36-462 Final Project: Classification

Santiago Roa

Contents


```
price = read.csv("price.csv")
review = read.csv("review.csv")
price.t = read.csv("price_test.csv")
review.t = read.csv("review_test.csv")
```

```
library(stringr)
library(geosphere)
library(e1071)
```

Data Pre-Processing

```
review.pre = review
review.pre$host_is_superhost = as.factor(as.numeric(review.pre$host_is_superhost) -1)
review.pre$host_identity_verified = as.factor(as.numeric(review.pre$host_identity_verified) -1)
review.pre$instant_bookable = as.factor(as.numeric(review.pre$instant_bookable) -1)
review.pre$host_response_rate = as.numeric(sub("%", "", review.pre$host_response_rate)) / 100
review.pre$host_response_time = as.factor(unclass(review.pre$host_response_time) -1)
review.pre$cleaning_fee = as.numeric(sub("$", "", review.pre$cleaning_fee, fixed = T))
review.pre$price = as.numeric(gsub("$", "", review.pre$price, fixed = T))
```

```
## Warning: NAs introduced by coercion
```

```
review.pre$cancellation_policy = as.factor(as.numeric(review.pre$cancellation_policy))
review.pre$review_scores_rating = as.factor(as.numeric(review.pre$review_scores_rating))

review.post = review.t
review.post$host_is_superhost = as.factor(as.numeric(review.post$host_is_superhost) -1)
review.post$host_identity_verified = as.factor(as.numeric(review.post$host_identity_verified) -1)
review.post$instant_bookable = as.factor(as.numeric(review.post$instant_bookable) -1)
review.post$host_response_rate = as.numeric(sub("%", "", review.post$host_response_rate)) / 100
review.post$host_response_time = as.factor(unclass(review.post$host_response_time) -1)
review.post$cleaning_fee = as.numeric(sub("$", "", review.post$cleaning_fee, fixed = T))
review.post$price = as.numeric(gsub("$", "", review.post$price, fixed = T))
```

```
## Warning: NAs introduced by coercion
```

```
review.post$cancellation_policy = as.factor(as.numeric(review.post$cancellation_policy))
# review.post$review_scores_rating = as.factor(as.numeric(review.post$review_scores_rating))
```

Amenities

```
# library: stringr
amenities = sapply(review.pre$amenities, FUN=function(x) {
  # some function that just splits up all the words in each row of amenities
  s = toString(x)
  s = strsplit(s[[1]],",")
  newS = str_replace(s[[1]],'\\{',"")
  newS = str_replace(newS,'\\}',"")
  newS = str_replace(newS,'\"',"")
  newS = str_replace(newS,'\"',"")
  tolower(newS)
})

amenitiesPost = sapply(review.post$amenities, FUN=function(x) {
  # some function that just splits up all the words in each row of amenities
  s = toString(x)
  s = strsplit(s[[1]],",")
  newS = str_replace(s[[1]],'\\{',"")
  newS = str_replace(newS,'\\}',"")
  newS = str_replace(newS,'\"',"")
  newS = str_replace(newS,'\"',"")
  tolower(newS)
})

num.amenities = c()
for(i in 1:length(review.pre$amenities)) {
  num.amenities[i] = length(amenities[[i]])
}

num.amenities.post = c()
for(i in 1:length(review.post$amenities)) {
  num.amenities.post[i] = length(amenitiesPost[[i]])
}

review.post$num_amenities = num.amenities.post
review.pre$num_amenities = num.amenities
```

Top Amenities

```
listOfAmenities = c()
for (i in c(1:length(review.pre$amenities))) {
  temp = amenities[[i]]
  listOfAmenities = c(listOfAmenities,temp)
}
t = sort(table(listOfAmenities),decreasing = TRUE)
```

```
# t[1:30]
```

