=ReservedRoomType, y=ADR,fill=ReservedRoomType)) + geom_boxplot()
ggplot(Resort, aes(x=ReservedRoomType, y=ADR,fill=ReservedRoomType)) + geom_boxplot()
#####################################################################
#Maps
#####################################################################
#Maps
world <- map_data("world")

world$region <- countrycode(world$region, origin = "country.name", destination = "iso3c")

dfNew <- merge(world, countryCity, by="region")
dfNew<-dfNew[!duplicated(dfNew$region), ]

dfNew2 <- merge(world, countryResort, by="region")
dfNew2<-dfNew2[!duplicated(dfNew2$region), ]

#MapCity
mapCity<-ggplot() +
  geom_map(aes(map_id = region), map = world, data = world, fill="white", color="grey") +
  expand_limits(x = dfNew$long, y = dfNew$lat)
mapCity<-mapCity+ geom_point(aes(x = dfNew$long, y =
dfNew$lat,color=dfNew$Freq,size=dfNew$Freq))
```

```
mapCity
#MapResort
mapResort<-ggplot() +
  geom_map(aes(map_id = region), map = world, data = world, fill="white", color="grey") +
  expand_limits(x = dfNew2$long, y = dfNew2$lat)
mapResort<-mapResort+ geom_point(aes(x = dfNew2$long, y =
dfNew2$lat,color=dfNew2$Freq,size=dfNew2$Freq))
mapResort
#################################################################
#convert fields into factor variables & Association Rules Mining
#city

CityCon <- data.frame(IsCanceled=as.factor(City$IsCanceled),
                      LeadTime=as.factor(City$LeadTime),
                      PreviousCancellations=as.factor(City$PreviousCancellations),
                      IsRepeatedGuest=as.factor(City$IsRepeatedGuest),
                      BookingChanges=as.factor(City$BookingChanges),
                      RequiredCarParkingSpaces=as.factor(City$RequiredCarParkingSpaces),
                      TotalOfSpecialRequests=as.factor(City$TotalOfSpecialRequests))

str(CityCon)
CityCon.trans <- as(CityCon, "transactions")
ruleset.City <- apriori(CityCon.trans, parameter=list(support=0.005, confidence=0.6),
                        appearance=list(default="lhs",rhs=("IsCanceled=1")))
inspect(ruleset.City)


#resort

ResortCon <- data.frame(IsCanceled=as.factor(Resort$IsCanceled),
                      LeadTime=as.factor(Resort$LeadTime),
                      PreviousCancellations=as.factor(Resort$PreviousCancellations),
                      IsRepeatedGuest=as.factor(Resort$IsRepeatedGuest),
                      BookingChanges=as.factor(Resort$BookingChanges),
                      RequiredCarParkingSpaces=as.factor(Resort$RequiredCarParkingSpaces),
                      TotalOfSpecialRequests=as.factor(Resort$TotalOfSpecialRequests))

ResortCon.trans <- as(ResortCon, "transactions")
ruleset.Resort <- apriori(ResortCon.trans, parameter=list(support=0.005, confidence=0.6),
                          appearance=list(default="lhs",rhs=("IsCanceled=1")))
inspect(ruleset.Resort)

#################################################################
#linear models
```

