Problem Set 6

MGSC 310, Fall 2019, Professor Hersh (BEST PROFESSOR EVER!!!)

Geoffrey Hughes

10/11/2019

Question 1) What Predicts Movie Blockbusters?

a. Clean and modify the data; create train / test datasets

```
library('tidyverse')
```

```
## -- Attaching packages -----------------------------------------------------------------------

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts --------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
options(scipen = 50)
set.seed(1861)
movies <- read.csv("/Users/geoffreyhughes/Documents/MGSC_310/MGSC310/Datasets/movie_metadata.csv")

movies <- movies %>% filter(budget < 400000000) %>%
filter(content_rating != "",
       content_rating != "Not Rated",
       !is.na(gross))

movies <- movies %>%
  mutate(genre_main = unlist(map(strsplit(as.character(movies$genres),"\\|"),1)),
         grossM = gross / 1000000,
         budgetM = budget / 1000000,
         profitM = grossM - budgetM,
         blockbuster = ifelse(grossM > 200,1,0))

movies <- movies %>% mutate(genre_main = fct_lump(genre_main,5),
                            content_rating = fct_lump(content_rating,3),
                            country = fct_lump(country,2),
                            cast_total_facebook_likes000s =
                              cast_total_facebook_likes / 1000,) %>%
drop_na()
top_director <- movies %>%
  group_by(director_name) %>%
  summarize(num_films = n()) %>%
```

```r
  top_frac(.1) %>%
  mutate(top_director = 1) %>%
  select(-num_films)
```

```
## Selecting by num_films
```

```r
movies <- movies %>%
  left_join(top_director, by = "director_name") %>%
  mutate(top_director = replace_na(top_director,0))

train_idx <- sample(1:nrow(movies),size = floor(0.75*nrow(movies)))
movies_train <- movies %>% slice(train_idx)
movies_test <- movies %>% slice(-train_idx)
```

b.

```r
#movies_train$blockbuster
train_bb_mean <- mean(movies_train$blockbuster)
test_bb_mean <- mean(movies_test$blockbuster)

t_test <- t.test(movies_train$blockbuster, movies_test$blockbuster)
t_test
```

```
##
##  Welch Two Sample t-test
##
## data:  movies_train$blockbuster and movies_test$blockbuster
## t = -2.4067, df = 1370, p-value = 0.01623
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.037626956 -0.003832732
## sample estimates:
##  mean of x  mean of y
## 0.03935599 0.06008584
```

- We get a p-value of 0.01623
- This p-value, given a resonable alpha of 0.05, is less than the level of significance (alpha). That means that we reject the null hypothesis, which means **the difference in means is statistically significant.**

