Detailed description and implementation of the model: **Comments:** The glmnet package was used to construct logistic regression models for classification of movie genres represented in binary 0-1 values. The binary representation aided in dealing with the multi-label and imbalanced data issues within this data set; data was acquired from TBDM, with missing data being supplemented from IMDB where appropriate.

The cv.glm function was used because it is both stringent and flexible- it allows for selecting the optimal lambda value via built-in tuning and also does built-in cross-validation, but the function also has settings that allow control for the model. For these test runs, the lowest values for thresh and maxit were used in order to run the models quickly and troubleshoot where needed, and see what the accuracy may be on these lower values to test the efficacy of the document-term-matrices.

Further information on the model, and the various control settings can be found here:
https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html
(https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html)

Text2Vec was used to construct document-term-matrices from the Overview field in the data set- the Overview field contains descriptions of the plots of the movies. The objective of creating the DTMs is to use them as “bag-of-words” to drive genre classification via logistic regression. The DTMs were also normalized in order to suppress the overrepresentation of any particular words.

Description of your performance metrics **Comments:** The following performance metrics were used for genre classification:

1. Misclassification Error plots: the misclassification error plots indicate model classification accuracy for different values of lambda. This is one of the reasons why separate models were constructed for each genre- it seems that different values of lambda are needed for higher accuracy by genre. Glmnet fits a generalized linear model via penalized maximum likelihood- a grid of values of lambda, the regularization parameter, is used to compute the regularization path for the lasso or the elasticnet penalty (Glmnet Vignette).
2. Coefficients vs. lambda plots: The model coefficients are shown to vary with different values of lambda. Each curve in this plot relates to a variable. The top-axis is more useful in this case because it indicates the number of nonzero coefficients at the current λ, the effective degrees of freedom (df) (Glmnet Vignette). Certainly the coeffient plots should be used in conjunction with the misclassifcation error plots to choose the optimal value of lambda (if the user simply does not want to use the $min function).
3. Classification accuracy: Misclassification accuracy was calculated from the caret package instead in an attempt to used an “outside” function so that any classification accuracy is not just attributed to the glmnet package algorithms alone.

Careful performance evaluations for the models **Comments:** Please see below: performance evaluations were done by plotting and calculating the metrics mentioned previously.

Visualizations of the metrics for performance evaluation **Comments:** Please see below.

Discussion of the differences between the models, their strengths, weaknesses, etc. **Comments:** The second model used for classification was SVM. In comparison to logistic regression, the SVM does not classify via probabilistic determination, but instead focuses on geometric separation by determining an optimal separating hyperplane.

The advantage of LR is that it is great if there is a lot of noise in the data. Meanwhile, the disadvantage of using an SVM is that it may attemp to separate two classes of data that could be composed of multiple dimensions, and SVMs have difficulty when the classes are not clearly separable.

Discussion of the performances you achieved, and how you might be able to improve them in the future
Comments: Models can be improved by tuning alpha. It will also help to fine tune the values of the maxit and thresh controls. Additionally, it may help to construct differently sized training sets, and measure performance based on increase in training data.

Different accuracies were achieved for different genres, some genres as low as approx. 60%- this is most likely because the Overview words for these genres were very generic. Perhaps increasing the training data size may help.

```
set.seed(200)
suppressWarnings(suppressMessages(library(text2vec)))
suppressWarnings(suppressMessages(library(data.table)))
suppressWarnings(suppressMessages(library(mclust)))

#Note: I followed the directions here to construct a document-term matrix for the Overvi
ew field in the data set: http://text2vec.org/vectorization.html

movies <- read.csv("Meta_data_49_cleaned.csv")

#the genre labels matrix attempts to override the imbalanced data and multi-label proble
ms in this data set
g.labels <- read.csv("Genres_labels_49.csv")

head(movies, 4)
```

```
##   X budget             director
## 1 0      0 Jean-Paul Lilienfeld
## 2 1      0         Terence Hill
## 3 2      0        Steve McQueen
## 4 3      0        Naoko Ogigami
##                                                                  genres
## 1                                      [{u'id': 18, u'name': u'Drama'}]
## 2    [{u'id': 35, u'name': u'Comedy'}, {u'id': 37, u'name': u'Western'}]
## 3     [{u'id': 18, u'name': u'Drama'}, {u'id': 36, u'name': u'History'}]
## 4 [{u'id': 35, u'name': u'Comedy'}, {u'id': 10769, u'name': u'Foreign'}]
##      id
## 1 16277
## 2 11175
## 3 10360
## 4 25656
##                                                              keywords
## 1                   ['adolescence', 'hostage', 'teachers and students']
## 2         ['saloon', 'horse', 'brother', 'arrest', 'spaghetti western']
## 3 ['prison', 'prisoner', 'hunger', 'hunger strike', 'northern ireland']
## 4                                ['independent film', 'woman director']
##
```

```
                  overview
## 1
```

```
                                                                        A troubled a
nd emotionally fragile woman finds herself at the center of a firestorm, in this gut-wre
nching psychological drama.
## 2
```

```
                     Lucky Luke becomes the Sheriff of Daisy Town and runs out all the c
riminals. Then the Dalton brothers arrive and try to get the Indians to break the peace
 treaty and attack the town.
## 3
```

```
                                                                      The story
```

```
of Bobby Sands, the IRA member who led the 1981 hunger strike in which Republican prison
ers tried to win political status. It dramatises events in the Maze prison in the six we
eks prior to Sands' death.
## 4 Harried and overworked, Taeko (Kobayashi Satomi) leaves the city for a much-needed
 island vacation. Stepping off a propeller plane onto golden sands, she drags her giant
 suitcase across the beach to Hamada Inn. Owner Yuji (Mitsuishi Ken) is impressed; it's
 the first time in three years a guest has made it to his inn without getting lost. The
 next morning Taeko wakes up to the greetings of peculiar fellow vacationer Sakura (Mota
i Masako) who leads the townspeople in funny morning calisthenics on the beach. Thus beg
ins Taeko's strange vacation on this strange island full of strange people. At first Tae
ko finds the laidback attitude and mass idleness hard to bear, but soon she too begins t
o see the joy in "twilighting".
##   popularity                      poster_path releaseyear revenue runtime
## 1   0.159173 /pqFtrX1t40LzMEBBcWEvh4Fc08w.jpg        2008       0      88
## 2   0.136823 /qmUrVZ1jYv766AwqGVwGrqnna3O.jpg        1991       0      92
## 3   0.834962 /397BUdT1bIt9tICgq7wSbF05Dsy.jpg        2008 2724474      96
## 4   0.065552  /P3HvurbR1Fc3lXNDVQ6ksMZZUP.jpg        2007       0     106
##        title
## 1  Skirt Day
## 2 Lucky Luke
## 3     Hunger
## 4    Glasses
```

```
head(g.labels, 4)
```

```
##   X Action Adventure Animation Comedy Crime Documentary Drama Family
## 1 0      0         0         0      0     0           0     1      0
## 2 1      0         0         0      1     0           0     0      0
## 3 2      0         0         0      0     0           0     1      0
## 4 3      0         0         0      1     0           0     0      0
##   Fantasy History Horror Music Mystery Romance Science.Fiction TV.Movie
## 1       0       0      0     0       0       0               0        0
## 2       0       0      0     0       0       0               0        0
## 3       0       1      0     0       0       0               0        0
## 4       0       0      0     0       0       0               0        0
##   Thriller War Western
## 1        0   0       0
## 2        0   0       1
## 3        0   0       0
## 4        0   0       0
```

```
movies <- cbind(movies, g.labels)

#coerce data table
setDT(movies)

#create key on data table called id
setkey(movies, id)

#set all_id variable to easily refer to ids when splitting
all_ids = movies$id

#set train ids, difference is test set ids
train_ids = sample(all_ids, 2500)
test_ids = setdiff(all_ids, train_ids)

#subset by ids
train = movies[J(train_ids)]
test = movies[J(test_ids)]
```

```
#The objective is to create a document term matrix, this chunk of code creates a vocabul
ary vector.

prep_fun = tolower
tok_fun = word_tokenizer

#itoken() creates an iterator over tokens
it_train = itoken(as.character(train$overview),
             preprocessor = prep_fun,
             tokenizer = tok_fun,
             ids = train$id,
             progressbar = FALSE)

vocab = create_vocabulary(it_train)
vocab
```

```
## Number of docs: 2500
## 0 stopwords:  ...
## ngram_min = 1; ngram_max = 1
## Vocabulary:
##           terms terms_counts doc_counts
##     1:     koen            2          1
##     2: literacy            1          1
##     3:   taught            1          1
##     4:   mourns            1          1
##     5:      mui            4          1
##    ---
## 17105:  hipster            2          2
## 17106: earnshaw            1          1
## 17107:  getting           27         24
## 17108:     1691            1          1
## 17109:   before          111        102
```

```
#Document term matrix is created from vocab vector
vectorizer = vocab_vectorizer(vocab)
dtm_train = create_dtm(it_train, vectorizer)

#check the dimensions of the DTM to ensure same # of rows as train set, this will be imp
ortant when running the model
dim(dtm_train)
```

```
## [1]  2500 17109
```

```
#repeat above steps for test set
ts.prep_fun = tolower
ts.tok_fun = word_tokenizer

it_test = itoken(as.character(test$overview),
             preprocessor = ts.prep_fun,
             tokenizer = ts.tok_fun,
             ids = test$id,
             progressbar = FALSE)

ts.vocab = create_vocabulary(it_test)

ts.vocab = create_vocabulary(it_test)
ts.vocab
```

```
## Number of docs: 2014
## 0 stopwords:  ...
## ngram_min = 1; ngram_max = 1
## Vocabulary:
##              terms terms_counts doc_counts
##     1: desperation            1          1
##     2:    lonesome            1          1
##     3:      dickon            1          1
##     4:    peruvian            1          1
##     5:    lombardi            1          1
##    ---
## 14985:    business           36         35
## 14986:  precarious            2          2
## 14987:    currency            1          1
## 14988:     illness            3          3
## 14989:     history           26         26
```

