Problem Set 7

Jadyn Gonzalez

10/18/2019

What Predicts Blockbuster Movies?

We'll be working with the movies dataset again so we need to set our directory and import.

```
setwd("~/Documents/MGSC310")
movies = read.csv("movie_metadata.csv")
```

- a) We need to run the given code to clean the data and generate test and train sets.

```
options(scipen = 50)
# removing missing values
movies = movies[complete.cases(movies),]

# removing empty content rating or not rated
movies = movies[(movies$content_rating != "" & movies$content_rating != "Not Rated"), ]

# removing movies with budget > 400M
movies = movies[movies$budget < 400000000,]

# creating budget, gross, and profit columns in millions
movies$grossM = movies$gross/1e+6
movies$budgetM = movies$budget/1e+6
movies$profitM = movies$grossM - movies$budgetM

# creating a column for main genre
movies$genre_main = do.call('rbind',strsplit(as.character(movies$genres), '|', fixed=TRUE))[,1]
```

```
## Warning in rbind(c("Action", "Adventure", "Fantasy", "Sci-Fi"),
## c("Action", : number of columns of result is not a multiple of vector
## length (arg 2)
```

```
# creating a dummy for blockbuster movies
movies$blockbuster = ifelse(movies$grossM > 200, 1, 0)
library(forcats)
movies$genre_main = fct_lump(movies$genre_main,5)
movies$content_rating = fct_lump(movies$content_rating,3)
movies$country = fct_lump(movies$country,2)
movies$cast_total_facebook_likes000s = movies$cast_total_facebook_likes / 1000

# top director
director_props = data.frame(prop.table(table(movies$director_name)))
directors_indx = order(director_props$Freq,decreasing = TRUE)
top_directors_indx = directors_indx[1:floor(0.1*nrow(director_props))]
top_directors_names = director_props[top_directors_indx, 1]
movies$top_director = ifelse(movies$director_name %in% top_directors_names, 1, 0)
```

```
# train/test split
set.seed(1861)
train_idx = sample(1:nrow(movies),size = floor(0.75*nrow(movies)))
movies_train = movies[train_idx,]
movies_test = movies[-train_idx,]
```

- b) Lets display the mean for blockbuster movies in our test and train sets.

```
mean(movies_train$blockbuster)
```

```
## [1] 0.03935599
```

```
mean(movies_test$blockbuster)
```

```
## [1] 0.06008584
```

- c) Now we'll run a log model of blockbuster against several variables and output the summary as well as the exponentiated coefficients.

```
mod1 = glm(blockbuster ~ budgetM + top_director + cast_total_facebook_likes000s
           + content_rating + genre_main, data = movies_train, family = binomial)
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
## -2.3373  -0.1923  -0.1099  -0.0538   3.5763
##
## Coefficients:
##                                Estimate Std. Error z value
## (Intercept)                   -4.818430   0.420363 -11.463
## budgetM                        0.022993   0.002165  10.622
## top_director                   0.595282   0.266037   2.238
## cast_total_facebook_likes000s  0.006687   0.002775   2.410
## content_ratingPG-13           -0.196937   0.310937  -0.633
## content_ratingR               -1.920051   0.526607  -3.646
## content_ratingOther            0.390799   0.502856   0.777
## genre_mainAdventure            0.434639   0.331964   1.309
## genre_mainComedy              -0.442454   0.451648  -0.980
## genre_mainCrime              -14.558908 737.031912  -0.020
## genre_mainDrama               -0.461686   0.518009  -0.891
## genre_mainOther               -0.065099   0.527714  -0.123
##                                          Pr(>|z|)
## (Intercept)                   < 0.0000000000000002 ***
## budgetM                       < 0.0000000000000002 ***
## top_director                              0.025248 *
```

```
## cast_total_facebook_likes000s           0.015971 *
## content_ratingPG-13                      0.526494
## content_ratingR                          0.000266 ***
## content_ratingOther                      0.437065
## genre_mainAdventure                      0.190434
## genre_mainComedy                         0.327262
## genre_mainCrime                          0.984240
## genre_mainDrama                          0.372785
## genre_mainOther                          0.901821
## ---
## 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: 556.41  on 2783  degrees of freedom
## AIC: 580.41
##
## Number of Fisher Scoring iterations: 18
```

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

```
##                (Intercept)                       budgetM
##            0.0080794633498               1.0232589268520
##               top_director cast_total_facebook_likes000s
##            1.8135424882354               1.0067094885772
##        content_ratingPG-13               content_ratingR
##            0.8212421783181               0.1465995112896
##        content_ratingOther           genre_mainAdventure
##            1.4781614419006               1.5444052754299
##           genre_mainComedy               genre_mainCrime
##            0.6424576441243               0.0000004754959
##            genre_mainDrama               genre_mainOther
##            0.6302204930683               0.9369742427844
```

- d) The p-value of content_ratingR tells us that it is statistically significant. The exponent of the coefficient is 0.147 which means that compared to a G rated movie (content_ratingG is used as the reference variable) a movie with a rating of R is expected to be (.147-1) 85.3% less likely to be a blockbuster.

