HW2

Vivian Chu

August 15, 2015

Flights at ABIA

Questions: What day in the week is the worst time to fly? (Most amount of delays)
What month in the year is the worst time to fly

```
library(dplyr)
```

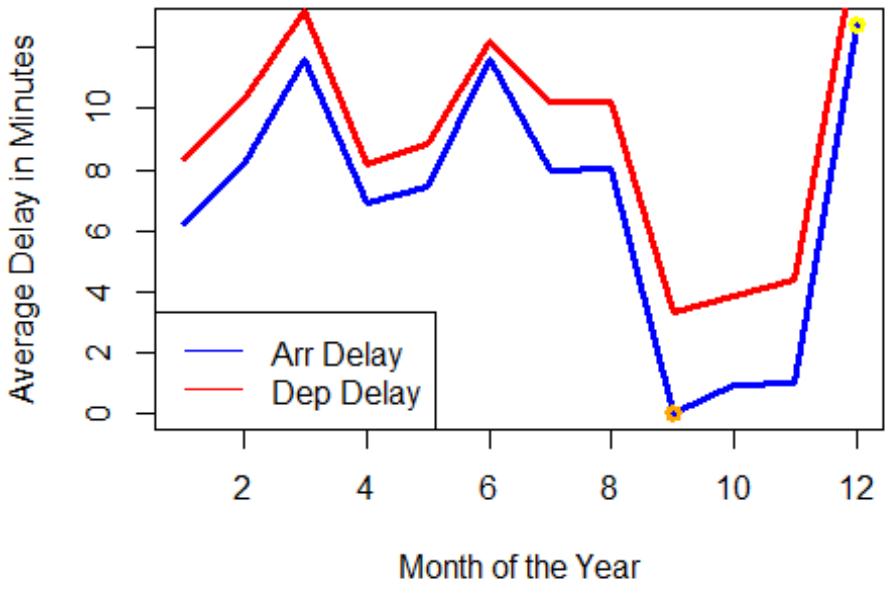
```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
flights =
read.csv('https://raw.githubusercontent.com/jgscott/STA380/master/data/
ABIA.csv')
flight = flights
# turn days of the week and months in a year into factors
df = flight %>%
  mutate(
    day_of_week=factor(DayOfWeek,levels=c(1,2,3,4,5,6,7),
                       labels=
                         c("Monday",
                           "Tuesday",
                           "Wednesday",
                           "Thursday",
                           "Friday",
                           "Saturday",
                           "Sunday"
                         )))
df = df %>%
  mutate(
    Month=factor(Month,levels=c(1,2,3,4,5,6,7,8,9,10,11,12),
                       labels=
                         c("Jan",
                           "Feb",
```

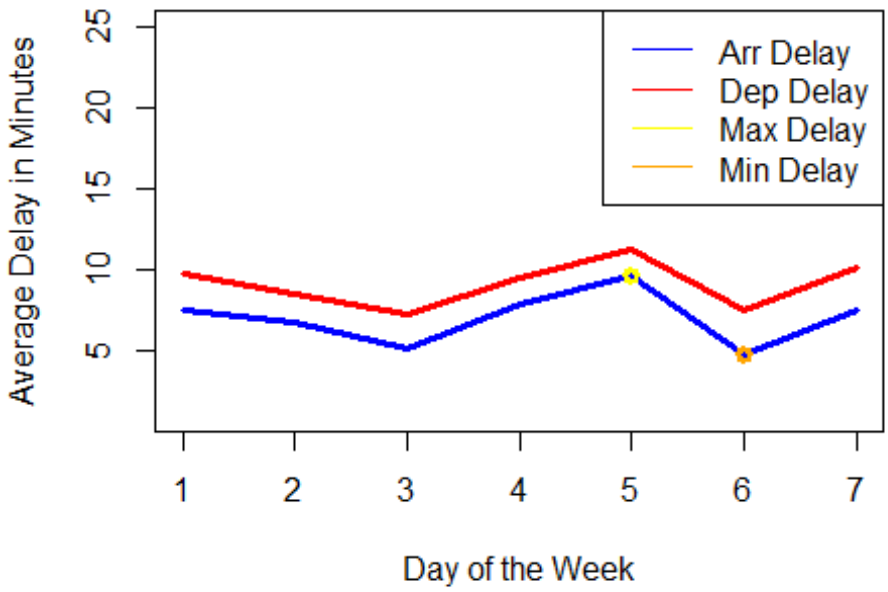
```
                              "Mar",
                              "Apr",
                              "May",
                              "Jun",
                              "Jul",
                              "Aug",
                              "Sept",
                              "Oct",
                              "Nov",
                              "Dec"
                            )))

#create a line chart showing the average delay in minutes across the
months in a year, circling the worst time to fly in yellow and the best
time to fly in orange

aggdelay = aggregate(df$ArrDelay, by=list(df$Month), FUN=mean,
na.rm=TRUE)
plot(aggdelay$x, type = "l", xlab= "Month of the Year", ylab = "Average
Delay in Minutes",col ="blue", lwd = "3")
aggdepdelay = aggregate(df$DepDelay, by=list(df$Month), FUN=mean,
na.rm=TRUE)
lines(aggdepdelay$x, type="l",col ="red", lwd="3")
legend('bottomleft',c("Arr Delay","Dep Delay"),lty=1,
col=c("blue","red"))
points(which.max(aggdelay$x),max(aggdelay$x),lwd=3,col="yellow")
points(which.min(aggdelay$x),min(aggdelay$x),lwd=3,col="orange")
```



```r
#create a line chart showing the average delay in minutes across the
days in the week, circling the worst time to fly in yellow and the best
time to fly in orange
aggdelay2 = aggregate(df$ArrDelay, by=list(df$day_of_week), FUN=mean,
na.rm=TRUE)
aggdepdelay2 = aggregate(df$DepDelay, by=list(df$day_of_week),FUN=mean,
na.rm=TRUE)
plot(aggdelay2$x, type="l", xlab="Day of the Week", ylab="Average Delay
in Minutes", ylim=c(1,25),col="blue", lwd="3")
lines(aggdepdelay2$x, type="l", col="red", lwd="3")
legend('topright',c("Arr Delay","Dep Delay","Max Delay","Min
Delay"),lty=1, col=c("blue","red","yellow","orange"))
points(which.max(aggdelay2$x),max(aggdelay2$x),lwd=3,col="yellow")
points(which.min(aggdelay2$x),min(aggdelay2$x),lwd=3,col="orange")
```



