Project

Renjie Wei

12/16/2017

Data Preparation

Before going through the details of those two prediction problems, I first prepare and clean the data as follows.

```
# Download the tmdb movie dataset from Kaggle.
setwd("~/Desktop/2018Fall/STAT542/HW5")
raw = read.csv("tmdb_5000_movies.csv",stringsAsFactors = F)
### Data Clean
# 1. Remove instances which have at least one NA variable
movies = raw
movies = movies[complete.cases(movies), ]
# 2. Remove instances which are duplicated (duplicated based on title)
movies = movies[!duplicated(movies$title),]
# 3. Extract genresID, keywordsID, companyID and countryABB
library(stringr)
genresID = str_extract_all(movies$genres, "[0-9]+[0-9]")
for( i in 1:dim(movies)[1]){
  # extract first three genres of each movie if possible; otherwise, return 0
  if(is.na(genresID[[i]][1])){movies$genresID1[i] = 0}else{movies$genresID1[i] =
    genresID[[i]][1]}
  if(is.na(genresID[[i]][2])){movies$genresID2[i] = 0}else{movies$genresID2[i] =
    genresID[[i]][2]}
  if(is.na(genresID[[i]][3])){movies$genresID3[i] = 0}else{movies$genresID3[i] =
    genresID[[i]][3]}
}
#extract first keyword
movies$keywordsID = str_extract(movies$keywords, "[0-9]+[0-9]")
#extract produce company
movies$companyID = str_extract(movies$production_companies, "[0-9]+[0-9]")
#extarct produce country
movies$countryABB = substr(movies$production_countries,18,19)
# 4. Change data types
movies$release_date = as.Date(movies$release_date)
movies$original_language = as.factor(movies$original_language)
movies$genresID1 = as.factor(movies$genresID1)
movies$genresID2 = as.factor(movies$genresID2)
movies$genresID3 = as.factor(movies$genresID3)
movies$keywordsID = as.factor(movies$keywordsID)
movies$companyID = as.factor(movies$companyID)
movies$countryABB = as.factor(movies$countryABB)
# 5. Extract year and month
movies$year = format(movies$release_date,"%Y") #extract year
movies$year = as.integer(movies$year)
movies$month = format(movies$release_date,"%m") #extract month
movies$month = as.integer(movies$month)
```

As required, I ignore the information after the release data, for example, popularity and vote_count. In addition, I select the covariates which I hypothesize might influence those to target variables.

```
# Select variables that I am interested in (including dependet variables)
data = na.omit(movies)
data = data[,c(1,4,6,14,21:23,25:28,13,19)]
colnames(data)
```

```
##  [1] "budget"            "id"                "original_language" "runtime"           "genresID1"
```

I use all odd id as the training data and even id as testing data.

```
# Use all odd id as training data and even id as testing data.
is.odd = function(x) {x %% 2 != 0}
is.even = function(x) {x %% 2 == 0}
train = data[which(is.odd(data$id)),]
test = data[which(is.even(data$id)),]
```

Data Analysis 1: Predictng Revenue

In this section, I focus on the revenue variable. I first remove the missing data as well as the outliers in the revenue variable.

```
train1 = train[which(train$revenue != 0),] #Remove missing data in train
train1 = subset(train1, train1$revenue<1.0e+09) #Remove outlier in train
test1 = test[which(test$revenue != 0),] #Remove missing data in test
```

The main goal is to predict the revenue using my selected covariates and examine whether each covariate is important to the target variable. Based on this goal, I set up "X" and "y" as follows. The reason why I modify original_language and country into dummy variables is that: for one thing, I reduce the computation complexity since most movies are produced by US and spoken in English; for another, if I use original variables to fit prediction models, the factors original_language and country might have new levels in the teasing data set that were not captured in the training data set.

```
X1 = train1[,c(1,3:11)]
colnames(X1) #independet variables that I am intereted in
```

```
##  [1] "budget"            "original_language" "runtime"           "genresID1"         "genresID2"
```

```
y1 = train1[,12] #depentent variable #hist(y1) #distribution of dependent variable
# Independent variable change: original_language and country
X1$original_language = ifelse(X1$original_language=="en", 1,0)
X1$original_language = as.factor(X1$original_language)
X1$countryABB = ifelse(X1$countryABB=="US", 1,0)
X1$countryABB = as.factor(X1$countryABB)
test1$original_language = ifelse(test1$original_language=="en", 1,0)
test1$original_language = as.factor(test1$original_language)
test1$countryABB = ifelse(test1$countryABB=="US", 1,0)
test1$countryABB = as.factor(test1$countryABB)
```