c. Logistic Model for blockbuster variable

```r
mod1 <- glm(blockbuster ~ budgetM + top_director + cast_total_facebook_likes000s + content_rating + genr
           data = movies_train,
           family = "binomial")

preds_train <- data.frame(
  scores_mod1 = predict(mod1, type = "response"),
  class_pred05 = ifelse(predict(mod1,
                                type = "response") > 0.5, 1, 0),
```

```r
  movies_train
)

preds_test <- data.frame(
  scores_mod1 = predict(mod1, type = "response"),
  class_pred05 = ifelse(predict(mod1,
                                type = "response") > 0.5, 1, 0),
  movies_train
)


summary(mod1)
```

```
##
## Call:
## glm(formula = blockbuster ~ budgetM + top_director + cast_total_facebook_likes000s +
##     content_rating + genre_main, family = "binomial", data = movies_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3617  -0.1909  -0.1111  -0.0534   3.5660
##
## Coefficients:
##                                 Estimate Std. Error z value
## (Intercept)                    -4.784644   0.415845 -11.506
## budgetM                         0.023097   0.002158  10.702
## top_director                    0.607554   0.248837   2.442
## cast_total_facebook_likes000s   0.006694   0.002772   2.415
## content_ratingPG-13            -0.184195   0.309130  -0.596
## content_ratingR                -1.918355   0.526983  -3.640
## content_ratingOther             0.402269   0.504138   0.798
## genre_mainAdventure             0.419475   0.331818   1.264
## genre_mainComedy               -0.458585   0.452521  -1.013
## genre_mainCrime               -14.592267 734.472604  -0.020
## genre_mainDrama                -0.482782   0.519183  -0.930
## genre_mainOther                -0.087916   0.527822  -0.167
##                                           Pr(>|z|)
## (Intercept)                   < 0.0000000000000002 ***
## budgetM                       < 0.0000000000000002 ***
## top_director                              0.014623 *
## cast_total_facebook_likes000s             0.015734 *
## content_ratingPG-13                       0.551275
## content_ratingR                           0.000272 ***
## content_ratingOther                       0.424909
## genre_mainAdventure                       0.206168
## genre_mainComedy                          0.310869
## genre_mainCrime                           0.984149
## genre_mainDrama                           0.352429
## genre_mainOther                           0.867713
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 927.34  on 2794  degrees of freedom
## Residual deviance: 555.53  on 2783  degrees of freedom
## AIC: 579.53
##
## Number of Fisher Scoring iterations: 18
```

```
exp(mod1$coefficients)
```

```
##                 (Intercept)                     budgetM
##             0.0083571006069             1.0233660631999
##                top_director cast_total_facebook_likes000s
##             1.8359347119416             1.0067168814577
##          content_ratingPG-13             content_ratingR
##             0.8317734653623             0.1468483415090
##          content_ratingOther          genre_mainAdventure
##             1.4952131157085             1.5211634571618
##             genre_mainComedy             genre_mainCrime
##             0.6321778570518             0.0000004598951
##              genre_mainDrama             genre_mainOther
##             0.6170641061037             0.9158377819063
```

d. Interpret coefficients: content_ratingR, genre_mainAdventure, and top_director

- content_ratingR, genre_mainAdventure, and top_director have coefficients, -1.918355, 0.419475, and 0.607554, respectively. To find meaning from these, we simply do exp(mod1$coefficients) and subtract 1 from those new values. After that we have -0.8531516585 for content_ratingR, 0.5211634571618 for genre_mainAdventure, and 0.8359347119416 for top_director.

These translate to: * Movies rated R have 85.3152% less of a chance of being a blockbuster compared to movies rated G. * Movies with the genre of Adventure have a 52.1163% greater chance of being a blockbuster than an action movie. * Movies with a top director have 83.5934% greater of a chance of being a blockbuster, when compared to movies without a top director.