Departure delay follows arrival delays, as expected, since if the plane arrives late, it will depart late, as the passengers can't leave if the plane hasn't come in yet. As expected, December has the most delays in the year as that's when most people take their vacations for Christmas. September has the least delays in the year, presumably because that's when school season starts and neither kids nor parents can get away. Friday has the max delays in the week as people are most likely to travel on Friday for a weekend away.

Busy airports lead to more possibilities of late passengers that have already checked their bags, and the plane cannot leave without the passenger if the bag is already

checked. More planes flying in and out also means more planes queing for the landing field.

Author Attribution

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.2.2
```

```
library(rpart)
library(plyr)
```

```
## ------------------------------------------------------------------------
-----
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr
first, then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------
-----
##
## Attaching package: 'plyr'
##
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(tm)
```

```
## Loading required package: NLP
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.2
```

```
## Loading required package: lattice
## Loading required package: ggplot2
##
## Attaching package: 'ggplot2'
##
```

```
## The following object is masked from 'package:NLP':
##
##     annotate

#Read in the reader's function from tm_examples here
readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),
            id=fname, language='en') }
```

Create training set corpus

```
#Training set
author_dirs.tr = Sys.glob('../data/ReutersC50/C50train/*')
file_list = NULL
labels.tr = NULL
for(author in author_dirs.tr) {
  author_name = substring(author, first=29)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list = append(file_list, files_to_add)
  labels.tr = append(labels.tr, rep(author_name, length(files_to_add)))
}
#Preprocessing
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))

tr.corpus = Corpus(VectorSource(all_docs))
names(tr.corpus) = file_list

tr.corpus = tm_map(tr.corpus, content_transformer(tolower))
tr.corpus = tm_map(tr.corpus, content_transformer(removeNumbers))
tr.corpus = tm_map(tr.corpus, content_transformer(removePunctuation))
tr.corpus = tm_map(tr.corpus, content_transformer(stripWhitespace))
tr.corpus = tm_map(tr.corpus, content_transformer(removeWords),
stopwords("SMART"))

#Forming document term matrix and dense matrix
tr.DTM = DocumentTermMatrix(tr.corpus)
tr.DTM = removeSparseTerms(tr.DTM, 0.975)
```

Create a test set corpus

```
#Test set
author_dirs.ts = Sys.glob('../data/ReutersC50/C50test/*')
file_list = NULL
labels.ts = NULL
for(author in author_dirs.ts) {
  author_name = substring(author, first=28)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list = append(file_list, files_to_add)
  labels.ts = append(labels.ts, rep(author_name, length(files_to_add)))
```

