```r
###############
# Parametric Analyses
# Author: Ravi Makhija
# Version 1.5
#
# Description:
# Parametric tests conducted on review data, including t-tests and logistic
# regression.
#
# File Dependencies:
#   'data/tripadvisor_data.Rdata'
#   'data/yelp_data.Rdata'
#
# How to run:
#    Source this script (no need to set wd beforehand if directory structure is
#    maintained as downloaded).
#
# References
#   1) Set working directory to the file path of a script:
#      http://stackoverflow.com/questions/13672720/r-command-for-setting-working-dire
#  2) Assumptions for a t-test:
#     https://statistics.laerd.com/spss-tutorials/independent-t-test-using-spss-stati

require(data.table)

## Loading required package:  data.table
## Warning:  package 'data.table' was built under R version 3.1.3

require(bit64)

## Loading required package:  bit64
## Warning:  package 'bit64' was built under R version 3.1.3
## Loading required package:  bit
## Warning:  package 'bit' was built under R version 3.1.3
## Attaching package bit
## package:bit (c) 2008-2012 Jens Oehlschlaegel (GPL-2)
## creators:  bit bitwhich
## coercion:  as.logical as.integer as.bit as.bitwhich which
## operator:  !  & | xor != ==
## querying:  print length any all min max range sum summary
```

```
## bit access:  length<- [ [<- [[ [[<-
## for more help type ?bit
##
## Attaching package:  'bit'
##
## The following object is masked from 'package:data.table':
##
##    setattr
##
## The following object is masked from 'package:base':
##
##    xor
##
## Attaching package bit64
## package:bit64 (c) 2011-2012 Jens Oehlschlaegel (GPL-2 with commercial
restrictions)
## creators:  integer64 seq :
## coercion:  as.integer64 as.vector as.logical as.integer as.double
as.character as.bin
## logical operator:  !  & | xor != == < <= >= >
## arithmetic operator:  + - * / %/% %% ^
## math:  sign abs sqrt log log2 log10
## math:  floor ceiling trunc round
## querying:  is.integer64 is.vector [is.atomic} [length] is.na
format print
## aggregation:  any all min max range sum prod
## cumulation:  diff cummin cummax cumsum cumprod
## access:  length<- [ [<- [[ [[<-
## combine:  c rep cbind rbind as.data.frame
## for more help type ?bit64
##
## Attaching package:  'bit64'
##
## The following object is masked from 'package:bit':
##
##    still.identical
##
## The following objects are masked from 'package:base':
##
```

```
##    %in%, :, is.double, match, order, rank

library(pROC)

## Warning:  package 'pROC' was built under R version 3.1.3
## Type 'citation("pROC")' for a citation.
##
## Attaching package:  'pROC'
##
## The following objects are masked from 'package:stats':
##
##    cov, smooth, var

################
# Load in data
################

load("tripadvisor_data.Rdata")
load("yelp_data.Rdata")

################
# t-tests
################

# We begin by conducting t-tests for our hypotheses.

################ HYPOTHESIS 1
# We first hypothesize that there is a difference in the mean user rating
# for local and non-local restaurant goers. We found reasons why local
# may be higher, and why non-local may be higher, and therefore it was
# unclear whether these affects would cancel, or which would in fact be
# higher. Therefore, we use two-sided hypothesis tests. We consider the Yelp
# data as a sample of a larger population of restaurant goers in DC. And
# likewise, the TripAdvisor is a sample of the larger population of
# restaurant goers. We conduct t-tests for Yelp and TripAdvisor separately
# here, to avoid additional complexities and biases that may result from
# combining data from two websites.

# Before proceeding, we consider any t-test assumptions:
```