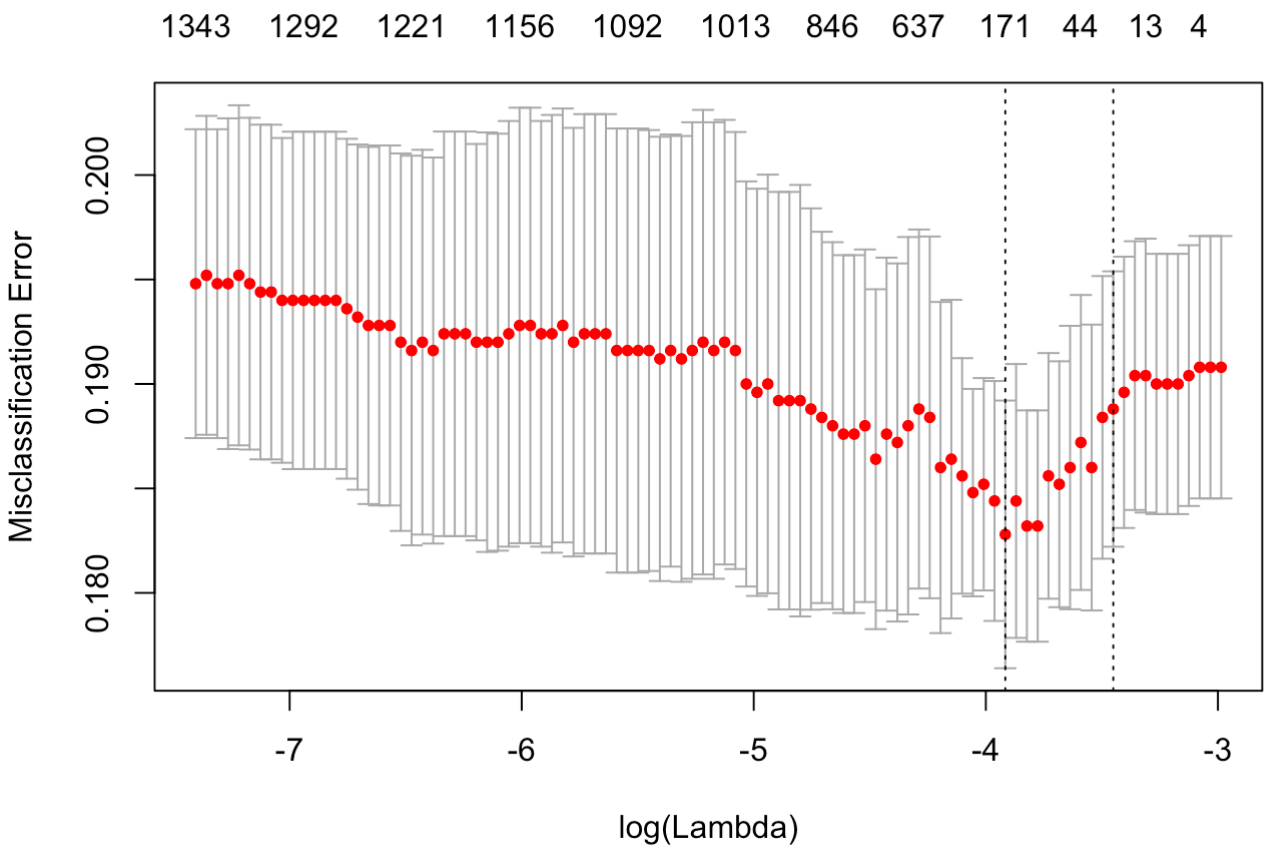
```
ts.vectorizer = vocab_vectorizer(ts.vocab)
dtm_test = create_dtm(it_test, ts.vectorizer)
dim(dtm_test)
```

```
## [1]  2014 14989
```

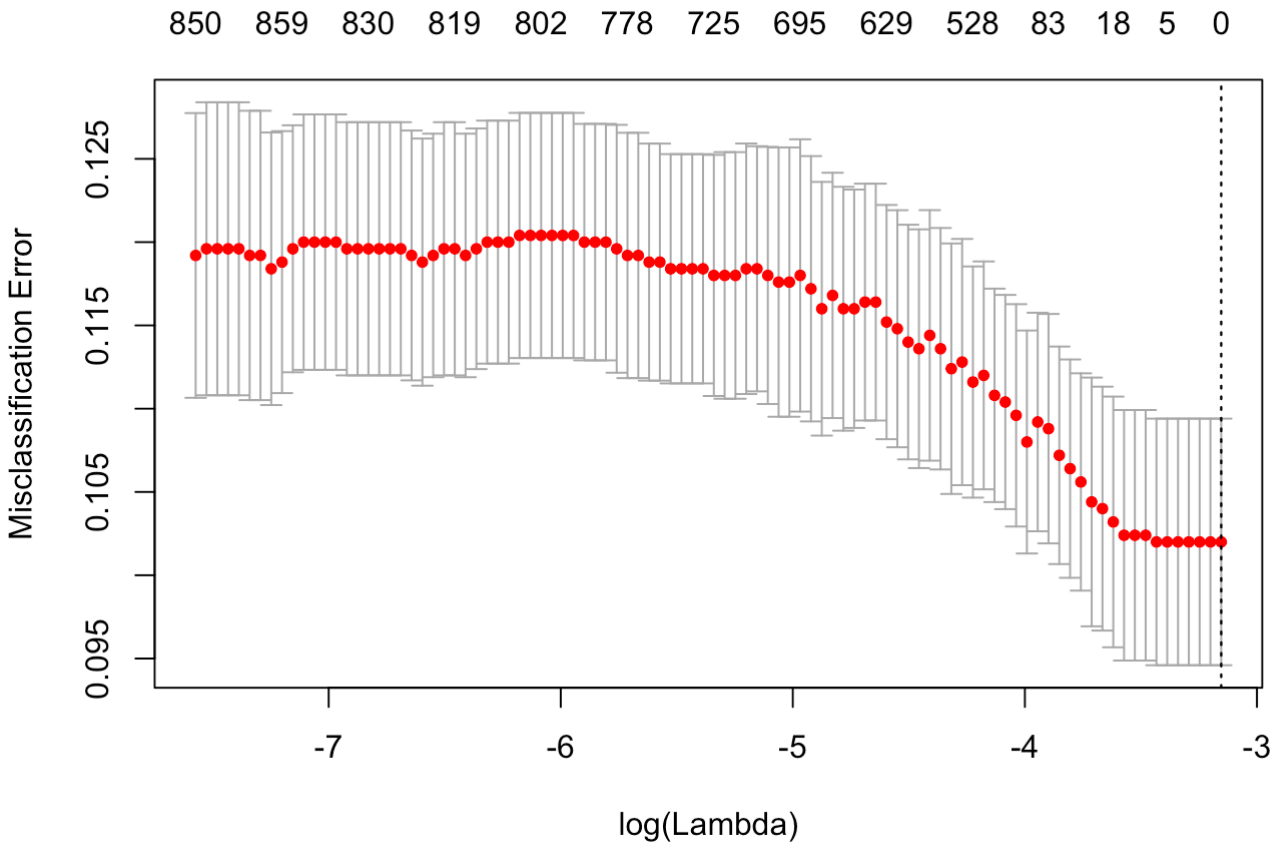
```
#normalize DTMs to remove unequal weight from overrepresentation of certain words
tfidf = TfIdf$new()

dtm_train_tfidf = fit_transform(dtm_train, tfidf)

dtm_test_tfidf  = create_dtm(it_test, vectorizer) %>%
  transform(tfidf)
```

```
suppressWarnings(suppressMessages(library(glmnet)))

NFOLDS = 5

#set to binomial due to 0-1 labeling in the genre lables matrix, alpha was not tuned. c
v.glmnet auto-tunes for the best value of lambda

glmnet_classifierA = cv.glmnet(x = dtm_train_tfidf, y = as.matrix(train[,15]),
                               family = 'binomial',
                               alpha = 1,
                               type.measure = "class",
                               nfolds = NFOLDS,
                               thresh = 1e-3,
                               maxit = 1e3)

glmnet_classifierB = cv.glmnet(x = dtm_train_tfidf, y = as.matrix(train[,16]),
                               family = 'binomial',
                               alpha = 1,
                               type.measure = "class",
                               nfolds = NFOLDS,
                               thresh = 1e-3,
                               maxit = 1e3)

glmnet_classifierC = cv.glmnet(x = dtm_train_tfidf, y = as.matrix(train[,17]),
                               family = 'binomial',
                               alpha = 1,
                               type.measure = "class",
                               nfolds = NFOLDS,
                               thresh = 1e-3,
                               maxit = 1e3)

glmnet_classifierD = cv.glmnet(x = dtm_train_tfidf, y = as.matrix(train[,18]),
                               family = 'binomial',
                               alpha = 1,
                               type.measure = "class",
                               nfolds = NFOLDS,
                               thresh = 1e-3,
                               maxit = 1e3)

glmnet_classifierE = cv.glmnet(x = dtm_train_tfidf, y = as.matrix(train[,19]),
                               family = 'binomial',
                               alpha = 1,
                               type.measure = "class",
                               nfolds = NFOLDS,
                               thresh = 1e-3,
                               maxit = 1e3)

glmnet_classifierF = cv.glmnet(x = dtm_train_tfidf, y = as.matrix(train[,21]),
                               family = 'binomial',
                               alpha = 1,
                               type.measure = "class",
                               nfolds = NFOLDS,
                               thresh = 1e-3,
```