```
}
#Preprocessing
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))

ts.corpus = Corpus(VectorSource(all_docs))
names(ts.corpus) = file_list

ts.corpus = tm_map(ts.corpus, content_transformer(tolower))
ts.corpus = tm_map(ts.corpus, content_transformer(removeNumbers))
ts.corpus = tm_map(ts.corpus, content_transformer(removePunctuation))
ts.corpus = tm_map(ts.corpus, content_transformer(stripWhitespace))
ts.corpus = tm_map(ts.corpus, content_transformer(removeWords),
stopwords("SMART"))

#Extract terms from training set corpus
dict =dimnames(tr.DTM)[[2]]

#Forming document term matrix and dense matrix with only words in the
dictionary from training set

ts.DTM = DocumentTermMatrix(ts.corpus, list(dictionary=dict))
ts.DTM = removeSparseTerms(ts.DTM, 0.975)
```

First selected model: Naive Bayes

```
#Naive Bayes Model
#convert DTM to data frame
tr.df = as.data.frame(inspect(tr.DTM))
ts.df = as.data.frame(inspect(ts.DTM))

NB.mod = naiveBayes(x=tr.df, y=as.factor(labels.tr), laplace =1)
NB.pred = predict(NB.mod, ts.df)
xtab = table(NB.pred,labels.ts)
CMdf = as.data.frame(confusionMatrix(xtab)$byClass)

attach(CMdf)
output = CMdf[order(-Sensitivity),][1]
output

##                          Sensitivity
## Class: LydiaZajc                0.94
## Class: AlanCrosby               0.82
## Class: KouroshKarimkhany        0.80
## Class: PeterHumphrey            0.80
## Class: RogerFillion             0.78
## Class: JimGilchrist             0.76
## Class: AaronPressman            0.60
## Class: LynneO'Donnell           0.50
```

```
## Class: RobinSidel                0.44
## Class: DavidLawder               0.42
## Class: BenjaminKangLim           0.30
## Class: JoWinterbottom            0.28
## Class: TimFarrand                0.24
## Class: MarcelMichelson           0.20
## Class: TheresePoletti            0.20
## Class: JanLopatka                0.18
## Class: BernardHickey             0.16
## Class: NickLouth                 0.12
## Class: MatthewBunce              0.10
## Class: EricAuchard               0.08
## Class: HeatherScoffield          0.08
## Class: LynnleyBrowning           0.08
## Class: FumikoFujisaki            0.06
## Class: KarlPenhaul               0.06
## Class: JohnMastrini              0.04
## Class: MichaelConnor             0.04
## Class: PierreTran                0.04
## Class: GrahamEarnshaw            0.02
## Class: JoeOrtiz                  0.02
## Class: KevinMorrison             0.02
## Class: MarkBendeich              0.02
## Class: SamuelPerry               0.02
## Class: TanEeLyn                  0.02
## Class: ToddNissen                0.02
## Class: AlexanderSmith            0.00
## Class: BradDorfman               0.00
## Class: DarrenSchuettler          0.00
## Class: EdnaFernandes             0.00
## Class: JaneMacartney             0.00
## Class: JonathanBirt              0.00
## Class: KeithWeir                 0.00
## Class: KevinDrawbaugh            0.00
## Class: KirstinRidley             0.00
## Class: MartinWolk                0.00
## Class: MureDickie                0.00
## Class: PatriciaCommins           0.00
## Class: SarahDavison              0.00
## Class: ScottHillis               0.00
## Class: SimonCowell               0.00
## Class: WilliamKazer              0.00
```

```
detach(CMdf)
```

We can see that the Naive Bayes is pretty good at predicting authors Alan Crosby, Lydia Zajc, ...etc. The overall accuracy is:

```
mean(output$Sensitivity)
```

```
## [1] 0.1852
```

The accuracy rate is not that high, only 18%.

Second selected model: Random Forest

```
#Random Forest
#make the columns align in both training and test data set
#add extra columns to the test set with zero values in them
tr.DTM = as.matrix(tr.DTM)
ts.DTM = as.matrix(ts.DTM)

dd = data.frame(ts.DTM[,intersect(colnames(ts.DTM), colnames(tr.DTM))])
ff = read.table(textConnection(""), col.names = colnames(tr.DTM),
colClasses = "integer")
new.ts.DTM = rbind.fill(dd,ff)
ts.df_new = as.data.frame(new.ts.DTM)
tr.df = as.data.frame(tr.DTM)
#run Random Forest
RF.mod = randomForest(tr.df,y=as.factor(labels.tr),ntree=200,mtry=3)
RF.pred = predict(RF.mod, data=new.ts.DTM)
xtab2=table(RF.pred,labels.ts)
CMdf2 = as.data.frame(confusionMatrix(xtab2)$byClass)

#overall accuracy
attach(CMdf2)
output2 = CMdf2[order(-Sensitivity),][1]
mean(output2$Sensitivity)