```r
# 1) dependent variable on continuous scale
# VIOLATED
# 2) independent variable is two categories, independent groups
# OK
# 3) independence of observations
# OK
# 4) no significant outliers
# OK
# 5) dependent variable approximately normally distributed in each category
#   VIOLATED
# 6) homogeneity of variances
#   N/A, since we are using a Welch Two Sample t-test.

# Assumption 1 was violated since these user ratings are on an ordinal scale
# of 1 to 5. However, it may be argued that this was only due to the website
# not allowing more fine grained ratings on a continuous scale, and that in
# fact the underlying scale is continuous. We adopt this approach, and more
# broadly justify our use of the mean rating with this approach.

# Assumption 5 was violated by default since we are again on an ordinal scale.
# However, if we imagine fillig in the continuous scale ratings, the data
# appear closer to a normal distribution. However, there does seem to be a
# left skew, e.g. ratings of 4 or 5 are generally more popular than than lower
# ratings (which is a nice sign of an optimistic society perhaps!) With this in
# mind, we relax this assumption and proceed with our t-tests.

# ------------------------
# Beginning with Yelp data.
# H_0: The mean user rating for local and non-local reviewers is the same.
# H_a: The mean user rating for local and non-local reviewers is not the same.

# Check how many user reviews are local/non-local. We notice there is a
# class imbalance, which should not affect the t-test, but comes into play
# later in the logistic regression.

print(table(yelp_data$user_is_local))

##
##  FALSE   TRUE
```

```
##  92552 162174

# A cursory look at the mean for local and non_local ratings.
# The mean for local yelp reviews is 3.723254, while the mean for non-local
# yelp reviews is 3.819291.

print(yelp_data[ , mean(user_rating), by=user_is_local])

##    user_is_local       V1
## 1:          TRUE 3.723254
## 2:         FALSE 3.819291

# We use a Welch Two Sample t-test, which handles the case of unequal variances.

# With a p-value of 2.2e-16, we can see that there is indeed a statistically
# significant difference between the local and non-local yelp user ratings,
# at the .05 significance level. We can also see this reflected in the
# confidence interval for the difference in ratings, which does not include 0.
# The test suggests the mean yelp rating for non-local reviews is higher than
# for local reviews.

yelp_ttest <- t.test(yelp_data[user_is_local == 1]$user_rating,
                     yelp_data[user_is_local == 0]$user_rating,
                     var.equal = FALSE)
print(yelp_ttest)

##
##  Welch Two Sample t-test
##
## data:  yelp_data[user_is_local == 1]$user_rating and yelp_data[user_is_local == 0]
## t = -20.5063, df = 199645.3, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.10521584 -0.08685765
## sample estimates:
## mean of x mean of y
##  3.723254  3.819291

# ------------------------
# Again, for TripAdvisor data:
```

```r
# H_0: The mean user rating for local and non-local reviewers is the same.
# H_a: The mean user rating for local and non-local reviewers is not the same.

# Check how many user reviews are local/non-local.

print(table(tripadvisor_data$user_is_local))

##
## FALSE  TRUE
## 72570 28261

# A cursory look at the mean for local and non_local ratings.
# The mean for local TripAdvisor reviews is 4.031421, while the mean for
# non-local TripAdvisor reviews is 4.181742. A small difference.

print(tripadvisor_data[ , mean(user_rating), by=user_is_local])

##    user_is_local       V1
## 1:         FALSE 4.181742
## 2:          TRUE 4.031421

# Now, we conduct a t-test as we did for Yelp data.

# With a p-value of 2.2e-16, we can see that there is indeed a statistically
# significant difference between the local and non-local yelp user ratings,
# at the .05 significance level. We can also see this reflected in the
# confidence interval for the difference in ratings, which does not include 0.
# The data suggests the mean TripAdvisor rating for non-local reviews is higher
# than for local reviews, as was the case for Yelp reviews. Though, we
# acknowledge that the difference is very small here, and therefore the
# practical significance is questionable.

tripadvisor_ttest <- t.test(tripadvisor_data[user_is_local == TRUE]$user_rating,
                            tripadvisor_data[user_is_local == FALSE]$user_rating,
                            var.equal = FALSE)
print(tripadvisor_ttest)

##
##  Welch Two Sample t-test
##
```

```
## data:  tripadvisor_data[user_is_local == TRUE]$user_rating and tripadvisor_data[us
## t = -21.0658, df = 47872.43, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1643065 -0.1363342
## sample estimates:
## mean of x mean of y
##  4.031421  4.181742