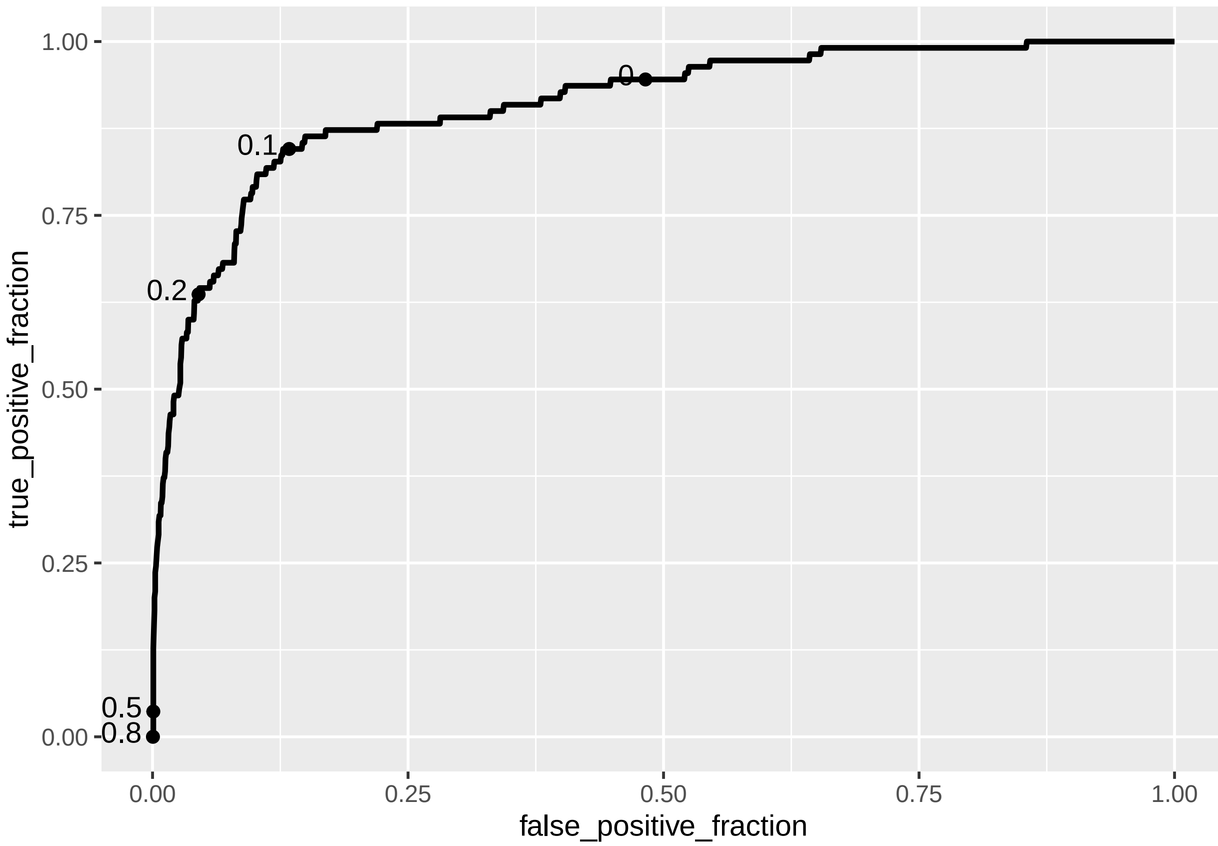
e & f. Use Leave-One-Out Cross Validation and store Predictions for train, test

```
preds_LOOCV_store <- rep(NA, nrow(movies_train))
preds_LOOCV_store <- nrow(movies_train)

num_rows <- nrow(movies_train)

for(i in 1:num_rows)
{
  mod2 <- glm(blockbuster ~ budgetM + top_director + cast_total_facebook_likes000s + content_rating + g
            data = movies_train)

  preds_LOOCV_store[i] <- predict(mod2, newdata = movies_train %>% slice(i))
}

preds_LOOCV <- data.frame(
  scores_mod2 = preds_LOOCV_store,
  class_pred05 = ifelse(predict(mod2,
                                type = "response") > 0.5, 1, 0),
```

```r
  movies_train
)

preds_LOOCV_test_store <- predict(mod2, newdata = movies_test)

preds_test <- predict(mod1, newdata = movies_test)
```

g. Plot the ROC curves for the test predictions, in-sample training predictions, and the LOOCV predictions