Locations

```
# library: geosphere
pike.place.lat = 47.6097 ; pike.place.long = -122.3422
space.needle.lat = 47.6205 ; space.needle.long = -122.3493
downtown.lat = 47.6050 ; downtown.long = -122.3344
gum.wall.lat = 47.6084 ; gum.wall.long = -122.3404
great.wheel.lat = 47.6062 ; great.wheel.long = -122.3425

pike.dist = c()
space.dist = c()
downtown.dist = c()
gum.dist = c()
wheel.dist = c()
for(i in 1:length(review.pre$latitude)) {
  pike.dist[i] = distm(c(pike.place.long, pike.place.lat), c(review.pre$longitude[i], review.pre$latitu
  space.dist[i] = distm(c(space.needle.long, space.needle.lat), c(review.pre$longitude[i], review.pre$la
  downtown.dist[i] = distm(c(downtown.long, downtown.lat), c(review.pre$longitude[i], review.pre$latitu
  gum.dist[i] = distm(c(gum.wall.long, gum.wall.lat), c(review.pre$longitude[i], review.pre$latitude[i]
  wheel.dist[i] = distm(c(great.wheel.long, great.wheel.lat), c(review.pre$longitude[i], review.pre$lati
}

review.pre$pike_dist = pike.dist ; review.pre$space_dist = space.dist
review.pre$downtown_dist = downtown.dist ; review.pre$gum_dist = gum.dist
review.pre$wheel_dist = wheel.dist

###

pike.dist.post = c()
space.dist.post = c()
downtown.dist.post = c()
gum.dist.post = c()
wheel.dist.post = c()
for(i in 1:length(review.post$latitude)) {
  pike.dist.post[i] = distm(c(pike.place.long, pike.place.lat), c(review.post$longitude[i], review.post$
                            fun = distGeo) / 1609.344
  space.dist.post[i] = distm(c(space.needle.long, space.needle.lat), c(review.post$longitude[i], review
                             fun = distGeo) / 1609.344
  downtown.dist.post[i] = distm(c(downtown.long, downtown.lat), c(review.post$longitude[i], review.post$
                                fun = distGeo) / 1609.344
  gum.dist.post[i] = distm(c(gum.wall.long, gum.wall.lat), c(review.post$longitude[i], review.post$latit
                           fun = distGeo) / 1609.344
  wheel.dist.post[i] = distm(c(great.wheel.long, great.wheel.lat), c(review.post$longitude[i], review.po
                             fun = distGeo) / 1609.344
}

review.post$pike_dist = pike.dist.post ; review.post$space_dist = space.dist.post
review.post$downtown_dist = downtown.dist.post ; review.post$gum_dist = gum.dist.post
review.post$wheel_dist = wheel.dist.post
```

Sample Train/Test Split: 70/30

```
nR <- nrow(review.pre)
review.price <- review.pre[sample(nR), ]
train.indicesR <- 1:round(0.7 * nR)
trainR <- review.price[train.indicesR,]
test.indicesR <- (round(0.7 * nR) + 1):nR
testR <- review.price[test.indicesR,]
trainR <- trainR[-c(9,17,19)]
testR <- testR[-c(9,17,19)]

review.pre <- review.pre[-c(9,18,19)]
review.post <- review.post[-c(9,18,19)]
# data[!(data$property_type %in% whatever), ]
```

Classification

Logistic Regression

```
logRegMdl = glm(review_scores_rating ~., data = trainR, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
logReg.results <- predict(logRegMdl,newdata=testR,type='response')
logReg.results <- ifelse(logReg.results > 0.5,1,0)
nas = which(is.na(logReg.results))
misClasificError <- mean(logReg.results[-nas] != testR$review_scores_rating[-nas])
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.866999168744805"
```

```
###


# logRegMdl = glm(review_scores_rating ~., data = review.pre, family = "binomial")

# logReg.results <- predict(logRegMdl,newdata=review.pre,type='response')
# logReg.results <- ifelse(logReg.results > 0.5,1,0)
# nas = which(is.na(logReg.results))
# misClasificError <- mean(logReg.results[-nas] != review.pre$review_scores_rating[-nas])
# print(paste('Accuracy',1-misClasificError))
```