To predict the target variable "revenue", I consider Generalized Additive Models (GAM). This is a flexible and smooth technique which captures the Non linearity in the data and helps us to fit non linear models. Simply saying GAMs are just a generalized version of linear models in which the predictors Xi depend linearly or non linearly on some smooth non linear functions like splines, polynomials or step functions etc.

The regression function F(x) gets modified in GAMs, and only due to this transformation the GAMs are better in terms of generalization to random unseen data, fits the data very smoothly and flexibly without adding complexities or much variance to the model most of the times. The basic idea in splines is that I am going to fit smooth non linear functions on a bunch of predictors Xi. Additive in the name means I am going to fit and retain the additivity of the Linear Models.

I use the gam() function in R to fit a GAM which is linear in ALL independent variables. It turns out that AIC is 62956.47 and that important covariates are budget, genresID1 and runtime.

```
# Fit gam
library(gam)
gamfit1 = gam(y1~., data = data.frame(y1,X1))
summary(gamfit1) #AIC: 62893.63
```

```
##
## Call: gam(formula = y1 ~ ., data = data.frame(y1, X1))
## Deviance Residuals:
##        Min         1Q     Median         3Q        Max
## -3.497e+08 -3.939e+07 -3.353e-08  1.685e+07  6.439e+08
##
## (Dispersion Parameter for gaussian family taken to be 1.510135e+16)
##
##     Null Deviance: 3.937695e+19 on 1564 degrees of freedom
## Residual Deviance: 1.443689e+19 on 956 degrees of freedom
## AIC: 63191.74
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                    Df     Sum Sq    Mean Sq   F value    Pr(>F)
## budget              1 1.8012e+19 1.8012e+19 1192.7504 < 2.2e-16 ***
## original_language   1 9.8424e+15 9.8424e+15    0.6518  0.419685
## runtime             1 1.0415e+17 1.0415e+17    6.8971  0.008772 **
## genresID1          17 9.4892e+17 5.5819e+16    3.6963 6.324e-07 ***
## genresID2          19 4.5409e+17 2.3899e+16    1.5826  0.053500 .
## genresID3          18 2.1504e+17 1.1947e+16    0.7911  0.712453
## companyID         548 5.1558e+18 9.4083e+15    0.6230  1.000000
## countryABB          1 5.6086e+13 5.6086e+13    0.0037  0.951418
## year                1 1.1348e+16 1.1348e+16    0.7514  0.386241
## month               1 2.8712e+16 2.8712e+16    1.9013  0.168257
## Residuals         956 1.4437e+19 1.5101e+16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Then I consider to fit a STEPWISE GAM which is non linear in "release_date" and "runtime" with 6 degrees of freedom because they are fitted using Smoothing Splines , whereas it is Linear in Terms of the other coviarates. It turns out that the selected model is y1 ~ budget + s(runtime, 6) + genresID1 + countryABB + s(month, 6) (AIC= 62526.63); the most important predictors are budget, runtime and genresID.

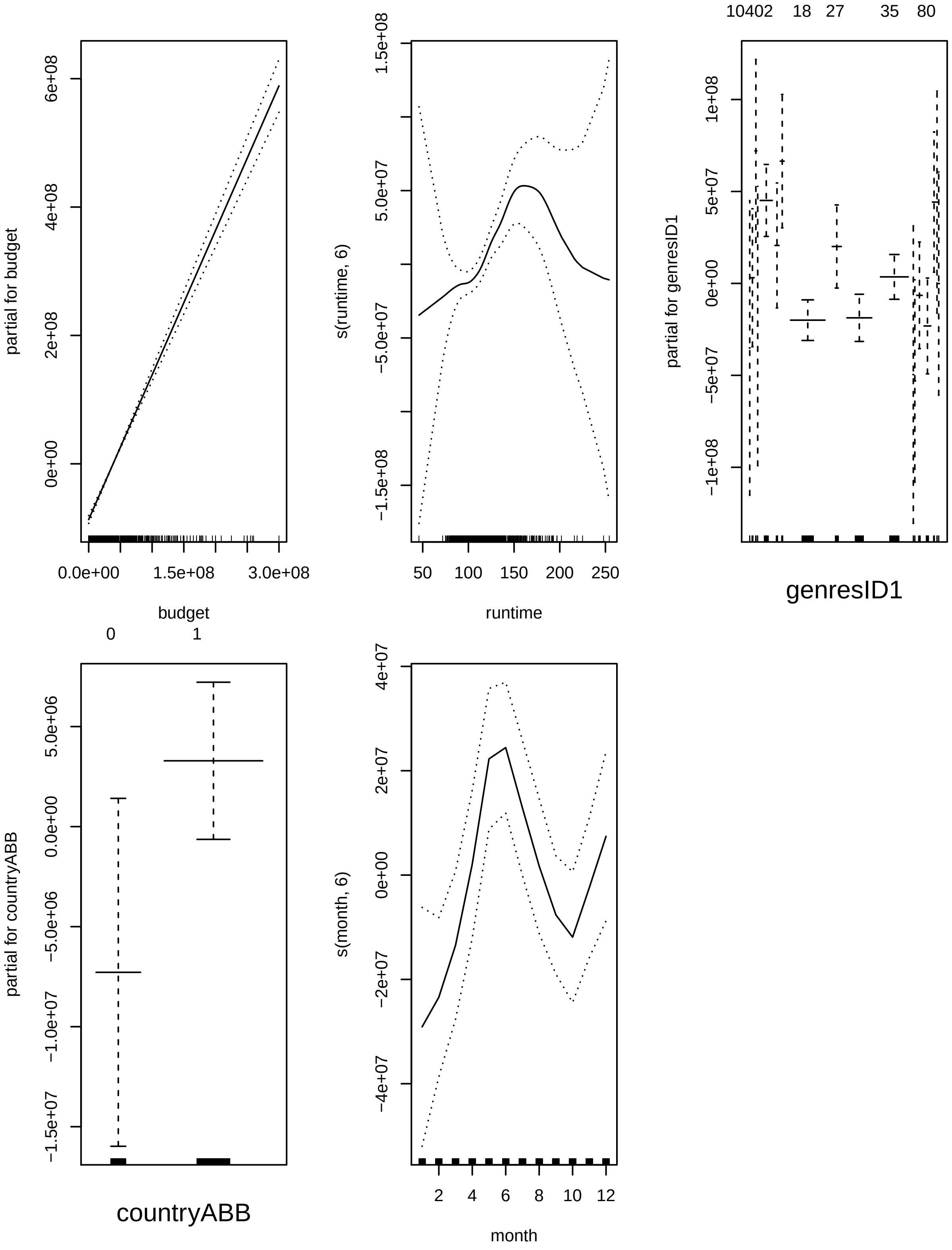
```
 # Stepwise
step.object1 =step.Gam(gamfit1,
                        scope=list("budget"=~1+budget,"original_language"=~1+original_language,
                                   "runtime"=~1+runtime+s(runtime,6),"genresID1"=~1+genresID1,
                                   "genresID2"=~1+genresID2,"genresID3"=~1+genresID3,
                                   "companyID"=~1+companyID,
                                   "countryABB"=~1+countryABB,"year"=~1+year+s(year,6),
                                   "month"=~1+month+s(month,6)))
```

