Technical Report

Jon deVries, Rohan Buggana, Bhav Kurana

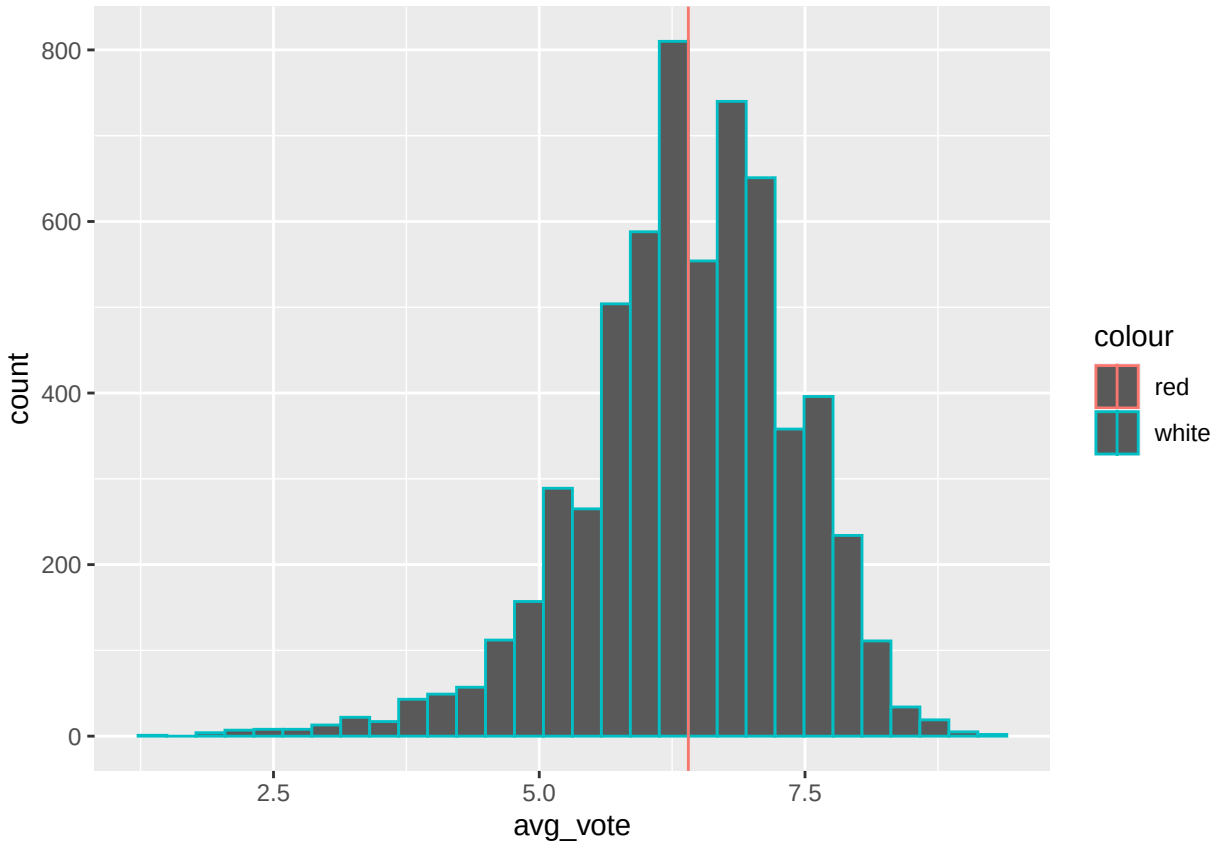
12/4/2021

Abstract We analyze a dataset of IMDb ratings and general information about the corresponding movies in order to predict the IMDb score, profit, and probablity of winning an oscar of upcoming movies. Using the stacks R package, we create three corresponding ensemble models comprised of linear regression, logistic regression, and knn models.