############### HYPOTHESIS 2
# We additionally hypothesize that the overall user rating is not different
# between the two websites. We again use a t-test to examine this difference.
# The assumptions from the first hypothesis still apply here.
# H_0: The mean user rating on Yelp and TripAdvisor is the same.
# H_a: The mean user rating on Yelp and TripAdvisor is not the same.

# We see that there is a statistically significant difference. The test suggests
# that the mean rating on TripAdvisor is higher. This suggests that it is
# a good idea to treat the data from the two websites separately, unless
# special care is given to accounting for the biases when combining.

inter_website_ttest <- t.test(yelp_data$user_rating,
                              tripadvisor_data$user_rating,
                              var.equal = FALSE)
print(inter_website_ttest)

##
##  Welch Two Sample t-test
##
## data:  yelp_data$user_rating and tripadvisor_data$user_rating
## t = -99.2043, df = 215060.2, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.3889984 -0.3739253
## sample estimates:
## mean of x mean of y
##  3.758148  4.139610

############### HYPOTHESIS 3
# Next, we try a logistic regression model, to try and predict user_is_local
```

```
# using the other variables in the data set. We fit two models, one for each
# website. We hypothesize that user ratings in particular should have some
# predictive power when predicting whether a user is local or non-local. And,
# that other variables in our data set may also contribute some predictive
# power.

################
# Logistic Regression
################

# Note on assumptions for logistic regression:
# We assume that logistic regression is a reasonable model here, e.g. that
# there is a linear relationship between the log odds of a local review,
# and the feature we use below. We also see that our sample size is large, so
# that the sampling distributions of the coefficient estimators are
# approximately normally distributed, so that we can draw inferences using
# hypothesis testing on the coefficients.