```
## Start:  y1 ~ .; AIC= 63191.74
## Step:1 y1 ~ budget + original_language + runtime + genresID1 + genresID2 +      genresID3 + countryA
## Step:2 y1 ~ budget + original_language + runtime + genresID1 + genresID2 +      genresID3 + countryAB
## Step:3 y1 ~ budget + original_language + runtime + genresID1 + genresID2 +      countryABB + year + s
## Step:4 y1 ~ budget + original_language + s(runtime, 6) + genresID1 +      genresID2 + countryABB + ye
## Step:5 y1 ~ budget + original_language + s(runtime, 6) + genresID1 +      genresID2 + countryABB + s
## Step:6 y1 ~ budget + s(runtime, 6) + genresID1 + genresID2 + countryABB +      s(month, 6) ; AIC= 625
## Step:7 y1 ~ budget + s(runtime, 6) + genresID1 + countryABB + s(month,      6) ; AIC= 62526.63
```

```r
summary(step.object1)
```

```
##
## Call: gam(formula = y1 ~ budget + s(runtime, 6) + genresID1 + countryABB +
##     s(month, 6), data = data.frame(y1, X1), trace = FALSE)
## Deviance Residuals:
##        Min         1Q     Median         3Q        Max
## -555197457  -52163889  -13228707   25323869  695680785
##
## (Dispersion Parameter for gaussian family taken to be 1.287068e+16)
##
##     Null Deviance: 3.937695e+19 on 1564 degrees of freedom
## Residual Deviance: 1.973074e+19 on 1533 degrees of freedom
## AIC: 62526.63
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                 Df     Sum Sq    Mean Sq   F value    Pr(>F)
## budget           1 1.6762e+19 1.6762e+19 1302.3636 < 2.2e-16 ***
## s(runtime, 6)    1 9.4604e+16 9.4604e+16    7.3504   0.00678 **
## genresID1       17 9.3404e+17 5.4943e+16    4.2689 1.312e-08 ***
## countryABB       1 3.6344e+16 3.6344e+16    2.8238   0.09308 .
## s(month, 6)      1 1.9091e+16 1.9091e+16    1.4833   0.22344
## Residuals     1533 1.9731e+19 1.2871e+16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##               Npar Df Npar F     Pr(F)
## (Intercept)
## budget
## s(runtime, 6)       5 2.2904   0.04365 *
## genresID1
## countryABB
## s(month, 6)         5 5.9406 1.893e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#Plotting the Model
par(mfrow=c(1,3)) #to partition the Plotting Window
plot(step.object1,se = TRUE)
```



The above image has three different plots for each variable included in the Model. The X-axis contains the variable values itself and the Y-axis contains the response values i.e the revenue. From the plots and their shapes we can see that the revenue increases with “budget”. Revenue first increases with “runtime” then decreases over around 150 minutes. For variable “month” the revenue is highest around June and December.

And for the Categorical variable “genresID” and “countryABB”, the revenue, for example, increases if the country is US. The curvy shapes for the variables “runtime” and “month” is due to the Smoothing splines which models the Non linearity in the data. The dotted Lines around the main curve lines are the Standard Error Bands.

Using the selected model to fit the teasing data, I have the predictions and calculate errors as follows. Note that when I calculate the mean error, I only include 95% smallest errors because a couple of errors are extremely large (> 100).

```r
## Predictions
# "Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = object$xleve
# Remove the new level entry
test1 = test1[-which(test1$genresID1==10769),]
test1$revenuePred = predict(step.object1, test1)
## Error
error = abs(test1$revenuePred-test1$revenue)/test1$revenue
errorMost = sort(error, decreasing = F)[1:round(length(error)*0.95)]
mean(errorMost)
```

```
## [1] 2.017729
```

Data Analysis 2: Predicting Whether Average Vote > 7

In this section, I focus on a classification problem which aims predict whether the vote_average is greater than 7. I consider the same predictors as Data Analysis 1.