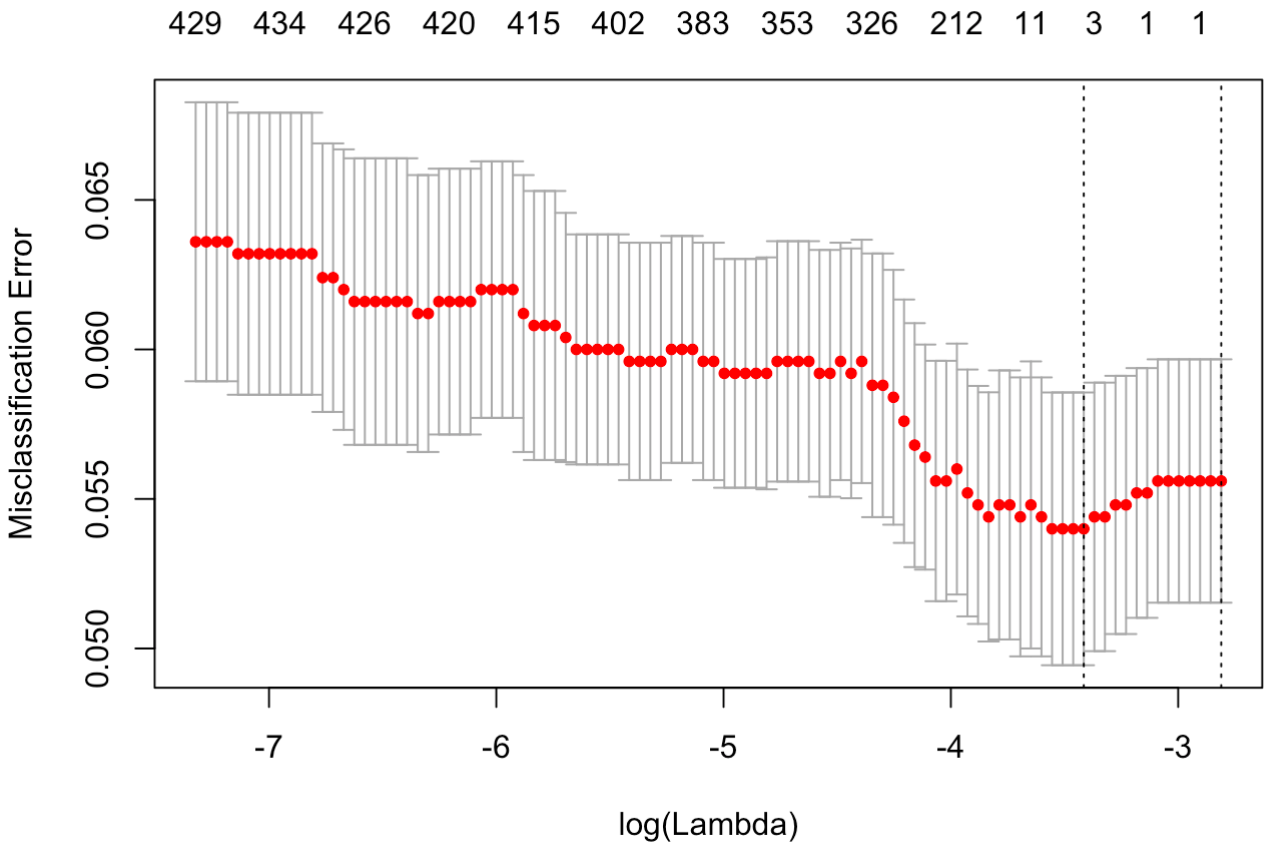
```
                                maxit = 1e3)

glmnet_classifierG = cv.glmnet(x = dtm_train_tfidf, y = as.matrix(train[,25]),
                                family = 'binomial',
                                alpha = 1,
                                type.measure = "class",
                                nfolds = NFOLDS,
                                thresh = 1e-3,
                                maxit = 1e3)

glmnet_classifierH = cv.glmnet(x = dtm_train_tfidf, y = as.matrix(train[,31]),
                                family = 'binomial',
                                alpha = 1,
                                type.measure = "class",
                                nfolds = NFOLDS,
                                thresh = 1e-3,
                                maxit = 1e3)

#Missclassification error plots for different values of lambda
plot(glmnet_classifierA)
```



