Technical Appendix - Model

KaZAM

11/26/2021

Part 1:loading bike activity dataset, station location dataset and weather dataset

```
#loading bike activity
bike_data<-read.csv("/Users/ceciliaxia/Desktop/bike.csv")
#loading location data (containing longitude and latitude)
geo<-read.csv("/Users/ceciliaxia/Desktop/lon.csv")
#loading weather data
weather<-read.csv("/Users/ceciliaxia/Desktop/2771020.csv")
weather<-weather[20081:(20081+364),c(5,10,13)]
colnames(weather)[1]<-"day"
weather$day<-as.Date(weather$day)
```

Part 2: process bike activity dataset and join it with weather dataset

```
dat<-bike_data
# convert the raw data the right format
dat$date<-as.POSIXct(dat$date, format="%Y-%m-%d %H:%M:%S")
dat$day<-as.Date(dat$date)
# add a column indicating which month the ride happened
dat$month<-months(dat$day)
# add a column indicating which day of the week the ride happened
dat$weekday<-weekdays(dat$day)
# add a column indicating whetehr the day the ride happened is weekday
dat$is_weekday<-ifelse(dat$weekday=="Sunday"|dat$weekday=="Saturday",0,1)
dat$station_id<-as.factor(dat$station_id)
# add a column indicating which hour of the day the ride happened
dat$hour<-format(dat$date,"%H")
# join the dat dataset with the location data
j_dat<-inner_join(dat,weather,by="day")
```

Part 3:calculate each stations' variance of availability

```
stations<-split(bike_data,bike_data$station_id)
name<-unique(bike_data$station_id)
#cal_var is a function to calculate variance of availability proportion
cal_var<-function(x){
  dat<-x
  return(variance=var(dat$availability))
}
#calculate variance of availability for each station
var<-data.frame(t(rbind(lapply(stations,cal_var))))
colnames(var)<-c("variance")
var$station_id<-as.numeric(rownames(var))
var$variance<-as.numeric(var$variance)
```

```
# join the variance dataset with location data
var_loc<-left_join(var,geo,by="station_id")
head(var_loc)
```

```
##    variance station_id capacity       lon      lat
## 1  9.046201      31000       15 -77.05323 38.85897
## 2 13.839249      31002       17 -77.04923 38.85643
## 3  6.924178      31003       16 -77.04942 38.86106
## 4  6.702383      31004       12 -77.05949 38.85787
## 5 23.136782      31005       18 -77.05994 38.86230
## 6 11.671840      31006       16 -77.06342 38.86331
```

Part 4: bring in the 20 stations' ids selected by Technical Appendix - Clustering

```
id<-c(31623, 31209, 31233, 31230, 31243, 31205, 31277, 31200, 31101, 31217, 31248, 31272, 31227, 31268,
```

Part 5: add a column indicating whether availability is Yes or No using 20% as the threshold