```
modCanCity <- lm(formula=IsCanceled ~
LeadTime+PreviousCancellations+IsRepeatedGuest+BookingChanges+RequiredCarParkingSp
aces+TotalOfSpecialRequests+ArrivalMonth, data=City)
summary(modCanCity)
effect_plot(modCanCity, pred = LeadTime, interval = TRUE)+ylim(0, 1)
effect_plot(modCanCity, pred = IsRepeatedGuest, interval = TRUE)+ylim(0, 1)
effect_plot(modCanCity, pred = BookingChanges, interval = TRUE)+ylim(0, 1)
effect_plot(modCanCity, pred = PreviousCancellations, interval = TRUE)+ xlim(0, 5)+ylim(0, 1)
effect_plot(modCanCity, pred = TotalOfSpecialRequests, interval = TRUE)+ xlim(0, 5)+ylim(0,
1)
effect_plot(modCanCity, pred = ArrivalMonth, interval = TRUE)+ylim(0, 1)
modCanResort <- lm(formula=IsCanceled ~
LeadTime+PreviousCancellations+IsRepeatedGuest+BookingChanges+RequiredCarParkingSp
aces+TotalOfSpecialRequests+ArrivalMonth, data=Resort)
summary(modCanResort)
effect_plot(modCanResort, pred = LeadTime, interval = TRUE)+ylim(0, 1)
effect_plot(modCanResort, pred = IsRepeatedGuest, interval = TRUE)+ylim(0, 1)
effect_plot(modCanResort, pred = BookingChanges, interval = TRUE)+ylim(0, 1)
effect_plot(modCanResort, pred = PreviousCancellations, interval = TRUE)+ xlim(0, 20)+ylim(0,
1)
effect_plot(modCanResort, pred = TotalOfSpecialRequests, interval = TRUE)+ xlim(0,
5)+ylim(0, 1)
effect_plot(modCanResort, pred = ArrivalMonth, interval = TRUE)+ylim(0, 1)
modADRCity <- lm(formula=ADR ~
AssignedRoomType+ReservedRoomType+MarketSegment+ArrivalMonth, data=City)
summary(modADRCity)
predADR<-
data.frame(AssignedRoomType='G',ReservedRoomType='G',MarketSegment='Direct')
predict(modADRCity, predADR)
effect_plot(modADRCity, pred = MarketSegment, interval = TRUE)+ expand_limits(y=c(0, 150))
effect_plot(modADRCity, pred = AssignedRoomType, interval = TRUE)+ expand_limits(y=c(0,
150))
effect_plot(modADRCity, pred = ReservedRoomType, interval = TRUE)+ expand_limits(y=c(0,
150))
effect_plot(modADRCity, pred = ArrivalMonth, interval = TRUE)+ expand_limits(y=c(0, 150))
modADRResort <- lm(formula=ADR ~
AssignedRoomType+ReservedRoomType+MarketSegment+ArrivalMonth, data=Resort)
summary(modADRResort)
effect_plot(modADRResort, pred = MarketSegment, interval = TRUE)+ expand_limits(y=c(0,
150))
effect_plot(modADRResort, pred = AssignedRoomType, interval = TRUE)+
expand_limits(y=c(0, 150))
effect_plot(modADRResort, pred = ReservedRoomType, interval = TRUE)+
expand_limits(y=c(0, 150))
```

```
effect_plot(modADRResort, pred = ArrivalMonth, interval = TRUE)+ expand_limits(y=c(0, 150))

#################################################################
#This is Scratch Idea of using Reservation season to draw some business insight
#load City-H2
Citydf <- read_excel("C:/Users/Justin/Desktop/Hotel Project/H2-City.xlsx")

#Divide Reservation dates by appropriate season

#1 Extract 6th and 7th letter (month) from reservation date and save to reservation month
Citydf$ReservationMonth <- substr(Citydf$ReservationStatusDate, 6,7)

#2 Change ReservationMonth's property to numbers
Citydf$ReservationMonth<-as.numeric(Citydf$ReservationMonth)

#3 Call month(12,1,2 to winter; 3,4,5 to spring; 6,7,8 to summer; 9,10,11 to Fall)
Citydf<-Citydf %>% mutate(ReservationSeason=ifelse(ReservationMonth %in% c(12,1,2),
"Winter", ifelse(ReservationMonth %in% c(3,4,5), "Spring", ifelse(ReservationMonth %in%
c(6,7,8), "Summer", ifelse(ReservationMonth %in% c(9,10,11), "Fall", "Error")))))

#4 Change data type of ReservationSeason, Status into factor and Status Date into Date
Citydf$ReservationStatus <- as.factor(Citydf$ReservationStatus)
Citydf$ReservationSeason <- factor(Citydf$ReservationSeason, levels= c("Spring", "Summer",
"Fall", "Winter"), ordered=T)
Citydf$ReservationStatusDate <-as.Date(Citydf$ReservationStatusDate)

#5 Draw graph of number of reservations(green) and number of cancellation(red) by month
ggplot(data=Citydf)+ geom_bar(aes(x=ReservationMonth), fill="green") +
stat_summary(aes(x=ReservationMonth,y=IsCanceled),fun=sum, geom="bar", fill="red",
position="identity")

#6 Draw graph of number of reservations(green) and number of cancellation(red) by season
ggplot(data=Citydf)+ geom_bar(aes(x=ReservationSeason), fill="green") +
stat_summary(aes(x=ReservationSeason, y = IsCanceled), fun=sum, geom="bar")
help("geom_histogram")
# looking at the histogram, we can make hypothesis that the percentage of cancellation is
highest during winter, especially January.



library(readxl)
library(dplyr)
```

```
library(ggplot2)
library(e1071)
library(caret)

#load data
Citydf <- read_excel("SUBSTITUTE WITH FILE LOCATION/H2-City.xlsx")
Resortdf <- read_excel("SUBSTITUTE WITH FILE LOCATION/H1-Resort.xlsx")