# ------------------------
# Yelp

# First we try to predict user_is_local with user_rating. Indeed, we can see
# that user_rating is statistically significant in our model at the .01
# significance level.

yelp_logit_1 <- glm(user_is_local ~ user_rating, data = yelp_data, family = "binomial
print(summary(yelp_logit_1))
```

```
##
## Call:
## glm(formula = user_is_local ~ user_rating, family = "binomial",
##     data = yelp_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5141  -1.3828   0.9288   0.9567   0.9850
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.837032   0.014309   58.50   <2e-16 ***
```

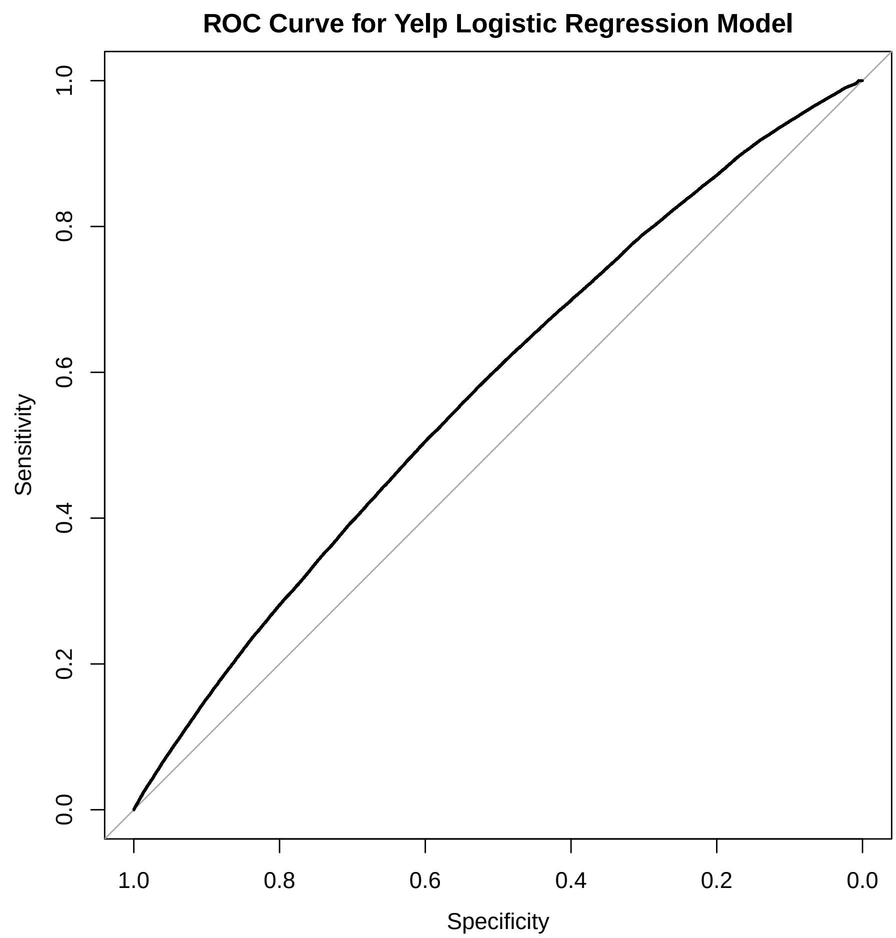
```
## user_rating -0.073209   0.003619  -20.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 333852  on 254725  degrees of freedom
## Residual deviance: 333439  on 254724  degrees of freedom
## AIC: 333443
##
## Number of Fisher Scoring iterations: 4
```

```
# Next, we add in the user's number of reviews. This is also statistically
# significant at the .01 level.

yelp_logit_2 <- glm(user_is_local ~ user_rating + user_num_reviews, data = yelp_data,
print(summary(yelp_logit_2))
```

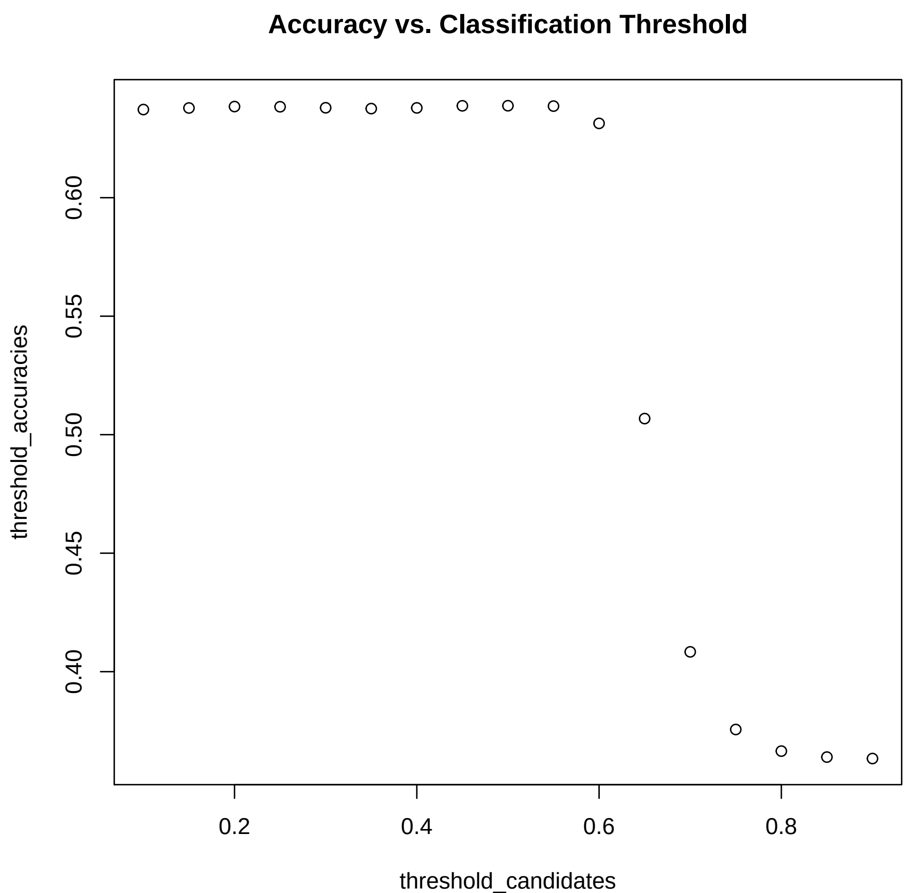
```
##
## Call:
## glm(formula = user_is_local ~ user_rating + user_num_reviews,
##     family = "binomial", data = yelp_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5627  -1.4104   0.9019   0.9517   2.0798
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       0.9500878  0.0146650   64.79   <2e-16 ***
## user_rating      -0.0777212  0.0036510  -21.29   <2e-16 ***
## user_num_reviews -0.0007238  0.0000166  -43.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 333852  on 254725  degrees of freedom
## Residual deviance: 331357  on 254723  degrees of freedom
## AIC: 331363
```

```
##
## Number of Fisher Scoring iterations: 4