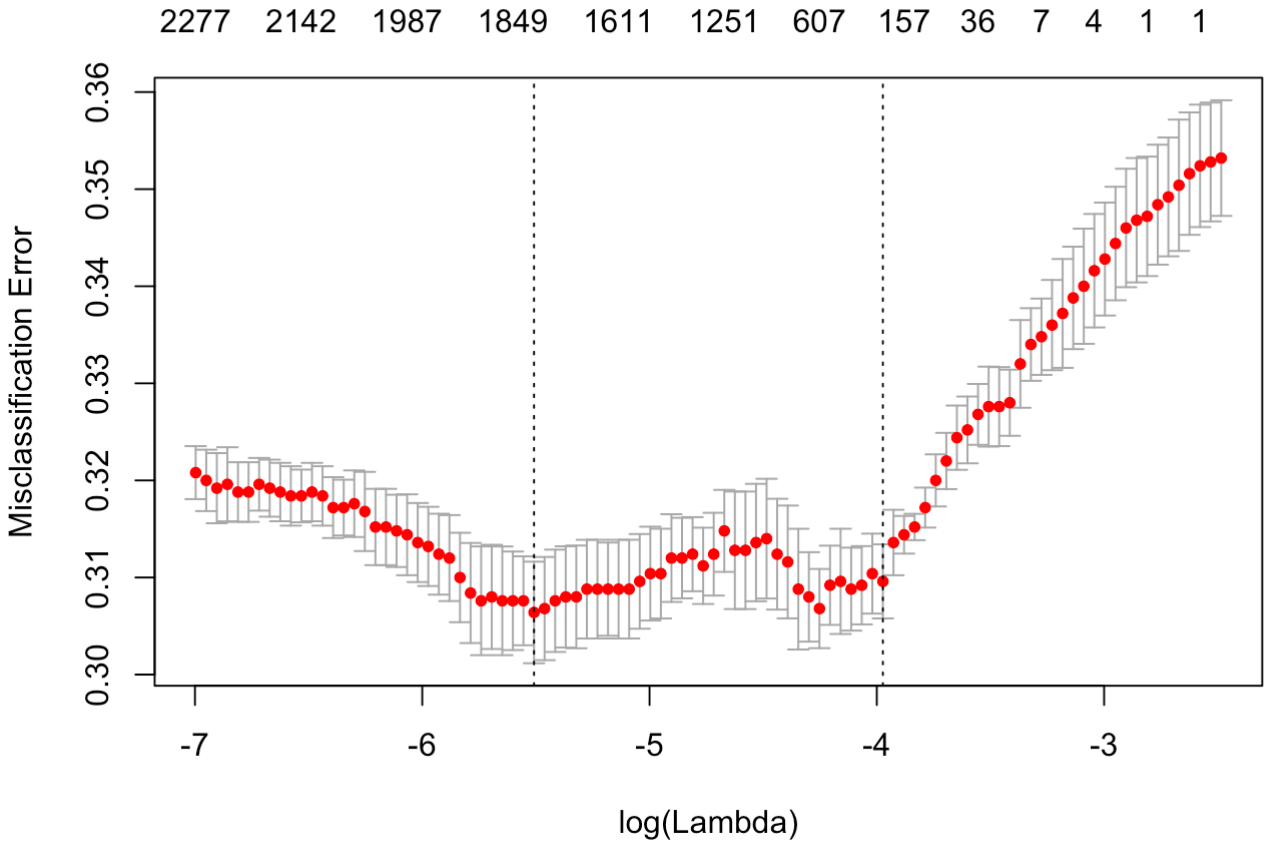

```
plot(glmnet_classifierB)
```



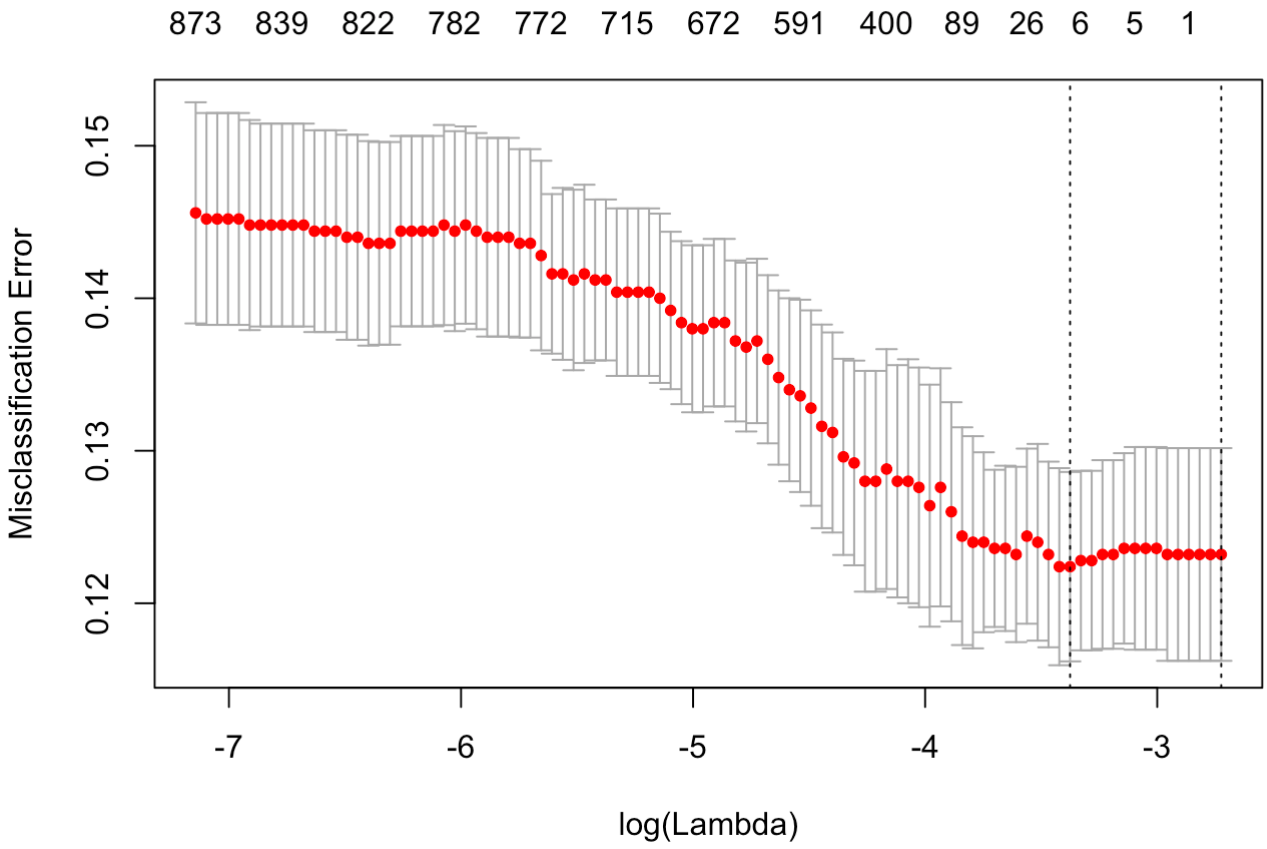

```
plot(glmnet_classifierC)
```



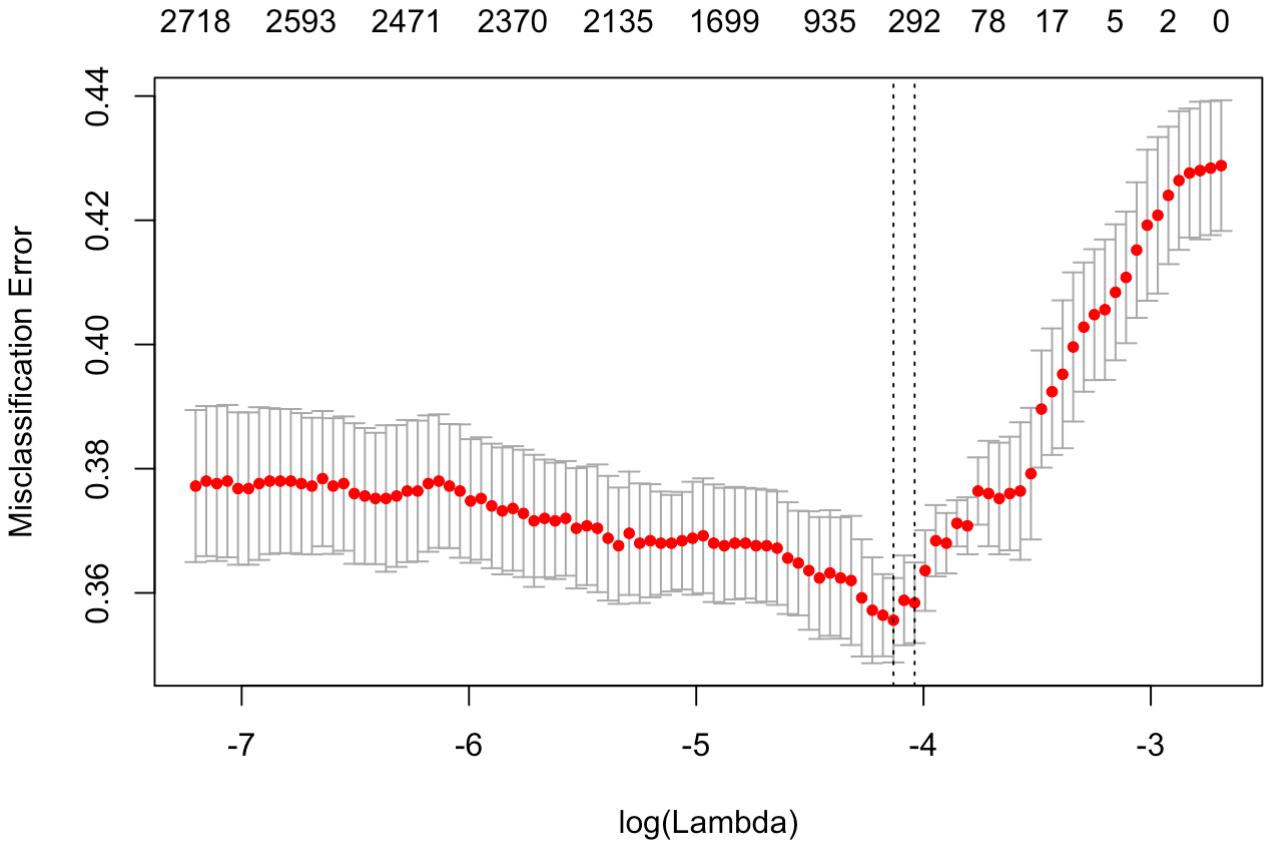

```
plot(glmnet_classifierD)
```