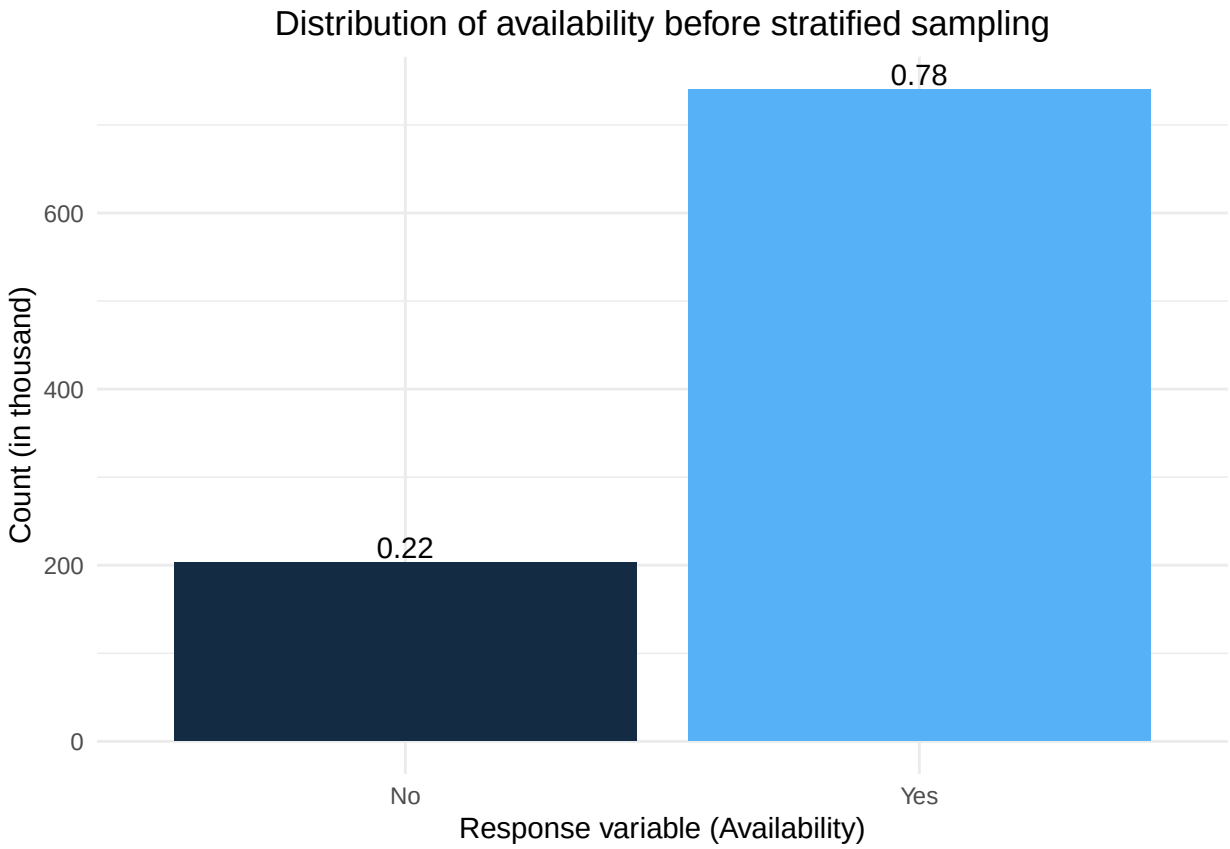
```
# select the data of the 20 stations
j1_dat<-j_dat[which(j_dat$station_id %in% id),]
# add a column indicating whether availability is Yes or No using 20% as the threshold
j1_dat$logi_av<-ifelse(j1_dat$availability_p>0.20,"Yes","No")
head(j1_dat)
```

```
##        station_id      station_name                date is_holiday is_weekend
## 586676      31101 STARTING BIKE NUM 2019-01-01 00:00:01          0          0
## 586677      31101    14th & V St NW 2019-01-01 00:54:28          0          0
## 586678      31101    14th & V St NW 2019-01-01 00:54:40          0          0
## 586679      31101    14th & V St NW 2019-01-01 00:55:20          0          0
## 586680      31101    14th & V St NW 2019-01-01 00:59:34          0          0
## 586681      31101    14th & V St NW 2019-01-01 01:17:38          0          0
##        reshuffle capacity availability availability_p        day   month
## 586676         0       31           31          1.000 2019-01-01 January
## 586677         0       31           31          1.000 2019-01-01 January
## 586678         0       31           31          1.000 2019-01-01 January
## 586679         0       31           30          0.968 2019-01-01 January
## 586680         0       31           29          0.935 2019-01-01 January
## 586681         0       31           30          0.968 2019-01-01 January
##        weekday is_weekday hour PRCP TAVG logi_av
## 586676 Tuesday          1   00    0   56     Yes
## 586677 Tuesday          1   00    0   56     Yes
## 586678 Tuesday          1   00    0   56     Yes
## 586679 Tuesday          1   00    0   56     Yes
## 586680 Tuesday          1   00    0   56     Yes
## 586681 Tuesday          1   01    0   56     Yes
```

Part 6:Check the proportion of Yes and No in column availability in the dataset

```
count<-as.data.frame(table(j1_dat$logi_av))
count$p<-round(count$Freq/sum(count$Freq),2)
count$Freq<-count$Freq/1000
ggplot(count,aes(x=Var1,y=Freq,fill=p)) + geom_bar(stat="identity")+
geom_text(aes(label=p),vjust=-0.2) +
theme_minimal()+
theme(legend.position = "none")+
ylab("Count (in thousand)")+
xlab("Response variable (Availability)")+
```

```
ggtitle("Distribution of availability before stratified sampling")+
theme(plot.title = element_text(hjust = 0.5))
```