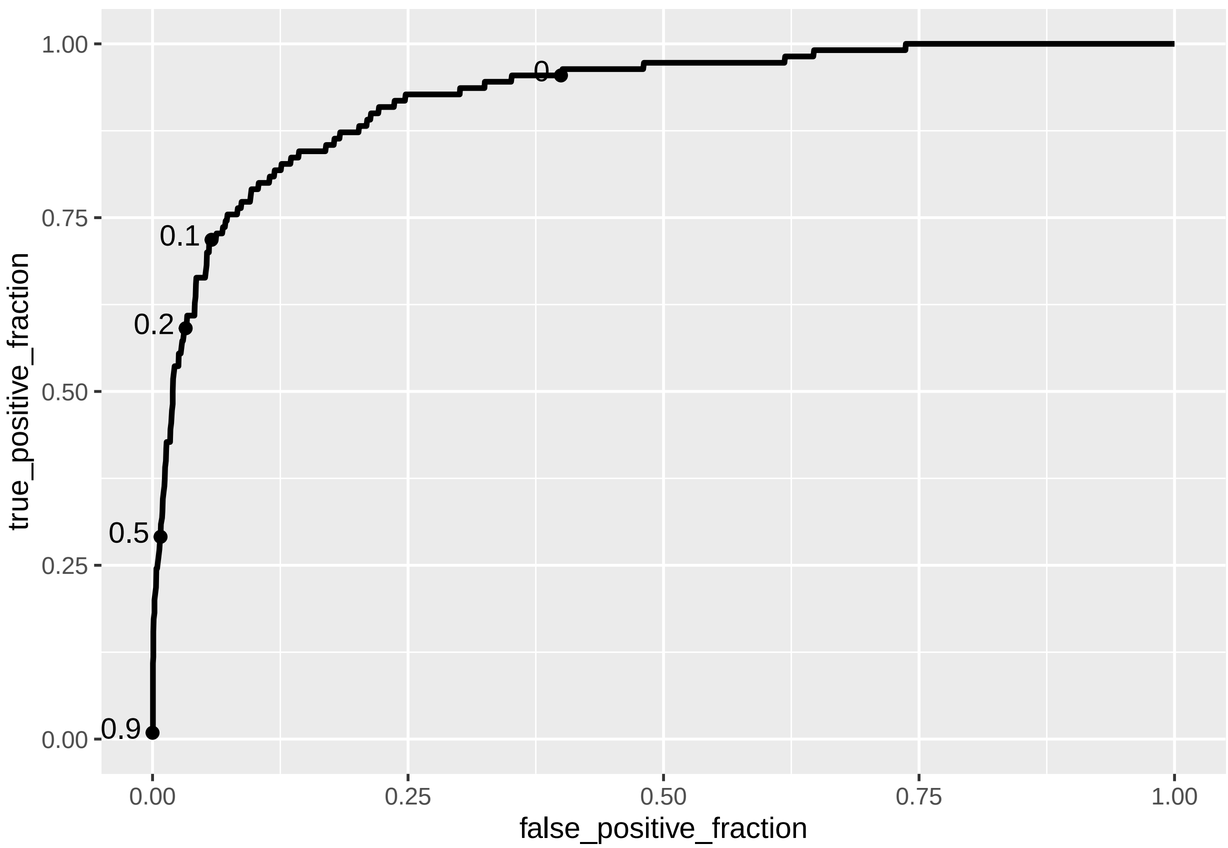
```r
library(plotROC)

p_LOOCV_train <- ggplot(preds_LOOCV, aes(m = scores_mod2,
                       d = movies_train$blockbuster)) +
  geom_roc(cutoffs.at = c(.99, 0.5, 0.2, 0.1, 0.01))

p_LOOCV_train
```




```r
calc_auc(p_LOOCV_train)
```

```
##   PANEL group       AUC
## 1     1    -1 0.9101608
```