```r
train2 = train
test2 = test
X2 = train2[,c(1,3:7,9:11)]
colnames(X2) #independet variables that I am intereted in
```

```
## [1] "budget"            "original_language" "runtime"           "genresID1"         "genresID2"
```

```r
y2 = train2[,13] #depentent variable
#hist(y2) #distribution of dependent variable

# Independent variable change: original_language and country
X2$original_language = ifelse(X2$original_language=="en", 1,0)
X2$original_language = as.factor(X2$original_language)
X2$countryABB = ifelse(X2$countryABB=="US", 1,0)
X2$countryABB = as.factor(X2$countryABB)

test2$original_language = ifelse(test2$original_language=="en", 1,0)
test2$original_language = as.factor(test2$original_language)
test2$countryABB = ifelse(test2$countryABB=="US", 1,0)
test2$countryABB = as.factor(test2$countryABB)
```

To predict whether the average vote is greater than 7, I fit a Logistic Regression Model using GAMs for predicting the probabilities of the binary response values. I use the identity I() function to convert the Response to a Binary variable. In this Logistic Regression Model I try to find the conditional probability for the Wage variable which can take 2 values either, P(vote.average $> 7|$Xi) and P(vote.average $< 7|$Xi). It turns out that the best model is I(y2 > 7) ~ original_language + s(runtime, 6) + genresID1 + genresID2+ genresID3 + countryABB + s(year, 6) + month, (AIC= 1635.337); the most important predictors are original_language, runtime, genresID1-3 and year.