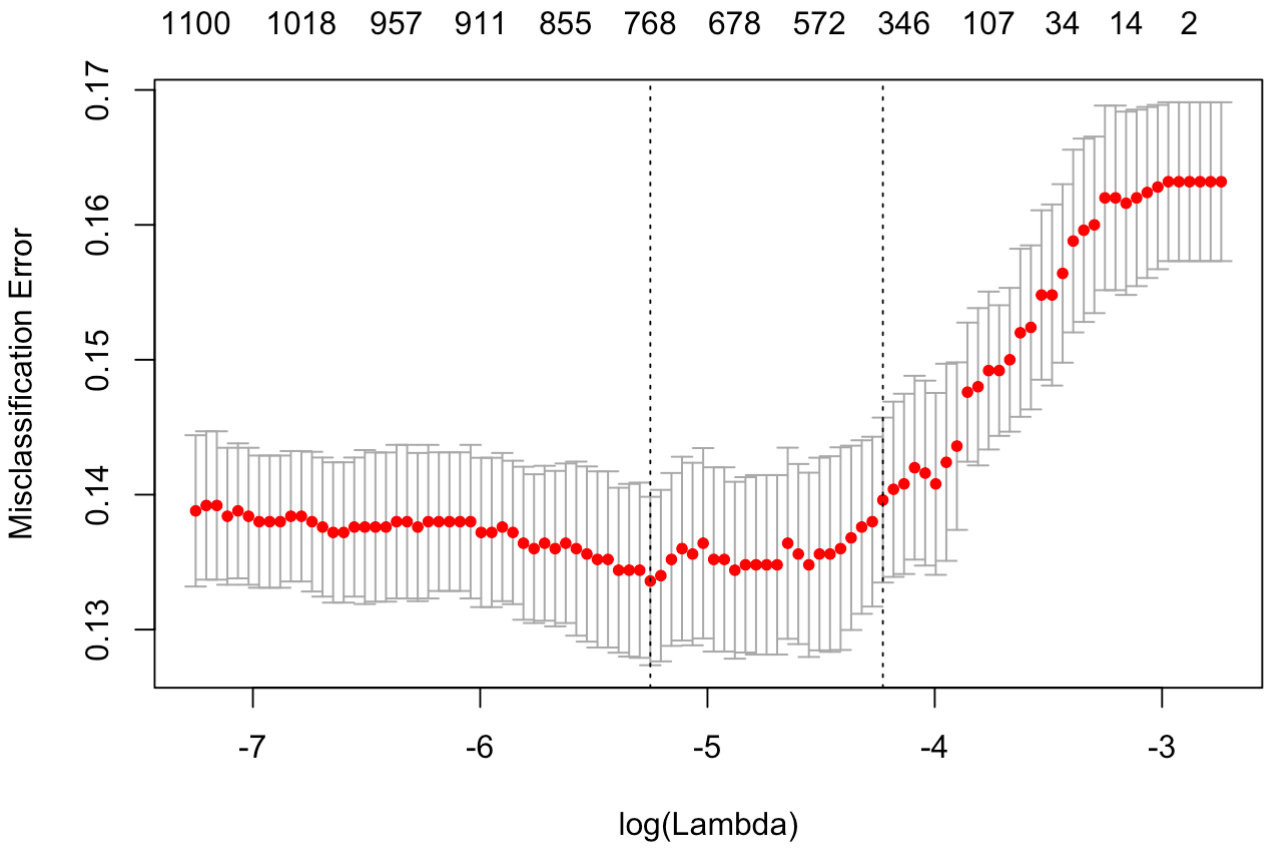

```
plot(glmnet_classifierE)
```



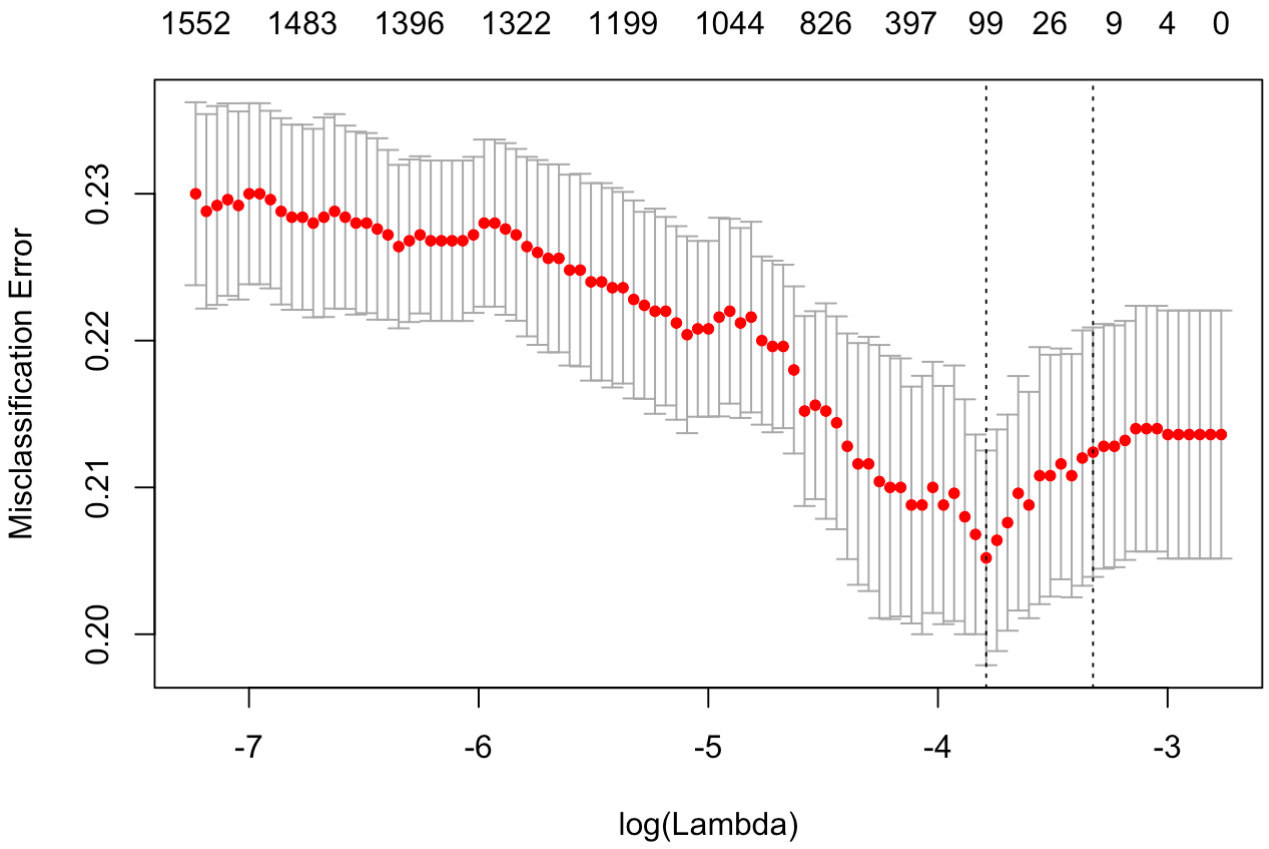

```
plot(glmnet_classifierF)
```



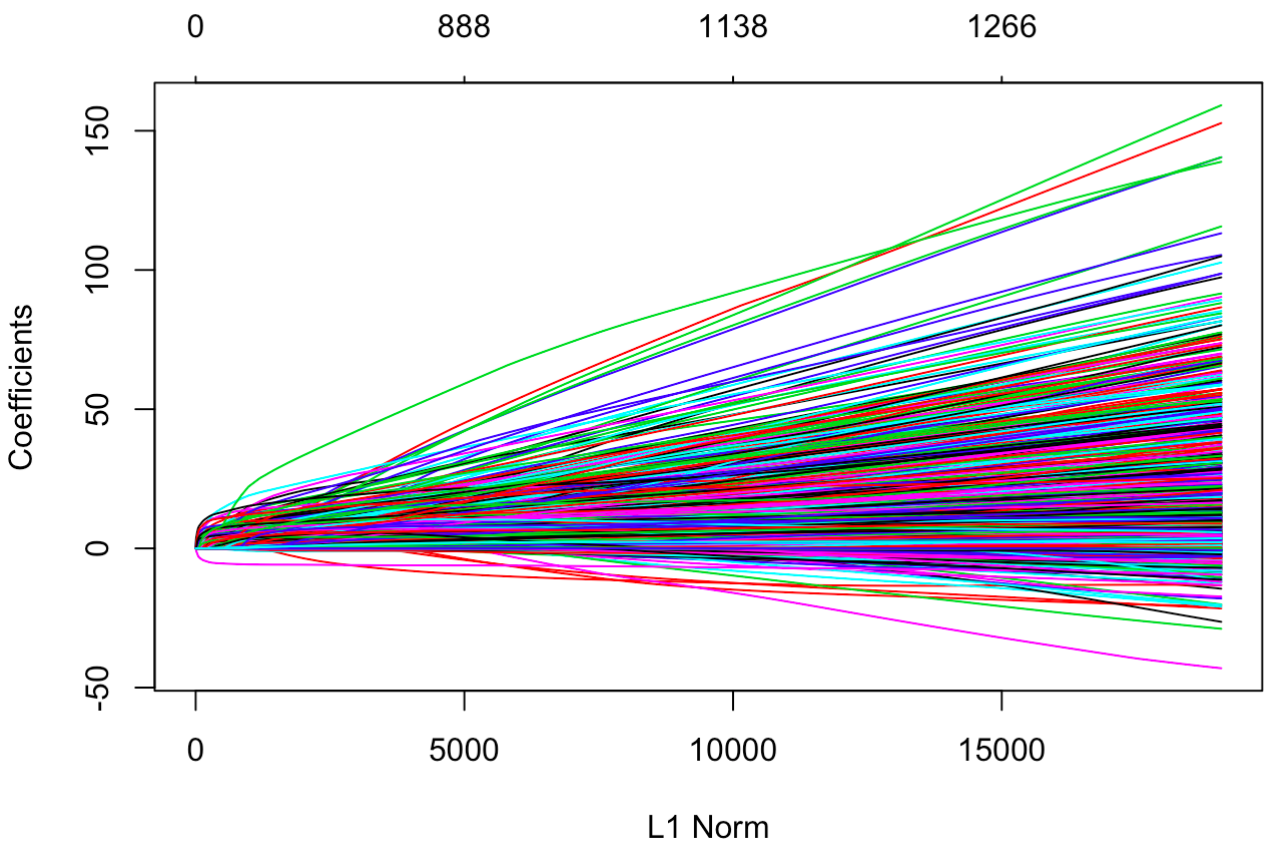

```
plot(glmnet_classifierG)
```



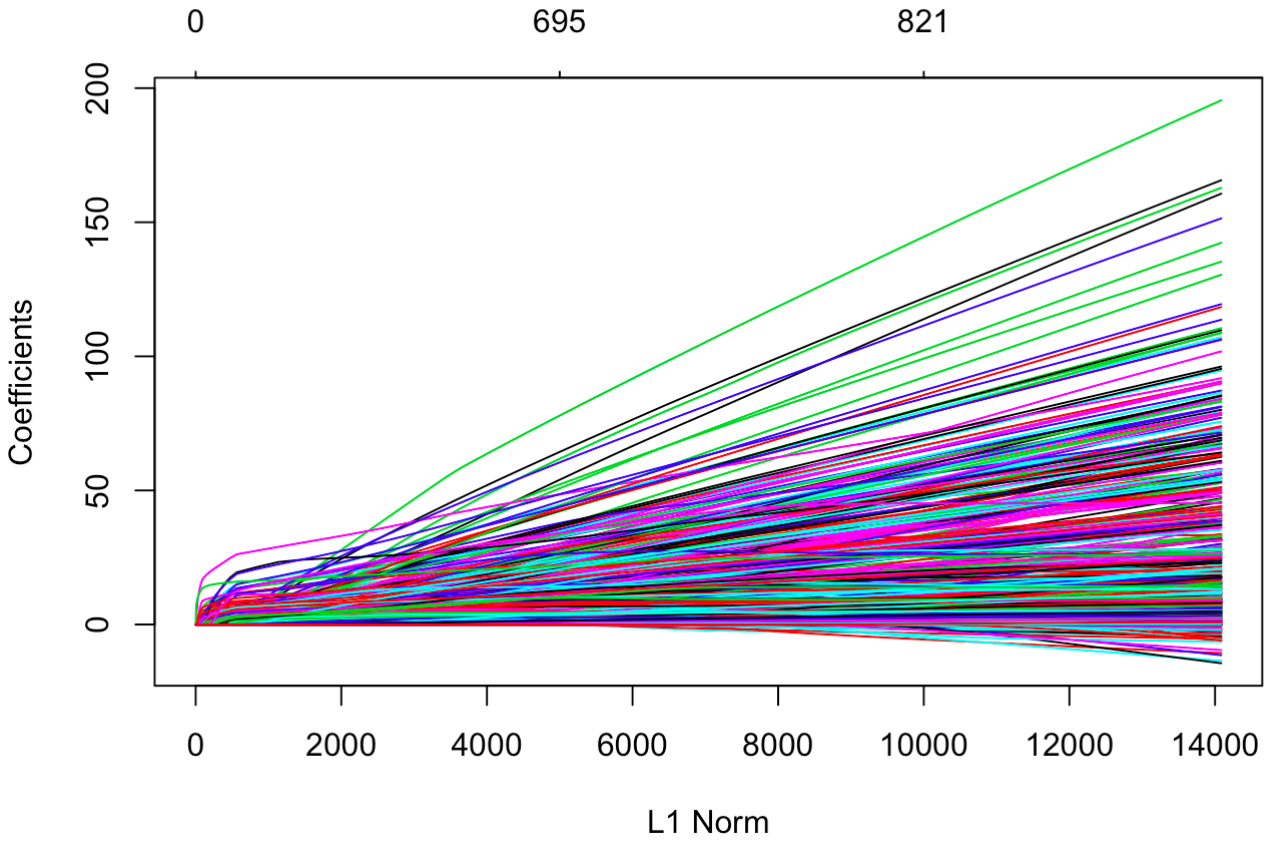