# Finally, we add the user's review length. Indeed, all three predictors in the
# model are statistically significant.

yelp_logit_3 <- glm(user_is_local ~ user_rating + user_num_reviews + user_review_leng
print(summary(yelp_logit_3))

##
## Call:
## glm(formula = user_is_local ~ user_rating + user_num_reviews +
##     user_review_length, family = "binomial", data = yelp_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0887  -1.3794   0.8856   0.9631   2.2945
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          7.089e-01  1.588e-02   44.63   <2e-16 ***
## user_rating         -6.094e-02  3.684e-03  -16.54   <2e-16 ***
## user_num_reviews    -8.566e-04  1.741e-05  -49.20   <2e-16 ***
## user_review_length  2.850e-04  7.342e-06   38.83   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 333852  on 254725  degrees of freedom
## Residual deviance: 329758  on 254722  degrees of freedom
## AIC: 329766
##
## Number of Fisher Scoring iterations: 4

# Next, we look at a ROC curve. The ROC curve slopes above the diagonal line
# which represents random guessing, suggesting that our model is better than
# guessing at random. However, it also suggests that there is plenty of room
# for improvement, from a prediction standpoint.
```

```r
yelp_prob=predict(yelp_logit_3,type=c("response"))
yelp_data$prob=yelp_prob
# Since the ROC function is time consuming, results were saved from earlier,
# and just loaded here.
#yelp_roc <- roc(user_is_local ~ prob, data = yelp_data)
#save(yelp_roc, file = "yelp_roc.Rdata")
load("yelp_roc.Rdata")
plot(yelp_roc,
     main = "ROC Curve for Yelp Logistic Regression Model")
```



```
##
## Call:
## roc.formula(formula = user_is_local ~ prob, data = yelp_data)
##
## Data: prob in 92552 controls (user_is_local FALSE) < 162174 cases (user_is_local T
## Area under the curve: 0.5754

# So, we adpot our logistic regression model yelp_logit_3, which has three
# predictors: user_rating + user_num_reviews + user_review_length

# In this case, we are interested in the accuracy of predicting whether a user
# is local or not. E.g. we don't necessarily want to prioritize one or the
# other, and therefore do not have a need for tweaking the false positive rate
# threshold. Therefore, we want to maximize the accuracy. We can see this
# is the case when the threshold is 0.5. Though, lower thresholds also provide
# fair accuracy. This is likely due to the class imbalance, e.g. there are
# almost twice as many local users than non-local users in this data set.

threshold_candidates <- seq(.1, .9, by=.05)
threshold_accuracies <- numeric()
for (thresh in threshold_candidates <- seq(.1, .9, by=.05)) {
  threshold_accuracies <- c(threshold_accuracies,
                            sum(yelp_data$user_is_local == (yelp_data$prob > thresh))
}
plot(threshold_candidates,
     threshold_accuracies,
     main = "Accuracy vs. Classification Threshold")
```