Let's list the genres before we interpret adventure.

```
summary(movies$genre_main)
```

```
##    Action Adventure    Comedy     Crime     Drama     Other
##       953       367       980       251       661       515
```

We see that Action movies are the reference for our model. The p-value is high for Adventure movies and is statistically insignificant. If we take the exponent of the coefficient we get 1.54 which means that compared to Action movies, Adventure movies are expected to be 54% more likely to be a blockbuster

The p value for top_director tells us that the variable is statistically significant. When we take the exponent of the coefficient we get 1.81 which means that having a top director increases the probability that a movie is a blockbuster by (1.81-1) 81%.

- e) Lets generate predictions for our data sets.

```
predsTrain = data.frame(
  movies_train,
  scores = predict(mod1, type = "response")
)

predsTest = data.frame(
  movies_test,
  scores = predict(glm(blockbuster ~ budgetM + top_director + cast_total_facebook_likes000s
                       + content_rating + genre_main, data = movies_test, family = binomial), type = "re
)
```

- f) Now we'll run LOOCV on our train set.

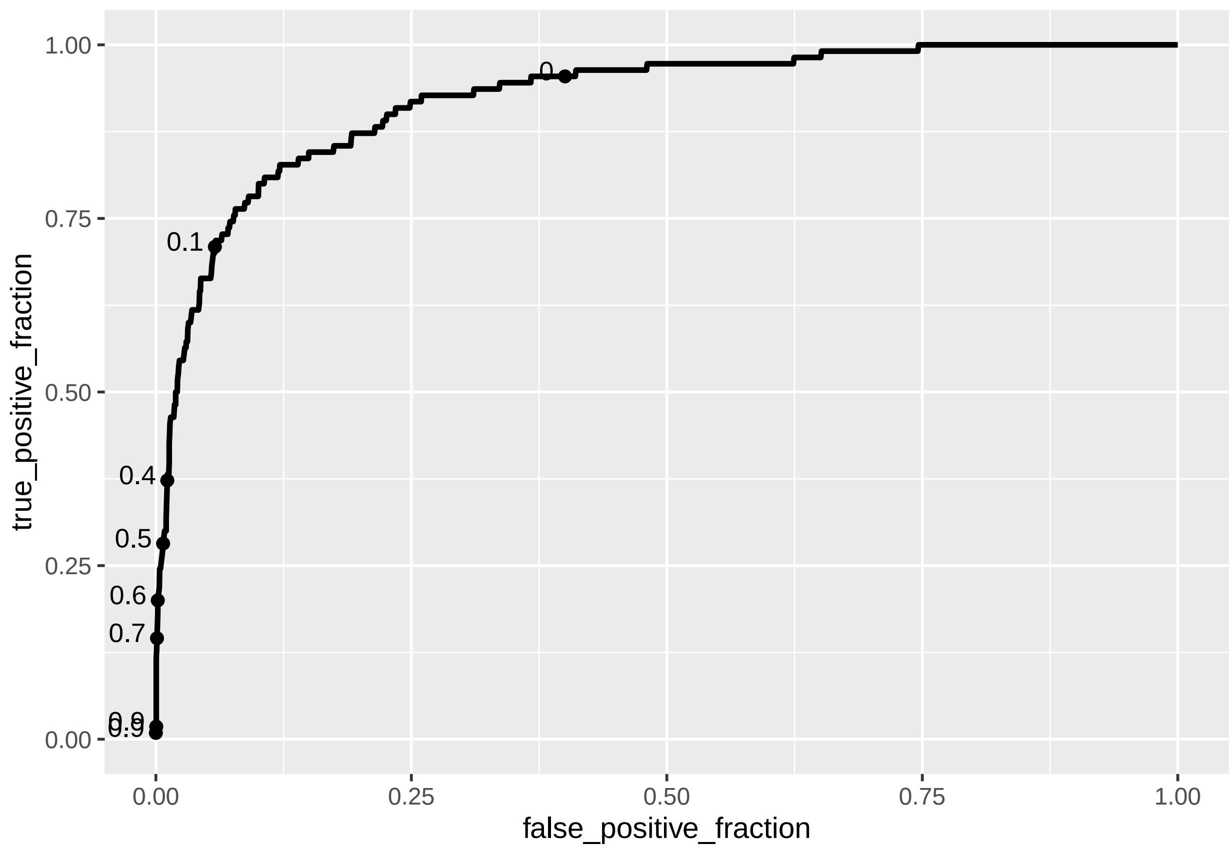
```
preds_LOOCV = NULL
for(i in 1:nrow(movies_train)){
  mod = glm(blockbuster ~ budgetM + top_director + cast_total_facebook_likes000s
            + content_rating + genre_main, data = movies_train[-i,], family = binomial)
  preds_LOOCV[i] = predict(mod, newdata = movies_train[i,])
}
```