#Divide Reservation dates by appropriate season

#1 Extract 6th and 7th letter (month) from reservation date and save to reservation month
Citydf$ReservationMonth <- substr(Citydf$ReservationStatusDate, 6,7)
Citydf$ArrivalMonth <- substr(Citydf$`Arrival Date`, 6,7)

Resortdf$ReservationMonth <- substr(Resortdf$ReservationStatusDate, 6,7)
Resortdf$ArrivalMonth <- substr(Resortdf$`Arrival Date`,6,7)

#2 Change ReservationMonth's property to numbers
Citydf$ReservationMonth<-as.numeric(Citydf$ReservationMonth)
Resortdf$ReservationMonth <-as.numeric(Resortdf$ReservationMonth)

#3 Call month(12,1,2 to winter; 3,4,5 to spring; 6,7,8 to summer; 9,10,11 to Fall)
Citydf <- Citydf %>% mutate(ReservationSeason=ifelse(ReservationMonth %in% c(12,1,2),
"Winter", ifelse(ReservationMonth %in% c(3,4,5), "Spring", ifelse(ReservationMonth %in%
c(6,7,8), "Summer", ifelse(ReservationMonth %in% c(9,10,11), "Fall", "Error"))))
Citydf <- Citydf %>% mutate(ArrivalSeason=ifelse(ArrivalMonth %in% c(12,1,2), "Winter",
ifelse(ArrivalMonth %in% c(3,4,5), "Spring", ifelse(ArrivalMonth %in% c(6,7,8), "Summer",
ifelse(ArrivalMonth %in% c(9,10,11), "Fall", "Error"))))
Resortdf <- Resortdf %>% mutate(ReservationSeason=ifelse(ReservationMonth %in%
c(12,1,2), "Winter", ifelse(ReservationMonth %in% c(3,4,5), "Spring",
ifelse(ReservationMonth %in% c(6,7,8), "Summer", ifelse(ReservationMonth %in% c(9,10,11),
"Fall", "Error"))))
Resortdf <- Resortdf %>% mutate(ArrivalSeason=ifelse(ArrivalMonth %in% c(12,1,2), "Winter",
ifelse(ArrivalMonth %in% c(3,4,5), "Spring", ifelse(ArrivalMonth %in% c(6,7,8), "Summer",
ifelse(ArrivalMonth %in% c(9,10,11), "Fall", "Error"))))

#Creating SVM Model

#1 Change data type of variables in Citydf to factor
Citydf$LeadTime <- as.factor(Citydf$LeadTime)
Citydf$ReservationStatus <- as.factor(Citydf$ReservationStatus)
Citydf$StaysInWeekendNights <- as.factor(Citydf$StaysInWeekendNights)
Citydf$StaysInWeekNights <- as.factor(Citydf$StaysInWeekNights)
Citydf$Meal <- as.factor(Citydf$Meal)
```

```
Citydf$Country <- as.factor(Citydf$Country)
Citydf$MarketSegment <- as.factor(Citydf$MarketSegment)
Citydf$DistributionChannel <- as.factor(Citydf$DistributionChannel)
Citydf$IsRepeatedGuest <- as.factor(Citydf$IsRepeatedGuest)
Citydf$PreviousBookingsNotCanceled <- as.factor(Citydf$PreviousBookingsNotCanceled)
Citydf$PreviousCancellations <- as.factor(Citydf$PreviousCancellations)
Citydf$ReservedRoomType <- as.factor(Citydf$ReservedRoomType)
Citydf$AssignedRoomType <- as.factor(Citydf$AssignedRoomType)
Citydf$BookingChanges <- as.factor(Citydf$BookingChanges)
Citydf$DepositType <- as.factor(Citydf$DepositType)
Citydf$Agent <- as.factor(Citydf$Agent)
Citydf$Company <- as.factor(Citydf$Company)
Citydf$DaysInWaitingList <- as.factor(Citydf$DaysInWaitingList)
Citydf$CustomerType <- as.factor(Citydf$CustomerType)
Citydf$ADR <- as.factor(Citydf$ADR)
Citydf$RequiredCarParkingSpaces <- as.factor(Citydf$RequiredCarParkingSpaces)
Citydf$TotalOfSpecialRequests <- as.factor(Citydf$TotalOfSpecialRequests)
Citydf$ArrivalMonth <- as.factor(Citydf$ArrivalMonth)
Citydf$ArrivalSeason <- as.factor(Citydf$ArrivalSeason)
Citydf$ReservationMonth <- as.factor(Citydf$ReservationMonth)
Citydf$ReservationSeason <- as.factor(Citydf$ReservationSeason)