```r
# Therefore, we use a classification threshold of 0.5.

# Next, we create a confusion matrix. Inspecting the results, it becomes more
# clear that there is a serious issue with our model. Namely, while the true
# positive rate is very good (about .98), the rate of false negatives is also
# very high at 0.9654. It seems then that the class imbalance of local and
# non_local is dominating here, and our model therefore may not be that great
# for prediction after all, despite observing statistical significance.

p <- nrow(yelp_data[user_is_local == TRUE])
tp <- sum(yelp_data[user_is_local == TRUE]$prob > .5)
fp <- p - tp
```

```r
n <- nrow(yelp_data[user_is_local == FALSE])
tn <- sum(yelp_data[user_is_local == FALSE]$prob <= .5)
fn <- n - tn

yelp_confusion_matrix <- matrix(c(tp/p, 1 - tp/p, 1-tn/n, tn/n), nrow=2)
colnames(yelp_confusion_matrix) <- c("local_observed", "non_local_observed")
rownames(yelp_confusion_matrix) <- c("local_predicted", "non_local_predicted")

print(yelp_confusion_matrix)
```

```
##                     local_observed non_local_observed
## local_predicted          0.9835917         0.96537082
## non_local_predicted      0.0164083         0.03462918
```

```r
# Finally, we can use 5-fold cross-validation to better assess the
# classification accuracy, sticking with a threshold of 0.5. This gives a
# classification rate of 0.6386705. But again, we acknowledge that the large
# class imbalance is likely dictating these results.

set.seed(1)
num_folds <- 5
n <- nrow(yelp_data)
fold_n <- floor(n/num_folds)
yelp_data_shuffled <- copy(yelp_data)[sample(1:n), ]
yelp_cv_results <- numeric(num_folds)

for (i in 1:num_folds) {
  if (i != num_folds) {
    fold_start_index <- (i-1)*fold_n + 1
    fold_end_index <- i*fold_n
    current_lm <- glm(user_is_local ~ user_rating + user_num_reviews + user_review_le
    current_prob=predict(current_lm,type=c("response"), newdata=yelp_data[fold_start_
    current_accuracy <- sum(yelp_data[fold_start_index:fold_end_index]$user_is_local
    yelp_cv_results[i] <- current_accuracy
  }
  else {
    fold_start_index <- (i-1)*fold_n + 1
    fold_end_index <- n
    current_lm <- glm(user_is_local ~ user_rating + user_num_reviews + user_review_le
    current_prob=predict(current_lm,type=c("response"), newdata=yelp_data[fold_start_
```

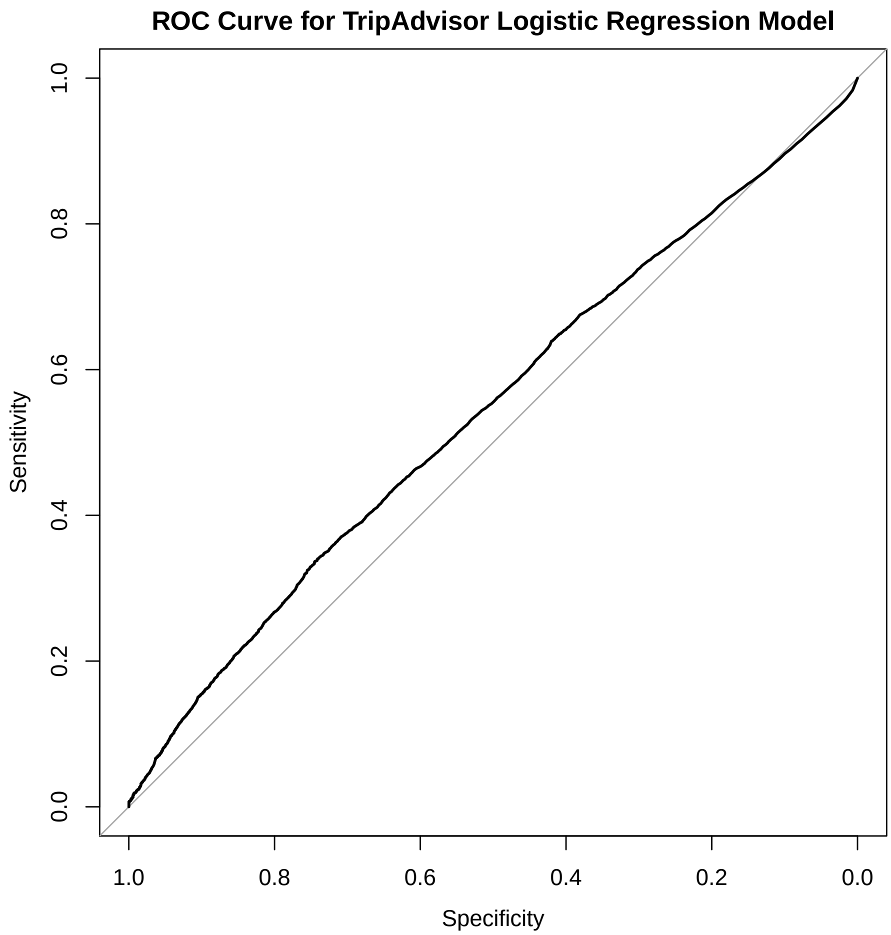
```
    current_accuracy <- sum(yelp_data[fold_start_index:fold_end_index]$user_is_local
    yelp_cv_results[i] <- current_accuracy
  }
}

# Taking the average of the classification rate for each iteration:
yelp_cv_classification_rate <- mean(yelp_cv_results)