```r
#logistic Regression Model
gamfit2 = gam(I(y2 > 7) ~ . ,data=data.frame(y2, X2),family=binomial) #AIC: 1689.451
```

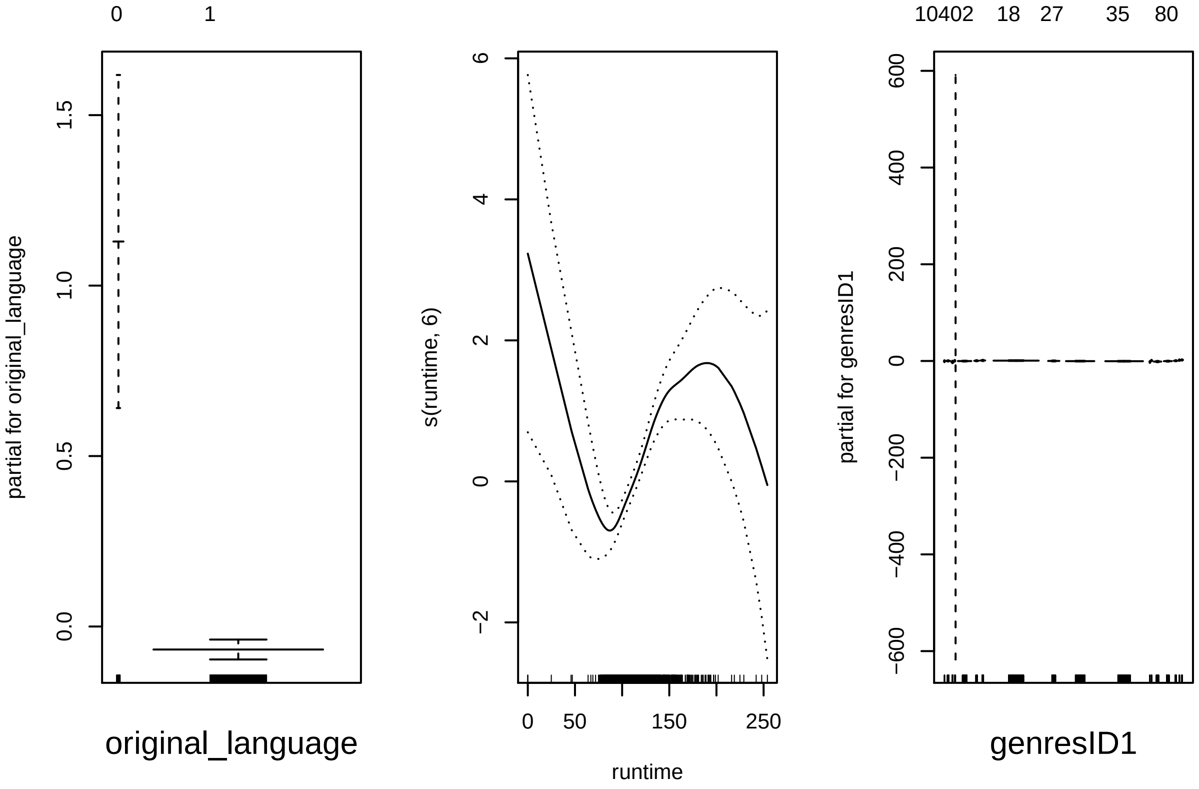
```r
# Stepwise
step.object2 =step.Gam(gamfit2,
                       scope=list("budget"=~1+budget,"original_language"=~1+original_language,
                                  "runtime"=~1+runtime+s(runtime,6),"genresID1"=~1+genresID1,
                                  "genresID2"=~1+genresID2,"genresID3"=~1+genresID3,