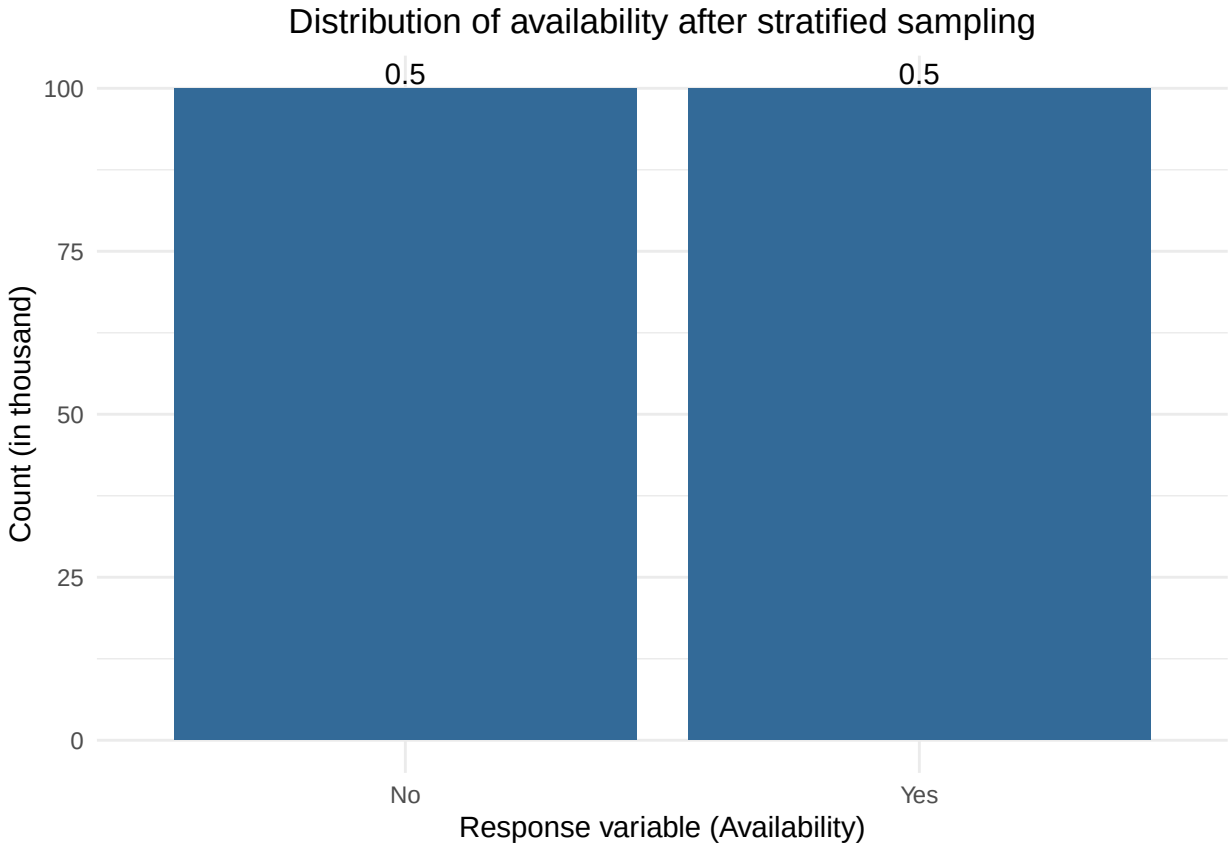
Part 7: do data sampling and make the proportion of Yes and No in column availability equal in the sampled dataset

```
names(j1_dat)
```

```
##  [1] "station_id"     "station_name"   "date"           "is_holiday"
##  [5] "is_weekend"     "reshuffle"      "capacity"       "availability"
##  [9] "availability_p" "day"            "month"          "weekday"
## [13] "is_weekday"     "hour"           "PRCP"           "TAVG"
## [17] "logi_av"
```

```
input1<-j1_dat[,c(1,3,11,13,15:17)]
input1$logi_av<-as.factor(input1$logi_av)
set.seed(10)
# We sampled 100,000 records from both Yes and No
sub_idx<-sampling::strata(input1,stratanames = ("logi_av"),size=rep(100000,2),"srswor")
input<-input1[sub_idx$ID_unit,]
count<-as.data.frame(table(input$logi_av))
count$p<-round(count$Freq/sum(count$Freq),2)
count$Freq<-count$Freq/1000
ggplot(count,aes(x=Var1,y=Freq,fill=p)) + geom_bar(stat="identity")+
geom_text(aes(label=p),vjust=-0.2) +
theme_minimal()+
theme(legend.position = "none")+
ylab("Count (in thousand)")+
```

```
xlab("Response variable (Availability)")+
ggtitle("Distribution of availability after stratified sampling")+
theme(plot.title = element_text(hjust = 0.5))
```