print(yelp_cv_classification_rate)

## [1] 0.6386705

# In conclusion, for the yelp data it seems that the class imbalance is
# dictating the results of our model selection. In particular, the proportion
# of the sample that is made up of local reviewers is 0.6366606. Which means,
# if one were to guess local every time, they would have a classification
# accuracy of 0.6366606. On the other hand, the cross-validated classification
# accuracy for the logistic regression model is 0.6386705. On one hand, this
# difference is very small. On the other hand, perhaps the increase in
# classification rate may be attributed to there being some predictive power
# in our features.

# ------------------------
# TripAdvisor

# We proceed similarly for TripAdvisor. However, since some of the user reviews
# are abridged, we do not consider review length for TripAdvisor.

# First we try to predict user_is_local with user_rating. Indeed, we can see
# that user_rating is statistically significant in our model at the .01
# significance level.

tripadvisor_logit_1 <- glm(user_is_local ~ user_rating, data = tripadvisor_data, fami
print(summary(tripadvisor_logit_1))

##
## Call:
## glm(formula = user_is_local ~ user_rating, family = "binomial",
##     data = tripadvisor_data)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -0.9834  -0.8167  -0.7656   1.5197   1.6555
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.324514   0.029119  -11.14   <2e-16 ***
## user_rating -0.150543   0.006927  -21.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 119630  on 100830  degrees of freedom
## Residual deviance: 119165  on 100829  degrees of freedom
## AIC: 119169
##
## Number of Fisher Scoring iterations: 4
```

```
# Next, we add in the user's number of reviews. This is also statistically
# significant at the .01 level.

tripadvisor_logit_2 <- glm(user_is_local ~ user_rating + user_num_reviews, data = tri
print(summary(tripadvisor_logit_2))
```

```
##
## Call:
## glm(formula = user_is_local ~ user_rating + user_num_reviews,
##     family = "binomial", data = tripadvisor_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3024  -0.8122  -0.7603   1.4746   1.6865
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -4.195e-01  2.966e-02  -14.14   <2e-16 ***
## user_rating      -1.455e-01  6.963e-03  -20.89   <2e-16 ***
## user_num_reviews  8.955e-04  4.775e-05   18.75   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 119630  on 100830  degrees of freedom
## Residual deviance: 118779  on 100828  degrees of freedom
## AIC: 118785
##
## Number of Fisher Scoring iterations: 4