- g) Let's plot the ROC curves for each of our predictions.

```
library(plotROC)
```

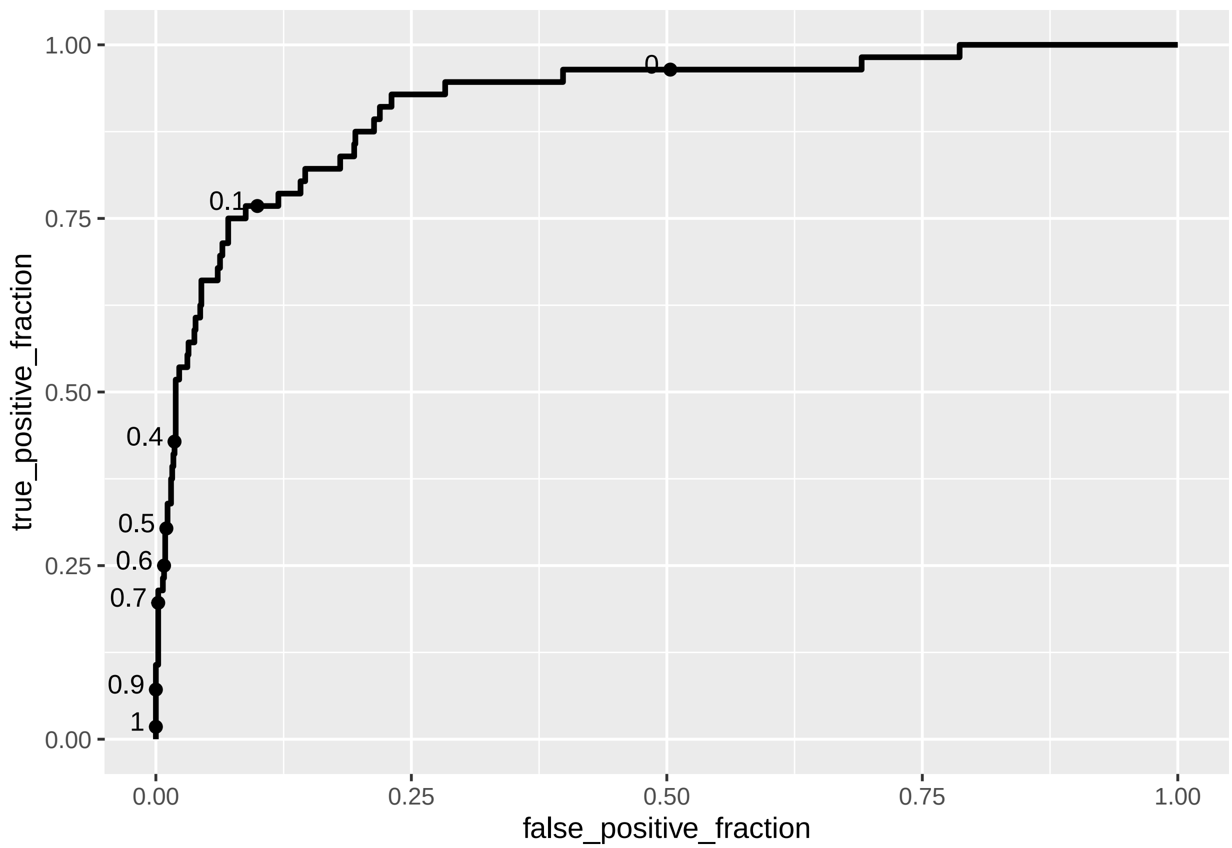
```
## Loading required package: ggplot2
```

```
TrainROC = ggplot(predsTrain, aes(m = scores, d = blockbuster)) +
  geom_roc(labelsize = 3.5,
           cutoffs.at = c(0.99, 0.9, 0.7, 0.6, 0.5, 0.4, 0.1, 0.01))
print(TrainROC)
```