Part 8: divide 24h by 30 minutes and assign each time a 30 minutes category (there are 48 categories in total)

```
#assign each time a 30 minutes category
input$h_m<-format(input$date, format='%H:%M')
mins <- 30 * round(as.double(as.difftime(input$h_m, format = "%H:%M"), "mins") / 30)
input$h_m<-format(as.POSIXct(60 * mins, origin = "1970-01-01", tz = "GMT"), "%H:%M")
input$h_m<-as.factor(input$h_m)
# show hour_minute category
levels(input$h_m)
```

```
##  [1] "00:00" "00:30" "01:00" "01:30" "02:00" "02:30" "03:00" "03:30" "04:00"
## [10] "04:30" "05:00" "05:30" "06:00" "06:30" "07:00" "07:30" "08:00" "08:30"
## [19] "09:00" "09:30" "10:00" "10:30" "11:00" "11:30" "12:00" "12:30" "13:00"
## [28] "13:30" "14:00" "14:30" "15:00" "15:30" "16:00" "16:30" "17:00" "17:30"
## [37] "18:00" "18:30" "19:00" "19:30" "20:00" "20:30" "21:00" "21:30" "22:00"
## [46] "22:30" "23:00" "23:30"
```

```
input$month<-as.factor(input$month)
# show month category
levels(input$month)
```

```
##  [1] "April"     "August"    "December"  "February"  "January"   "July"
##  [7] "June"      "March"     "May"       "November"  "October"   "September"
```

Part 9: make 20 categories for station_id as the original levels doesn't work

```
tmp1<-data.frame(station_id=unique(input$station_id),corres=rep(1:20))
input_f<-left_join(input,tmp1,by="station_id")[,-c(1:2)]
input_f$corres<-as.factor(input_f$corres)
# check whether the levels for the station_id is only 20 now
levels(input_f$corres)
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15"
## [16] "16" "17" "18" "19" "20"
```

```
head(input_f)
```

```
##     month is_weekday PRCP TAVG logi_av   h_m corres
## 1 January          1    0   56     Yes 01:00      1
## 2 January          1    0   56     Yes 02:00      1
## 3 January          1    0   56     Yes 09:00      1
## 4 January          1    0   56     Yes 13:30      1
## 5 January          1    0   56     Yes 14:30      1
## 6 January          1    0   56     Yes 15:00      1
```

Part 10: Split the dataset into 95% training set and 5% test set

```
set.seed(10)
#split the training set and test set
smp_size <- floor(0.95 * nrow(input_f))
train_ind <- sample(nrow(input_f), size = smp_size)
train <- input_f[train_ind, ]
test <- input_f[-train_ind, ]
```

Part 11: fit Random Forest Model to the training set and calculate the accuracy using test set

```
rf1<-randomForest::randomForest(logi_av~.,data=train)
forest.pred <- predict(rf1,test)
# show the importance of each explanatory variable and decide whether to include all the variables
randomForest::importance(rf1)
```

```
##            MeanDecreaseGini
## month              4561.399
## is_weekday         1928.914
## PRCP               3713.970
## TAVG               8273.700
## h_m               19775.554
## corres            22337.858
```

```
# make a confusion matrix about the Actual and the Predicted
table(test$logi_av,forest.pred,dnn = c('Actual','Predicted'))
```

```
##       Predicted
## Actual   No  Yes
##    No  4386  607
##    Yes  843 4164
```

Part 12: fit logistics regression model on the training set and calculate the accuracy using test set

```
logistic_model <- glm(logi_av~.,
                      data = train,
                      family = "binomial")

predict_reg <- predict(logistic_model,
```

```
                        test, type = "response")

predict_reg <- as.factor(ifelse(predict_reg>0.5, "Yes", "No"))
# make a confusion matrix about the Actual and the Predicted
table(test$logi, predict_reg,dnn = c('Actual','Predicted'))
```

```
##       Predicted
## Actual   No  Yes
##    No  3068 1925
##    Yes 1571 3436
```