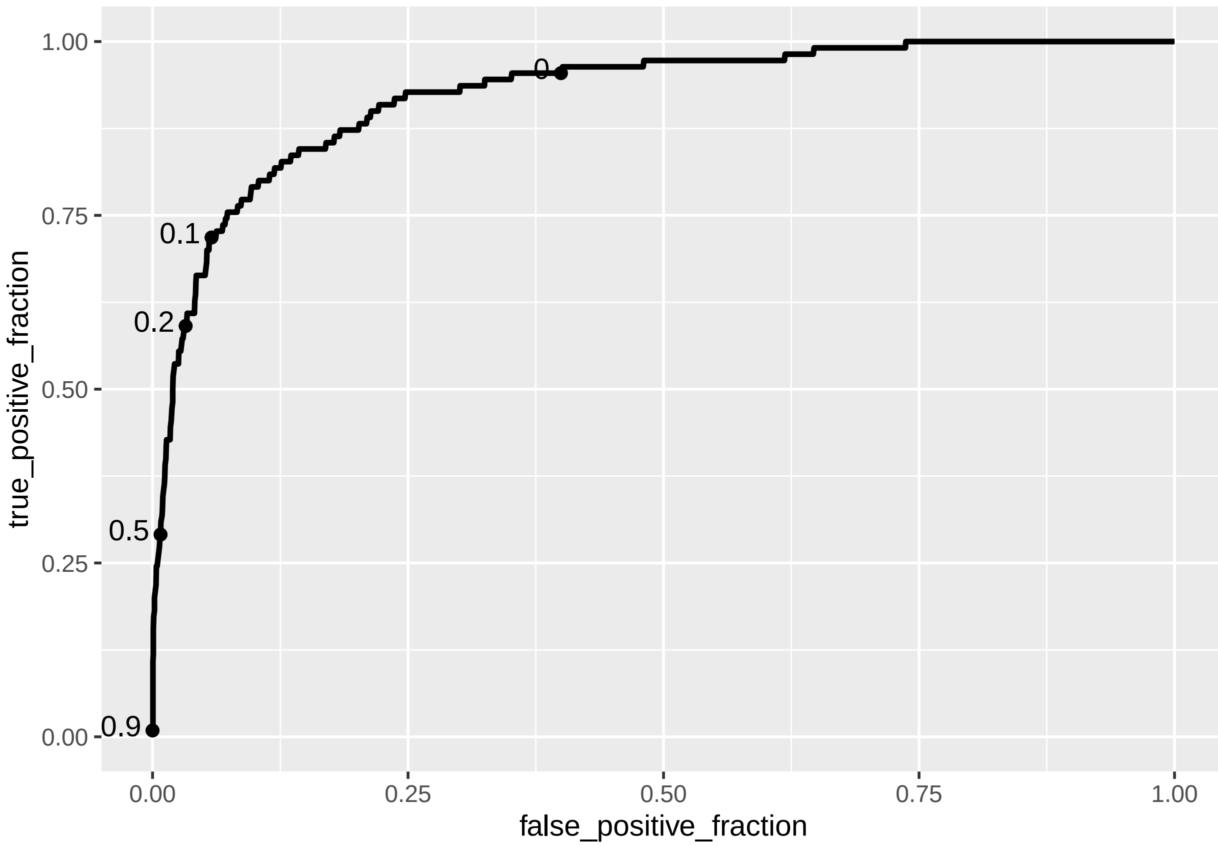
```r
p_train <- ggplot(preds_train, aes(m = scores_mod1,
                    d = movies_train$blockbuster)) +
  geom_roc(cutoffs.at = c(.99, 0.5, 0.2, 0.1, 0.01))

p_train
```




```r
calc_auc(p_train)
```

```
##   PANEL group       AUC
## 1     1    -1 0.9239733
```

```r
p_test <- ggplot(preds_train, aes(m = scores_mod1,
                    d = movies_train$blockbuster)) +
  geom_roc(cutoffs.at = c(.99, 0.5, 0.2, 0.1, 0.01))

p_test
```




```r
calc_auc(p_test)
```

```
##   PANEL group       AUC
## 1     1    -1 0.9239733
```

- The two ROC curves that use the first model (not LOOCV) are more gradual and smoothe. But the LOOCV one has performance spikes, and is less smooth, but overall has a huge jump in TPF right before cutoff = 0.1 to over 0.75. Whereas the others are gradual, and are not even at 0.75 when cutoff = 0.1.

h. AUC values, how do they relate to one another?

```r
calc_auc(p_train)
```

```
##   PANEL group       AUC
## 1     1    -1 0.9239733
```

```r
calc_auc(p_test)
```

```
##   PANEL group       AUC
## 1     1    -1 0.9239733
```

```r
calc_auc(p_LOOCV_train)
```

```
##   PANEL group       AUC
## 1     1    -1 0.9101608
```

- Our LOOCV has a slightly lower AUC, but that is only because the curve is much less smooth. It start out worse, but it has a huge jump up to a high FPR later on (right before cutoff - 0.1). This can explain why it has less AUC compared to the non-LOOCV, which hold a very smooth progression. Also the train glm does better than the test glm probably because it is fit to the training data.

i. Downsample and Upsample the data sets

```r
library("ROSE")
```

```
## Loaded ROSE 0.0-3
```

```r
# Downsampling
down_data <- ROSE(blockbuster ~ budgetM + top_director + cast_total_facebook_likes000s + content_rating
                  movies_train,
                  N = 220,
                  p = 1/2)

# Upsampling
up_data <- ROSE(blockbuster ~ budgetM + top_director + cast_total_facebook_likes000s + content_rating +
                movies_train,
                N = 5000,
                p = 1/2)
```