Random Forest

```
library(randomForest)
rand.fit = randomForest(review_scores_rating~., data = trainR, ntree = 500, na.action=na.exclude)
rand.predictions = predict(rand.fit, newdata = testR)
nas = which(is.na(rand.predictions))
misClasificError <- mean(rand.predictions[-nas] != testR$review_scores_rating[-nas])
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.892768079800499"
```

```
#
# plot(rand.fit)
# varImpPlot(rand.fit,
#            sort = T,
#            n.var=10,
#            main="Top 10 - Variable Importance")
#
# r$preds = predict(rand.fit, r)
# print(confusionMatrix(data = r$preds,
#                       reference = r$review_scores_rating,
#                       positive = '0'))
```

SVM

```
# tune.svm(review_scores_rating~.,data=trainR)
svmfitLinear <-svm(review_scores_rating~., data=trainR,kernel='linear')
misClasificError <- mean(predict(svmfitLinear, newdata=testR)!=testR$review_scores_rating)
```

```
## Warning in `!=.default`(predict(svmfitLinear, newdata = testR),
## testR$review_scores_rating): longer object length is not a multiple of
## shorter object length
```

```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.841714756801319"
```

```
svmfitPoly <-svm(review_scores_rating~., data=trainR,kernel='polynomial')
misClasificError <- mean(predict(svmfitPoly, newdata=testR)!=testR$review_scores_rating)
```

```
## Warning in `!=.default`(predict(svmfitPoly, newdata = testR),
## testR$review_scores_rating): longer object length is not a multiple of
## shorter object length
```

```
## Warning in `!=.default`(predict(svmfitPoly, newdata = testR),
## testR$review_scores_rating): longer object length is not a multiple of
## shorter object length
```

```
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.842539159109645"
```

```
svmfitRadial <-svm(review_scores_rating~., data=trainR,kernel='radial')
misClasificError <- mean(predict(svmfitRadial, newdata=testR)!=testR$review_scores_rating)
```

```
## Warning in `!=.default`(predict(svmfitRadial, newdata = testR),
## testR$review_scores_rating): longer object length is not a multiple of
## shorter object length
```

```
## Warning in `!=.default`(predict(svmfitRadial, newdata = testR),
## testR$review_scores_rating): longer object length is not a multiple of
## shorter object length
```

```r
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.827699917559769"
```

```r
# tune.out.linear = tune(svm, review_scores_rating~., data = trainR, kernel ='linear', ranges =list(cos
#                                                                                     gamma =c(0.01, 0.05, .1, .5, 1
# tune.out.linear$best.model

# tune.out.poly = tune(svm, review_scores_rating~., data = trainR, kernel ='polynomial', ranges =list(c
#                                                                                     gamma =c(0.01, 0.05, .1, .5, 1
# tune.out.poly$best.model

# tune.out.radial = tune(svm, review_scores_rating~., data = trainR, kernel ='radial', ranges =list(cos
#                                                                                     gamma =c(0.01, 0.05, .1, .5, 1
# tune.out.radial$best.model
```

Trees

```r
library(rpart)
tree = rpart(review_scores_rating~., data = trainR)
# plotcp(tree)
pruned.tree = prune(tree, cp = 0.019)
# plot(pruned.tree)
tree.predictions = predict(tree, testR)
pruned.tree.predictions = predict(pruned.tree, testR)

tree.predictions = ifelse(tree.predictions[,1] > 0.5,0, 1)
pruned.tree.predictions = ifelse(pruned.tree.predictions[,1] > 0.5,0, 1)


misClasificError <- mean(tree.predictions != testR$review_scores_rating)
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.877988458367683"
```

```r
misClasificError <- mean(pruned.tree.predictions != testR$review_scores_rating)
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.877988458367683"
```

LDA

```r
# library(MASS)
# lda.object <-lda(trainR[-nas-24], grouping = trainR[-nas,24])
# names(lda.object)
# ## [1] "prior"   "counts"  "means"   "scaling" "lev"     "svd"     "N"## [8] "call"
# Z = x.014.tr%*%lda.object$scalingdim(Z)
```