                                  "countryABB"=~1+countryABB,"year"=~1+year+s(year,6),
                                  "month"=~1+month+s(month,6)))
```

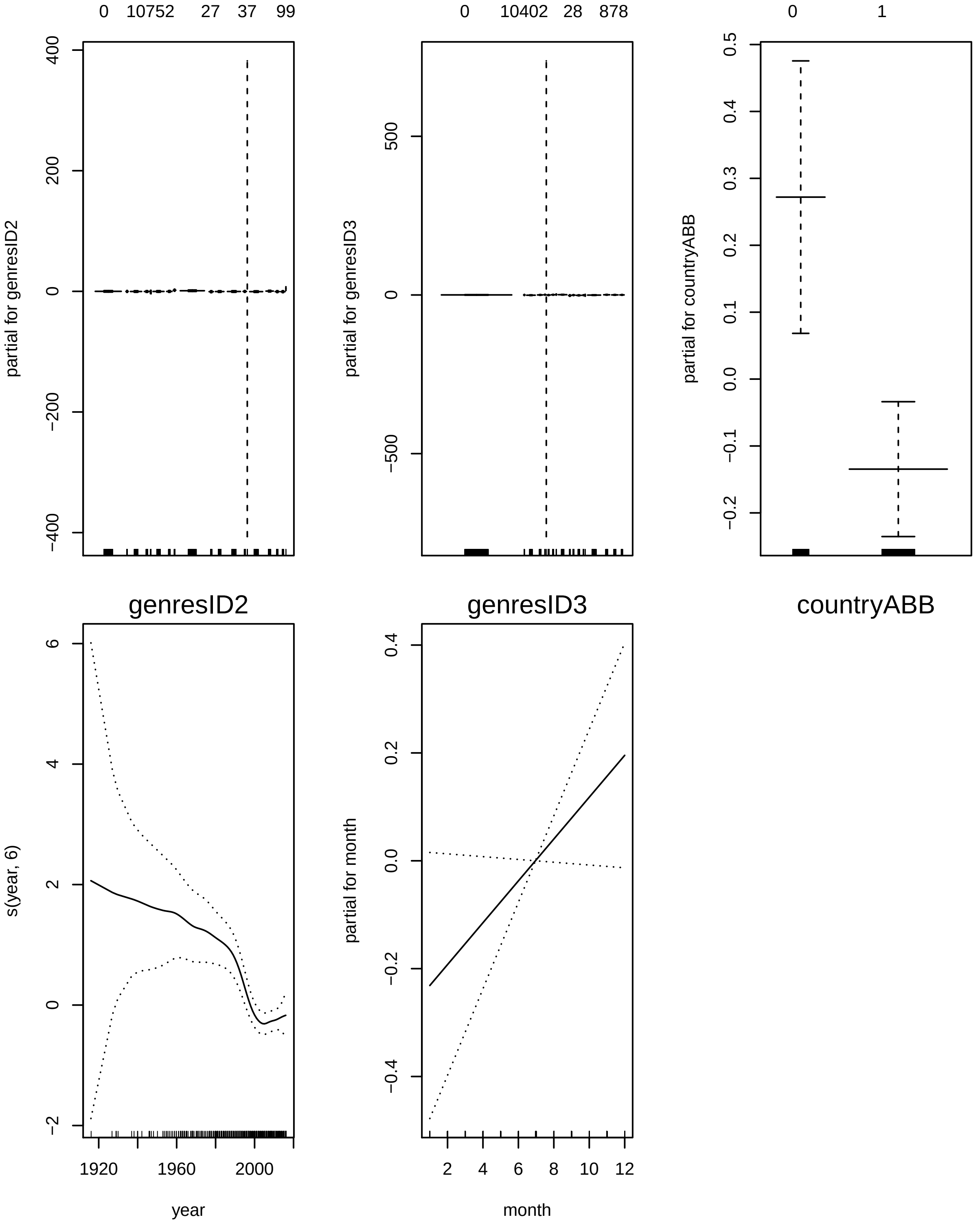
```
## Start:  I(y2 > 7) ~ .; AIC= 1664.995
## Step:1 I(y2 > 7) ~ budget + original_language + s(runtime, 6) + genresID1 +      genresID2 + genresI
## Step:2 I(y2 > 7) ~ budget + original_language + s(runtime, 6) + genresID1 +      genresID2 + genresI
## Step:3 I(y2 > 7) ~ original_language + s(runtime, 6) + genresID1 + genresID2 +      genresID3 + coun
```