#2 Change data type of variables in Resortdf to factor
Resortdf$LeadTime <- as.factor(Resortdf$LeadTime)
Resortdf$ReservationStatus <- as.factor(Resortdf$ReservationStatus)
Resortdf$StaysInWeekendNights <- as.factor(Resortdf$StaysInWeekendNights)
Resortdf$StaysInWeekNights <- as.factor(Resortdf$StaysInWeekNights)
Resortdf$Meal <- as.factor(Resortdf$Meal)
Resortdf$Country <- as.factor(Resortdf$Country)
Resortdf$MarketSegment <- as.factor(Resortdf$MarketSegment)
Resortdf$DistributionChannel <- as.factor(Resortdf$DistributionChannel)
Resortdf$IsRepeatedGuest <- as.factor(Resortdf$IsRepeatedGuest)
Resortdf$PreviousBookingsNotCanceled <- as.factor(Resortdf$PreviousBookingsNotCanceled)
Resortdf$PreviousCancellations <- as.factor(Resortdf$PreviousCancellations)
Resortdf$ReservedRoomType <- as.factor(Resortdf$ReservedRoomType)
Resortdf$AssignedRoomType <- as.factor(Resortdf$AssignedRoomType)
Resortdf$BookingChanges <- as.factor(Resortdf$BookingChanges)
Resortdf$DepositType <- as.factor(Resortdf$DepositType)
Resortdf$Agent <- as.factor(Resortdf$Agent)
Resortdf$Company <- as.factor(Resortdf$Company)
Resortdf$DaysInWaitingList <- as.factor(Resortdf$DaysInWaitingList)
Resortdf$CustomerType <- as.factor(Resortdf$CustomerType)
Resortdf$ADR <- as.factor(Resortdf$ADR)
Resortdf$RequiredCarParkingSpaces <- as.factor(Resortdf$RequiredCarParkingSpaces)
```

```
Resortdf$TotalOfSpecialRequests <- as.factor(Resortdf$TotalOfSpecialRequests)
Resortdf$ArrivalMonth <- as.factor(Resortdf$ArrivalMonth)
Resortdf$ArrivalSeason <- as.factor(Resortdf$ArrivalSeason)
Resortdf$ReservationMonth <- as.factor(Resortdf$ReservationMonth)
Resortdf$ReservationSeason <- as.factor(Resortdf$ReservationSeason)

#3 Setup training rows to random 70%
intrain_Citydf <- createDataPartition(y=Citydf$IsCanceled,p=0.7,list=FALSE)
intrain_Resortdf <- createDataPartition(y=Resortdf$IsCanceled,p=0.7,list=FALSE)


#4 SVM of Citydf (Change variables to get optimal Result) (Takes VERY LONG TIME!!!)
sv_Citydf<-svm(IsCanceled~ReservationSeason+LeadTime+StaysInWeekendNights+
        StaysInWeekNights+Adults+Children+Babies+Meal+Country+
        MarketSegment+DistributionChannel+IsRepeatedGuest+
        PreviousCancellations+PreviousBookingsNotCanceled+
        ReservedRoomType+AssignedRoomType+BookingChanges+
        DepositType+DaysInWaitingList+CustomerType+
        RequiredCarParkingSpaces+TotalOfSpecialRequests+
        ReservationMonth+ArrivalMonth,
       data=Citydf, subset=intrain_Citydf, kernel="linear", probability=T)

#5 SVM of Resortdf (Change variables to get optimal Result) (Takes VERY LONG TIME!!!)
sv_Resortdf<-svm(IsCanceled~ReservationSeason+LeadTime+StaysInWeekendNights+
              StaysInWeekNights+Adults+Children+Babies+Meal+Country+
              MarketSegment+DistributionChannel+IsRepeatedGuest+
              PreviousCancellations+PreviousBookingsNotCanceled+
              ReservedRoomType+AssignedRoomType+BookingChanges+
              DepositType+DaysInWaitingList+CustomerType+
              RequiredCarParkingSpaces+TotalOfSpecialRequests+
              ReservationMonth+ArrivalMonth,
             data=Resortdf, subset=intrain_Resortdf, kernel="linear", probability=T)
LeadTime+PreviousCancellations+IsRepeatedGuest+BookingChanges+RequiredCarParkingSpa
ces+TotalOfSpecialRequests+ArrivalMonth

#6 Create testset without NA and that are used in intrain
cleantestset_Citydf <-na.omit(Citydf[-intrain_Citydf])
cleantestset_Resortdf <-na.omit(Resortdf[-intrain_Resortdf])

#7 Predict using svm created
pr_Citydf <- predict(sv_Citydf, cleantestset_Citydf)
pr_Resortdf <- predict(sv_Reosrtdf, cleantestset_Resortdf)
```