```
plot(glmnet_classifierH)
```



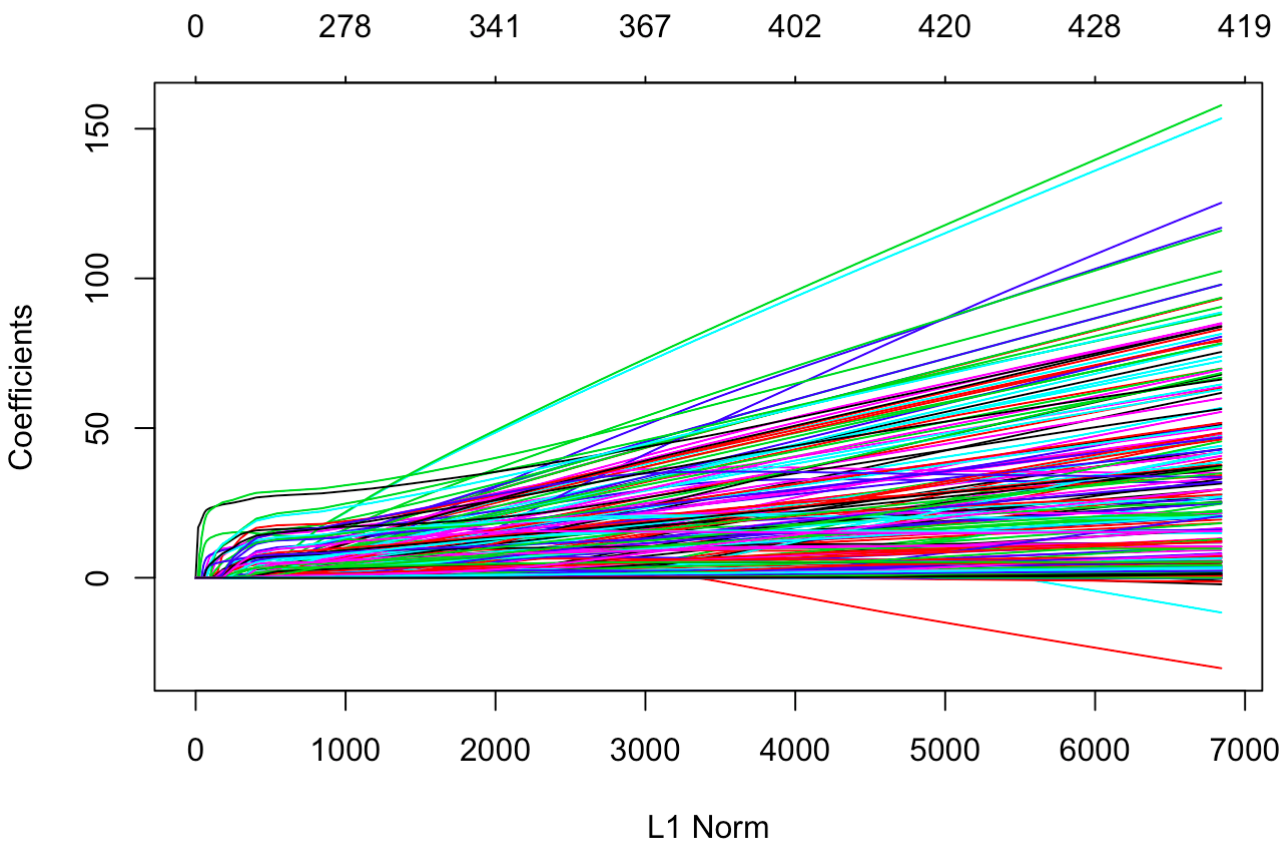

```
#coefficient performance over lambda
plot(glmnet_classifierA$glmnet.fit)
```



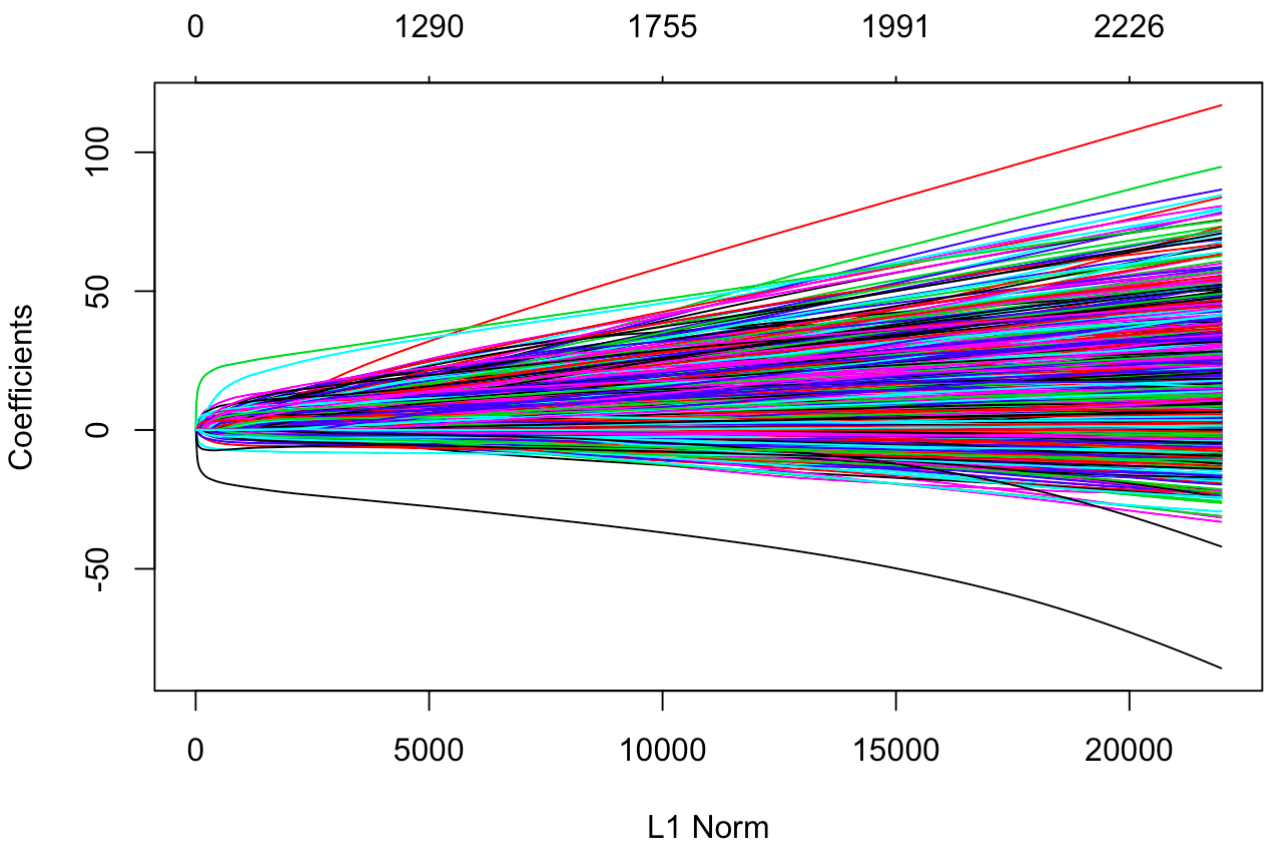

```
plot(glmnet_classifierB$glmnet.fit)
```