## [1] 0.698

detach(CMdf2)
```

about 70%!

Conclusion: Random Forest yields a much higher accuracy. Probably because Naive Bayes is so "naive", it assumes every variable is independent when it is in fact not. Word "BBQ" is probably highly associated with "Texas."

Practice w/ Association Rule Mining

```
library(arules)  # has a big ecosystem of packages built around it

## Warning: package 'arules' was built under R version 3.2.2

## Loading required package: Matrix
##
## Attaching package: 'arules'
##
## The following object is masked from 'package:tm':
##
##     inspect
##
```

```
## The following objects are masked from 'package:base':
##
##     %in%, write

# Read  in grocery file
groceries =
read.transactions("https://raw.githubusercontent.com/jgscott/STA380/mas
ter/data/groceries.txt", format = 'basket', sep = ',')

#Apply apriori to find items that occur frequently
groceriesrules = apriori(groceries, parameter=list(support=.01,
confidence=.5, maxlen=5))

##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen
maxlen
##         0.5    0.1    1 none FALSE            TRUE    0.01      1
5
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## apriori - find association rules with the apriori algorithm
## version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [15 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

# Look at the output
inspect(groceriesrules)

##    lhs                     rhs                   support confidence
lift
## 1  {curd,
##     yogurt}             => {whole milk}       0.01006609  0.5823529
2.279125
## 2  {butter,
##     other vegetables}   => {whole milk}       0.01148958  0.5736041
2.244885
## 3  {domestic eggs,
##     other vegetables}   => {whole milk}       0.01230300  0.5525114
2.162336
## 4  {whipped/sour cream,
```

```
##     yogurt}             => {whole milk}       0.01087951  0.5245098
2.052747
## 5  {other vegetables,
##     whipped/sour cream} => {whole milk}       0.01464159  0.5070423
1.984385
## 6  {other vegetables,
##     pip fruit}          => {whole milk}       0.01352313  0.5175097
2.025351
## 7  {citrus fruit,
##     root vegetables}    => {other vegetables} 0.01037112  0.5862069
3.029608
## 8  {root vegetables,
##     tropical fruit}     => {other vegetables} 0.01230300  0.5845411
3.020999
## 9  {root vegetables,
##     tropical fruit}     => {whole milk}       0.01199797  0.5700483
2.230969
## 10 {tropical fruit,
##     yogurt}             => {whole milk}       0.01514997  0.5173611
2.024770
## 11 {root vegetables,
##     yogurt}             => {other vegetables} 0.01291307  0.5000000
2.584078
## 12 {root vegetables,
##     yogurt}             => {whole milk}       0.01453991  0.5629921
2.203354
## 13 {rolls/buns,
##     root vegetables}    => {other vegetables} 0.01220132  0.5020921
2.594890
## 14 {rolls/buns,
##     root vegetables}    => {whole milk}       0.01270971  0.5230126
2.046888
## 15 {other vegetables,
##     yogurt}             => {whole milk}       0.02226741  0.5128806
2.007235
```

Choose a few subsets to inspect

```
#choose highest value of lift,3: will show the subsets with the highest
occurance in the itemsets
inspect(subset(groceriesrules, subset= lift>3))
```

```
##   lhs                   rhs                    support confidence
lift
## 1 {citrus fruit,
##    root vegetables} => {other vegetables} 0.01037112  0.5862069
3.029608
## 2 {root vegetables,
##    tropical fruit}  => {other vegetables} 0.01230300  0.5845411
3.020999
```

```
#choose highest value for confidence, 58%
inspect(subset(groceriesrules, subset=confidence>0.58))
```

```
##   lhs                   rhs                    support confidence
lift
## 1 {curd,
##    yogurt}          => {whole milk}       0.01006609  0.5823529
2.279125
## 2 {citrus fruit,
##    root vegetables} => {other vegetables} 0.01037112  0.5862069
3.029608
## 3 {root vegetables,
##    tropical fruit}  => {other vegetables} 0.01230300  0.5845411
3.020999
```

```
#choose highest values for both confidence and support (0.012) for the
highest exclusivity
inspect(subset(groceriesrules,subset=support>0.012 & confidence>0.58))
```

```
##   lhs                   rhs                  support confidence
lift
## 1 {root vegetables,
##    tropical fruit}  => {other vegetables} 0.012303  0.5845411
3.020999
```

The higher the support and confidence, the less items show up as it becomes more exclusive. We see that people who tend to buy root vegetables and tropical fruit will likely buy other vegetables. Why? I believe it's because people who don't eat vegetables to begin with won't bother with the exotic fruits and vegetables, and the people who eat these exotic fruits and vegetables must really like vegetables, and hence they will buy other kinds of vegetables.