```r
summary(step.object2)
```

```
##
## Call: gam(formula = I(y2 > 7) ~ original_language + s(runtime, 6) +
##     genresID1 + genresID2 + genresID3 + countryABB + s(year,
##     6) + month, family = binomial, data = data.frame(y2, X2),
##     trace = FALSE)
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9716 -0.5768 -0.3633 -0.2209  2.6408
##
## (Dispersion Parameter for binomial family taken to be 1)
##
##     Null Deviance: 1950.609 on 2056 degrees of freedom
## Residual Deviance: 1493.337 on 1986 degrees of freedom
## AIC: 1635.337
##
## Number of Local Scoring Iterations: 12
##
## Anova for Parametric Effects
##                     Df  Sum Sq Mean Sq F value    Pr(>F)
## original_language    1   34.20  34.201 37.6007 1.045e-09 ***
## s(runtime, 6)        1   77.02  77.016 84.6722 < 2.2e-16 ***
## genresID1           18   44.62   2.479  2.7250 0.0001205 ***
## genresID2           19   41.27   2.172  2.3880 0.0006737 ***
## genresID3           18   28.93   1.607  1.7671 0.0239717 *
## countryABB           1    2.96   2.957  3.2515 0.0715107 .
## s(year, 6)           1   49.75  49.754 54.7004 2.060e-13 ***
## month                1    3.52   3.516  3.8653 0.0494330 *
## Residuals         1986 1806.41   0.910
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                   Npar Df Npar Chisq    P(Chi)
## (Intercept)
## original_language
## s(runtime, 6)           5     41.338 8.019e-08 ***
## genresID1
## genresID2
## genresID3
## countryABB
## s(year, 6)                 5     15.034   0.01021 *
## month
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Plotting the Model
par(mfrow=c(1,3)) #to partition the Plotting Window
plot(step.object2,se = TRUE)
```





The above plots are almost the same as the first model; the only difference is that the Y-axis is now the Logit log P(X)/(1???P (X))of the probability values, and I now fit using 6 degrees of freedom for the variables “runtime” and “year” and linear in terms of “month” variable. One interesting finding is that for the variable “countryABB” the probability that vote_average> 7 is lower for US than the other countries. Then I use the selected model to fit the teasing data and calculate errors as follows. The prediction accuracy turns out to be 83.06% (error rate is around 17%). Specifically, 68 movies under 7 scores are mistakenly predicted as over 7 scores; 266 movies that are greater than 7 scores are predicted as under 7 scores; 1638 movies are predicted correctly.

```
 ## Predictions
library(e1071)
# Remove the new level entry
test2 = test2[-which(test2$genresID1==10769),]
test2 = test2[-which(test2$genresID1==0),]
test2 = test2[-which(test2$genresID3==99),]
test2$votePred = round(predict(step.object2, test2, type="response"))
## Error
error2 = sum(abs(test2$votePred-ifelse(test2$vote_average>7,1,0)))
error2/dim(test2)[1] #tesing error rate
```