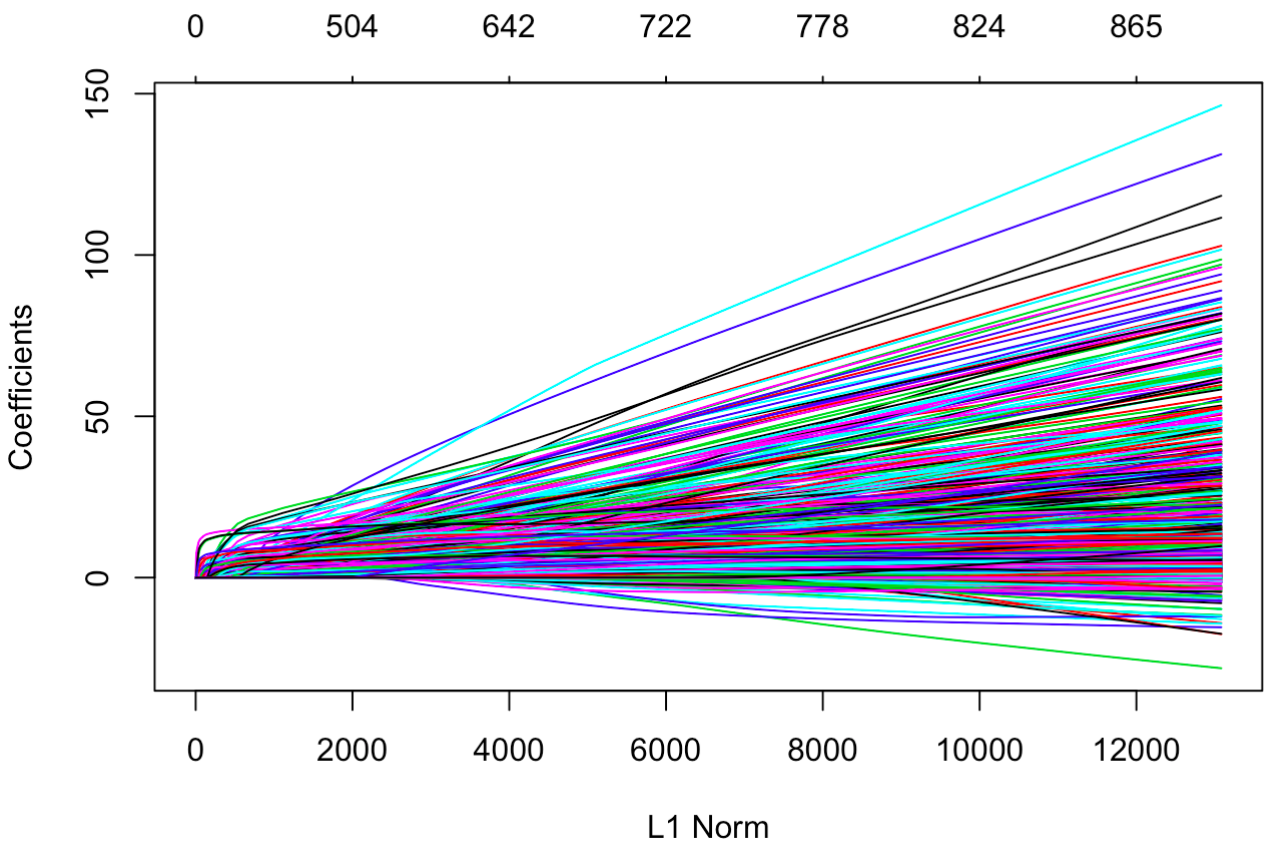

```
plot(glmnet_classifierC$glmnet.fit)
```



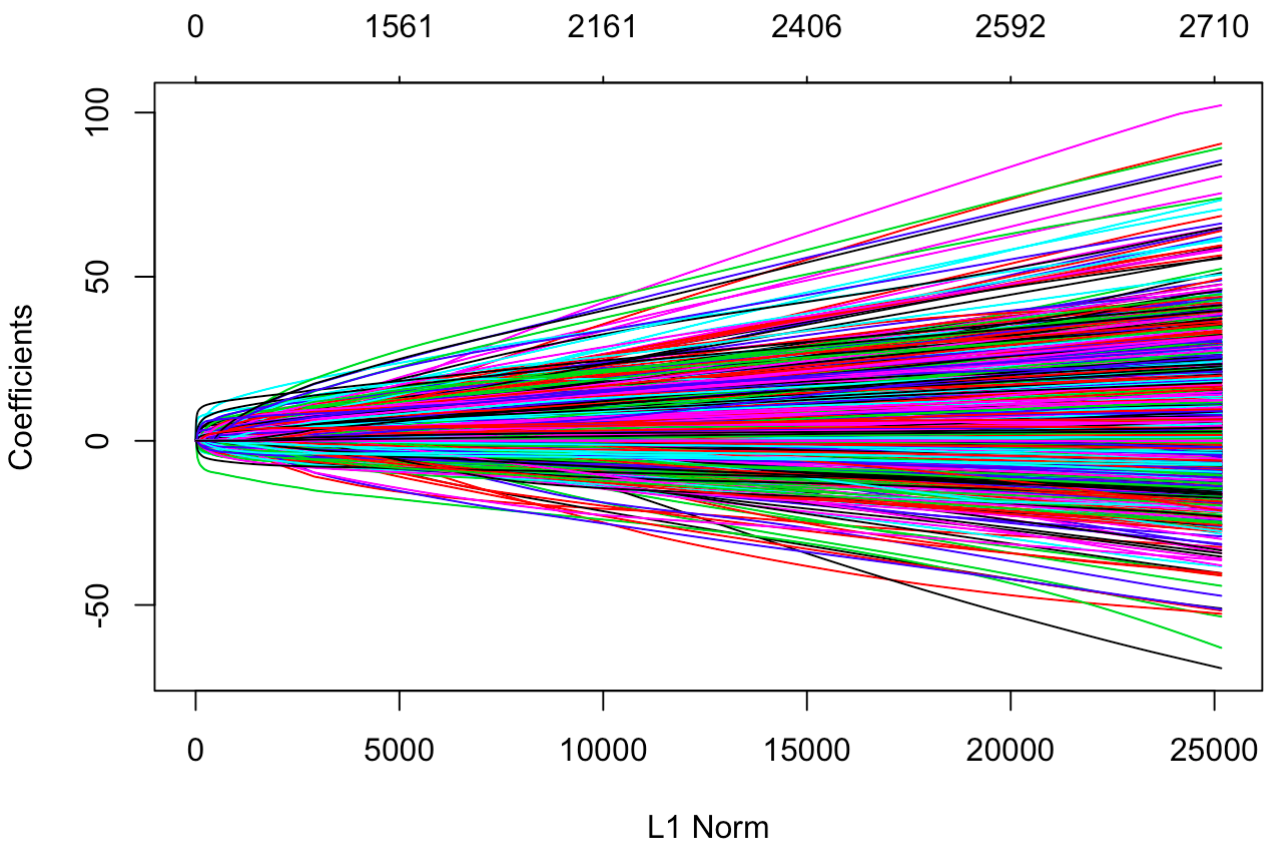

```
plot(glmnet_classifierD$glmnet.fit)
```



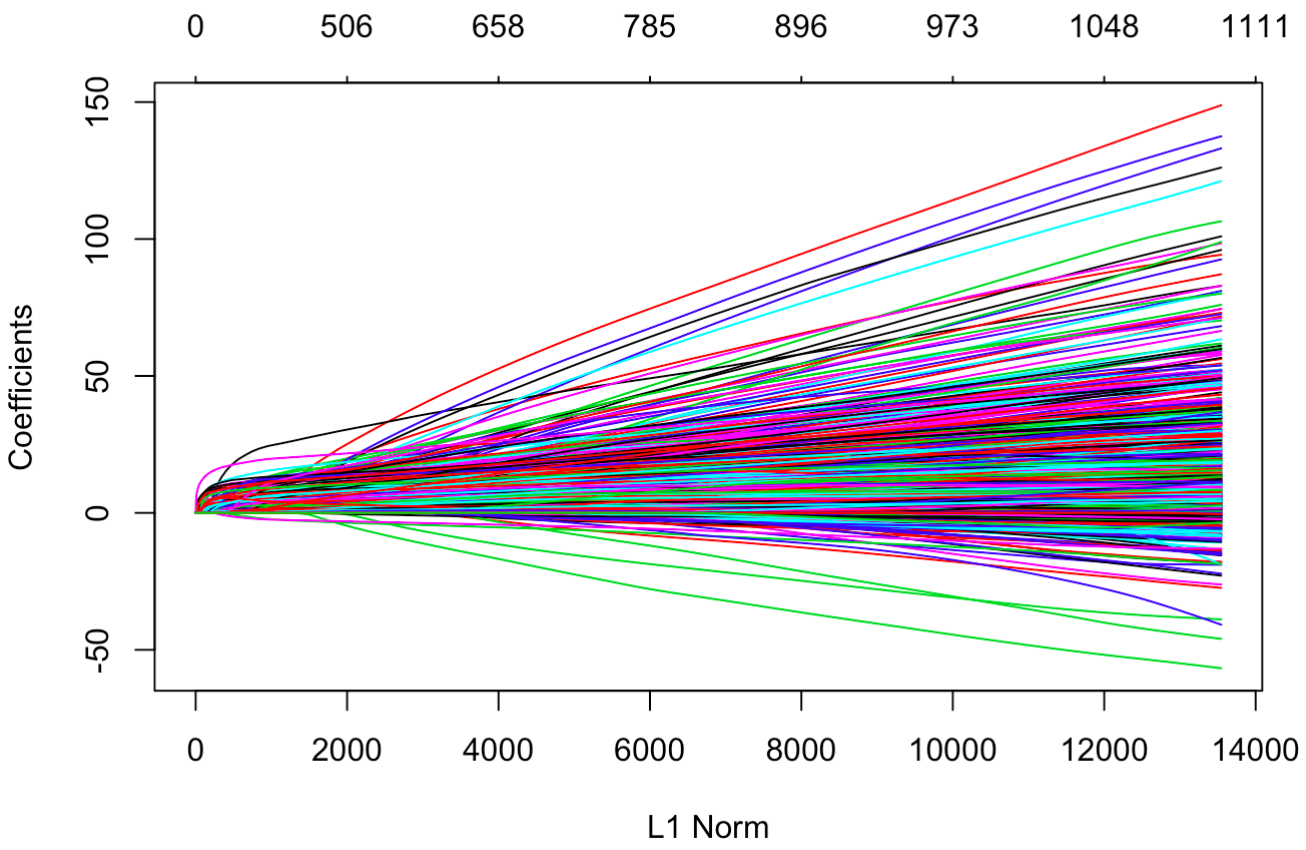

```
plot(glmnet_classifierE$glmnet.fit)
```