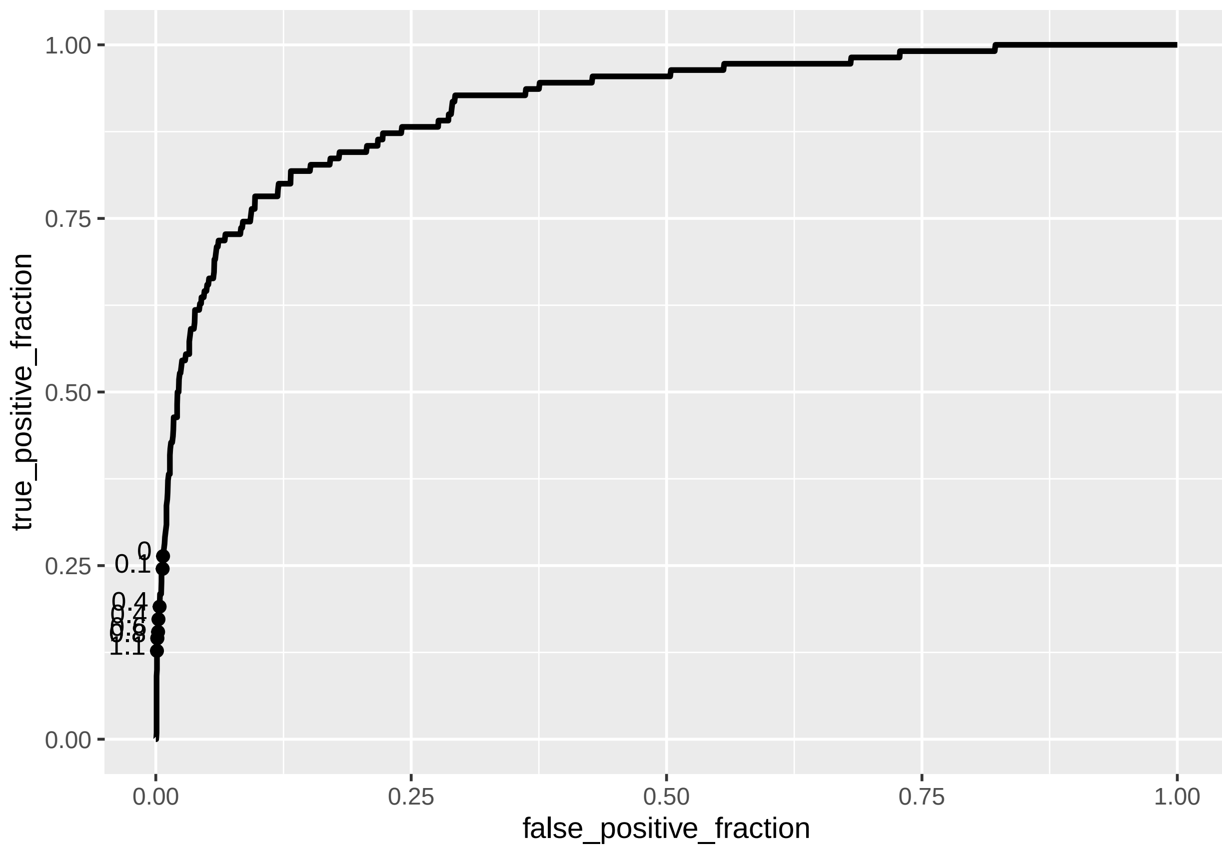
This is a generally good looking curve and appears that a 0.1 cutoff would yield the highest prediction power in the model.

```
TestROC = ggplot(predsTest, aes(m = scores, d = blockbuster)) +
  geom_roc(labelsize = 3.5,
           cutoffs.at = c(0.99, 0.9, 0.7, 0.6, 0.5, 0.4, 0.1, 0.01))
print(TestROC)
```



The curve for the test set looks similar to the train but is more rigid, This is likely due to higher variance and less samples in the set.

```
LOOCVDF = data.frame(
  movies_train,
  scores = preds_LOOCV
)
LOOROC = ggplot(LOOCVDF, aes(m = scores, d = blockbuster)) +
  geom_roc(labelsize = 3.5,
           cutoffs.at = c(0.99, 0.9, 0.7, 0.6, 0.5, 0.4, 0.1, 0.01))
print(LOOROC)
```



Again this curve looks similar to the others but appears to have smoothed the curve coming from the test set. Predicting using LOOCV may have helped reduce the variation that we were getting from the test set.

- h) Displaying AUC values.

```
LOOCVAUC = calc_auc(LOOROC)
print(LOOCVAUC)
```

```
##   PANEL group       AUC
## 1     1    -1 0.9108888
```

```
TestAUC = calc_auc(TestROC)
print(TestAUC)
```

```
##   PANEL group       AUC
## 1     1    -1 0.9148524
```

```
TrainAUC = calc_auc(TrainROC)
print(TrainAUC)
```

```
##   PANEL group       AUC
## 1     1    -1 0.9224581
```

In general we expect to see better in sample performance which explains the higher value of our train AUC. Test AUC was higher than LOOCV likely due to increased variance in the set which was reduced by using LOOCV. All of the AUCs were high which means that our model has relatively high predicting power.