```
## [1] 0.1693712
```

```
 ## Confusion Matrix
library(caret)
confusionMatrix(as.factor(test2$votePred), as.factor(ifelse(test2$vote_average>7,1,0)))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1558  266
##          1   68   80
##
##                Accuracy : 0.8306
##                  95% CI : (0.8133, 0.8469)
##     No Information Rate : 0.8245
##     P-Value [Acc > NIR] : 0.2491
##
##                   Kappa : 0.2445
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9582
##             Specificity : 0.2312
##          Pos Pred Value : 0.8542
##          Neg Pred Value : 0.5405
##              Prevalence : 0.8245
##          Detection Rate : 0.7901
##    Detection Prevalence : 0.9249
##       Balanced Accuracy : 0.5947
##
##        'Positive' Class : 0
##
```

I also consider Support Vector Machines (SVM), Decision Trees (DTs) and Random Forests (RFs) to predict whether the average vote is greater than 7.

```
library(e1071)
svm_tune = tune.svm( y2 ~ ., data = data.frame(X2,y2), cost = 2^(0:4), gamma = 2^(-1:1))
pred = predict(svm_tune$best.model, test2)
error = sum(abs(ifelse(pred>7,1,0) -ifelse(test2$vote_average>7,1,0)))
error/dim(test2)[1] #tesing error rate
```

```
## [1] 0.173428
```

```
library(party)
tree = ctree(y2~., data = data.frame(y2, X2))
pred = predict(tree, test2)
error = sum(abs(ifelse(pred>7,1,0) -ifelse(test2$vote_average>7,1,0)))
error/dim(test2)[1] #tesing error rate
```

```
## [1] 0.1739351
```

```
library(randomForest)
rf = randomForest(y2~., data = data.frame(y2, X2))
pred = predict(rf, test2)
# It turns out the the tesing errors of SVM and DT are higher than that of GAM.
# The testing error of RF is a little bit lower but its computation is more complex than GAM
```

Predicting the Movie "Star Wars: The Last Jedi"

The movie "Star Wars: The Last Jedi" is set to release at Dec 15th. I find its information from imdb.com so that I can apply my models to predict its total revenue and whether the rating is greater than 7. The predicted revenue is 619,780,925 dollars and the probability that the vote is greater than 7 is about 0.3346.

```
StarWar = matrix(data = NA, nrow=1, ncol = 13)
colnames(StarWar) = colnames(test)
StarWar = data.frame(StarWar)
StarWar$budget = 245000000 #budget information from Wikipedia
StarWar$original_language = as.factor(1) #English
StarWar$runtime = 153
StarWar$genresID1 = as.factor(28) #action
StarWar$genresID2 = as.factor(12) #adventure
StarWar$genresID3 = as.factor(14) #fantacy
StarWar$companyID = as.factor(1) #US
StarWar$countryABB = as.factor(1) #US
StarWar$year = 2017
StarWar$month = 12
StarWar$revenuePred = predict(step.object1, newdata = StarWar);
StarWar$revenuePred
```

```
## [1] 619780925
```

```
StarWar$vote_averagePred = predict(step.object2, newdata = StarWar, type = "response");
StarWar$vote_averagePred
```

```
## [1] 0.3345969
```

Conclusion

First I want to explain the reason why I choose logistic regression as my final classification model rather than SVM, DTs or RFs. For one thing, its cumputation complexity is relatively low and the prediction accuracy is relatively high. For another, which is more important, in the scope of management, GAM helps more to provide business insights and managerial implications than the other classification methods. From the results using GAM, I can draw conclusions on the predictor significance as well as the relationship between the predictors and the target; however, taking SVM as an example, I cannot see how a specific predictor impacts on the target variable.

In addition, I want to discuss the final prediction results of the movie "Star War: The Last Jedi". The movie has been released since 12/15 and the imdb score is now 8.1. This means my classification result is not very reliable. Also, I suppose the revenue result is underestimated. One of the limitations of my prediction model is that I assume that the observations are i.i.d, which is not always the case. For example, the movie "Star War: The Last Jedi" is the eighth episode of the series of Star War movies, and its rate or revenue might be influenced by the previous movies - the movie reputation and the loyalty of fans are not captured by my model. To overcome this limitation, I would use text mining techniques to further explore the data features including the movie title, overview and keywords and define whether it is a series movie and what the previous revenues and the previous scores are. Due to the page limit of this homework, those research possibilities are not explored in this report.