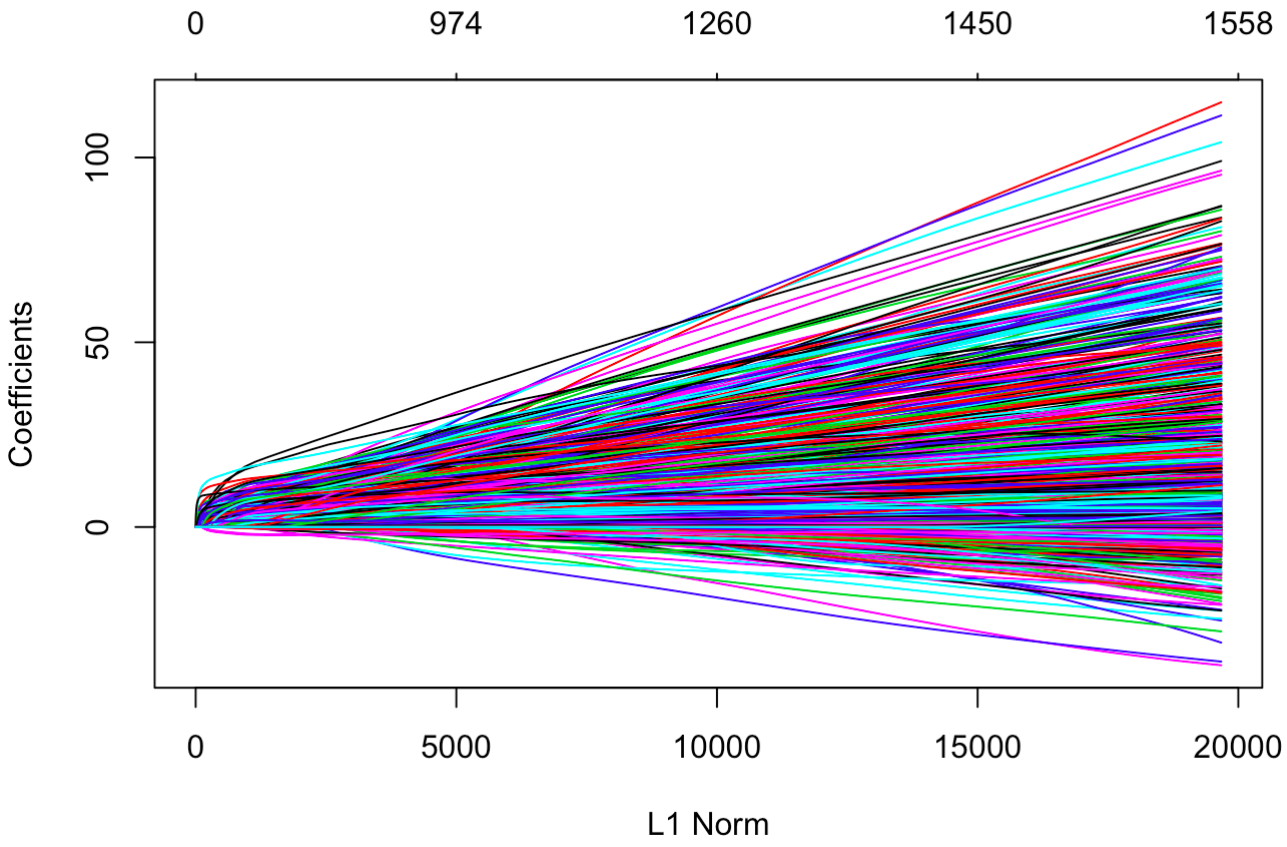

```
plot(glmnet_classifierF$glmnet.fit)
```




```
plot(glmnet_classifierG$glmnet.fit)
```




```
plot(glmnet_classifierH$glmnet.fit)
```




```
#predictions on test DTM
predsA = predict(glmnet_classifierA, dtm_test_tfidf, type = 'response')
predsB = predict(glmnet_classifierB, dtm_test_tfidf, type = 'response')
predsC = predict(glmnet_classifierC, dtm_test_tfidf, type = 'response')
predsD = predict(glmnet_classifierD, dtm_test_tfidf, type = 'response')
predsE = predict(glmnet_classifierE, dtm_test_tfidf, type = 'response')
predsF = predict(glmnet_classifierF, dtm_test_tfidf, type = 'response')
predsG = predict(glmnet_classifierG, dtm_test_tfidf, type = 'response')
predsH = predict(glmnet_classifierH, dtm_test_tfidf, type = 'response')
```

```
#accuracy
sprintf("Action genre classification accuracy is: %.3f percent", (1-classError(round(pre
dsA), as.matrix(test[,15]))$errorRate)*100)
```

```
## [1] "Action genre classification accuracy is: 81.480 percent"
```

```
sprintf("Adventure genre classification accuracy is: %.3f percent", (1-
classError(round(predsB), as.matrix(test[,16]))$errorRate)*100)
```

```
## [1] "Adventure genre classification accuracy is: 90.616 percent"
```

```
sprintf("Animation genre classification accuracy is: %.3f percent", (1-
classError(round(predsC), as.matrix(test[,17]))$errorRate)*100)
```

```
## [1] "Animation genre classification accuracy is: 94.141 percent"
```

```
sprintf("Comedy genre classification accuracy is: %.3f percent", (1-classError(round(pre
dsD), as.matrix(test[,18]))$errorRate)*100)
```

```
## [1] "Comedy genre classification accuracy is: 68.620 percent"
```

```
sprintf("Crime genre classification accuracy is: %.3f percent", (1-classError(round(pred
sE), as.matrix(test[,19]))$errorRate)*100)
```

```
## [1] "Crime genre classification accuracy is: 88.381 percent"
```

```
sprintf("Drama genre classification accuracy is: %.3f percent", (1-classError(round(pred
sF), as.matrix(test[,21]))$errorRate)*100)
```

```
## [1] "Drama genre classification accuracy is: 63.903 percent"
```

```
sprintf("Horror genre classification accuracy is: %.3f percent", (1-classError(round(pre
dsG), as.matrix(test[,25]))$errorRate)*100)
```

```
## [1] "Horror genre classification accuracy is: 87.090 percent"
```

```
sprintf("Thriller genre classification accuracy is: %.3f percent", (1-classError(round(p
redsH), as.matrix(test[,31]))$errorRate)*100)
```

```
## [1] "Thriller genre classification accuracy is: 80.288 percent"
```

CommentDuring my search for efficient tuning of alpha and lambda parameters, I came across this post for the glmnet method: https://stats.stackexchange.com/questions/69638/does-caret-train-function-for-glmnet-cross-validate-for-both-alpha-and-lambda?rq=1 (https://stats.stackexchange.com/questions/69638/does-caret-train-function-for-glmnet-cross-validate-for-both-alpha-and-lambda?rq=1)

Currently my settings on the model pasted below results in an infinite loop, but for future improvements I aim to configure this model to a working state because I think it is a good option via which both the alpha and lambda parameters can be tuned.

[/**Comment**]

```
suppressWarnings(suppressMessages(library(caret)))

eGrid <- expand.grid(.alpha = (1:10) * 0.1,
                     .lambda = (1:10) * 0.1)

Control <- trainControl(method = "repeatedcv", repeats = 10, classProbs =TRUE)


netFit <- train(x =as.matrix(dtm_train_tfidf), y = factor(as.matrix(train[,15]), labels
 = c("yes", "no")),
          method = "glmnet",
          tuneGrid = eGrid,
          trControl = Control)
```