# Using a ROC curve, we see that for TripAdvisor, the ROC curve shows that
# our model is better than random for most thresholds, with the exception being
# at high values for specificity. But again, the AUC is not great, and suggests
# only that our features here add some predictive power, but are far from
# predicting well.

tripadvisor_prob=predict(tripadvisor_logit_2,type=c("response"))
tripadvisor_data$prob=tripadvisor_prob
tripadvisor_roc <- roc(user_is_local ~ prob, data = tripadvisor_data)
plot(tripadvisor_roc,
     main = "ROC Curve for TripAdvisor Logistic Regression Model")
```




```
##
## Call:
## roc.formula(formula = user_is_local ~ prob, data = tripadvisor_data)
##
## Data: prob in 72570 controls (user_is_local FALSE) < 28261 cases (user_is_local TR
## Area under the curve: 0.5425

# Next, we look at a confusion matrix. This time, we see that the TNR is great,
# while the TPR is quite horrific. This, again, seems do be due to a class
# imbalance. For TripAdvisor, the class imbalanace is in favor of non_local
# users, which is why the confusion matrix this time favors FPR over TPR,
# due to the way local vs. non-local is coded.
```

```r
p <- nrow(tripadvisor_data[user_is_local == TRUE])
tp <- sum(tripadvisor_data[user_is_local == TRUE]$prob > .5)
fp <- p - tp
n <- nrow(tripadvisor_data[user_is_local == FALSE])
tn <- sum(tripadvisor_data[user_is_local == FALSE]$prob <= .5)
fn <- n - tn

tripadvisor_confusion_matrix <- matrix(c(tp/p, 1 - tp/p, 1-tn/n, tn/n), nrow=2)
colnames(tripadvisor_confusion_matrix) <- c("local_observed", "non_local_observed")
rownames(tripadvisor_confusion_matrix) <- c("local_predicted", "non_local_predicted")

print(tripadvisor_confusion_matrix)

##                     local_observed non_local_observed
## local_predicted        0.007466119        0.001088604
## non_local_predicted    0.992533881        0.998911396

# Here, a quick look at the class imbalance. The proportion of reviews that
# are from non-local reviewers is 0.7197191. This, of course, makes sense as
# well, given that TripAdvisor seems to be geared towards the person planning
# a trip.

table(tripadvisor_data$user_is_local)

##
## FALSE  TRUE
## 72570 28261

# We finish by again using cross-validation to come up with a prediction
# accuracy statistic for our the TripAdvisor model.

set.seed(1)
num_folds <- 5
n <- nrow(tripadvisor_data)
fold_n <- floor(n/num_folds)
tripadvisor_data_shuffled <- copy(tripadvisor_data)[sample(1:n), ]
tripadvisor_cv_results <- numeric(num_folds)

for (i in 1:num_folds) {
  if (i != num_folds) {
```

```
    fold_start_index <- (i-1)*fold_n + 1
    fold_end_index <- i*fold_n
    current_lm <- glm(user_is_local ~ user_rating + user_num_reviews + user_review_le
    current_prob=predict(current_lm,type=c("response"), newdata=tripadvisor_data[fold
    current_accuracy <- sum(tripadvisor_data[fold_start_index:fold_end_index]$user_is
    tripadvisor_cv_results[i] <- current_accuracy
  }
  else {
    fold_start_index <- (i-1)*fold_n + 1
    fold_end_index <- n
    current_lm <- glm(user_is_local ~ user_rating + user_num_reviews + user_review_le
    current_prob=predict(current_lm,type=c("response"), newdata=tripadvisor_data[fold
    current_accuracy <- sum(tripadvisor_data[fold_start_index:fold_end_index]$user_is
    tripadvisor_cv_results[i] <- current_accuracy
  }
}

# Taking the average of the classification rate for each iteration:
tripadvisor_cv_classification_rate <- mean(tripadvisor_cv_results)
print(tripadvisor_cv_classification_rate)

## [1] 0.7210582

# We get a classification rate of 0.7210582. This is slightly better than what
# we would get from predicting every review as non_local, which would be
# an accuracy of 0.7197191. However, the difference is very small, so the
# practical significance is questionable. However, it does suggest that perhaps
# these minute differences in the features used to at least offer some
# predictive power, in predicting whether a review is local or non_local.
```