Introduction IMDb is an online database of information on movies, tv shows and other visual entertainment media. It offers a wide range of details on individual productions including user submitted ratings. Users rate individual movies on 10 and the website aggregates them into a single score: a movie's 'IMDb Rating'. Another measure of a movie's quality and cultural relevance are the renowned Academy Awards, popularly known as the Oscars. Each spring, the new releases of the year compete for individual awards in a number of categories, with only one movie winning each category, each year. A movie's IMDb rating and its Oscar status can be influential in determining viewership and the cultural consensus around a production. *Thus, we are interested in best predicting IMDB Ratings, the probability of winning an Oscar statuses and profits for new and as of yet unreleased movies.*

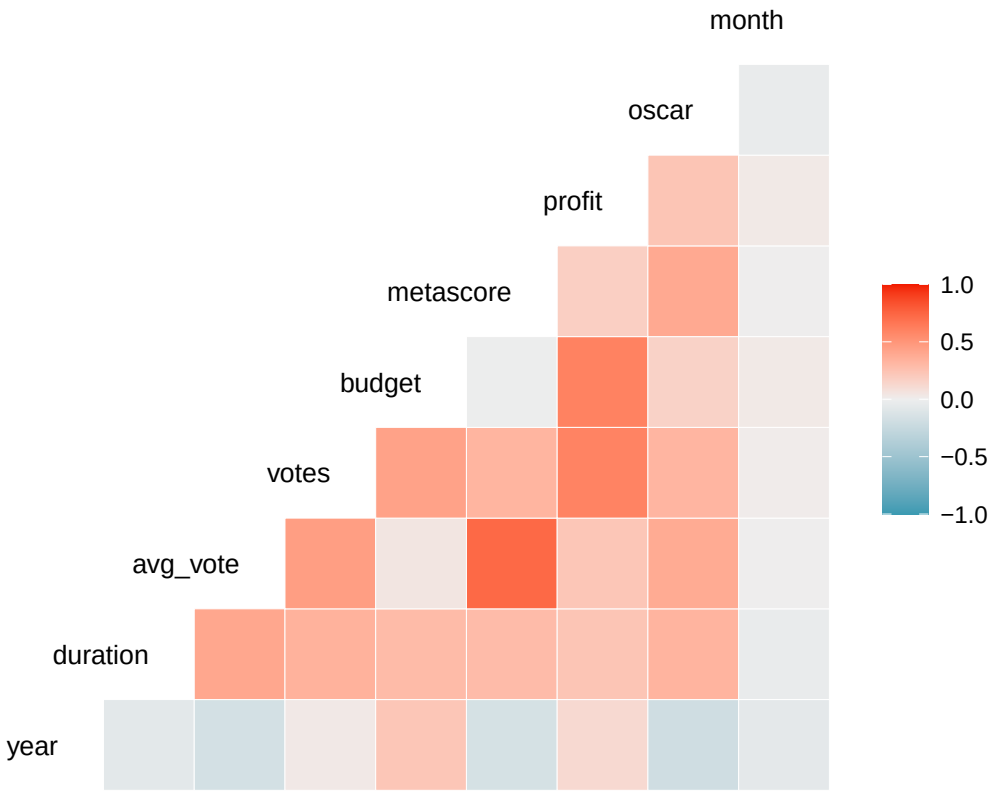
Methods The dataset we are using is the 'IMDb movies extensive dataset', accessed via kaggle. It describes many aspects of individual movies, along with their IMDb score, an average of ratings submitted to IMDb by users. The dataset contains 85,855 observations and 22 variables, where each observation describes an individual movie. In addition to this dataset, we've used the The 'The Oscar Award, 1927 - 2020' dataset to add to the IMDb dataset, a variable that describes whether or not each movie has won an Oscar (in any category).

Data Exploration Beyond adding the `oscar` variable we've performed multiple steps to wrangle and clean the data: - Removed movies who's budget was given in currencies other than the United States Dollar. - Created the `profit` variable from the existing income and budget variables. - Reduced the number of levels in `writer`, `director` and `production company` by setting all but the 30 most frequently appearing writers, directors and production companies to other. - Reduced the number of levels in the `genre`, `country` and `language` variables by selecting only the first of the multiple, comma separated words in each cell (ex: Action, Adventure, Comedy becomes Action). - Created the `years` and `month` variables from the `date_published` variable that was in year-month-day format. - Dropped `imdb_title_id`, `title`, `year`, `date_published`, `description`, `usa_gross_income`, `worldwide_gross_income`, `actors`, `metascore`, `reviews_from_users`, `reviews_from_critics` and `description` variables due to redundancies, usability and irrelevance to our models. - Dropped NAs



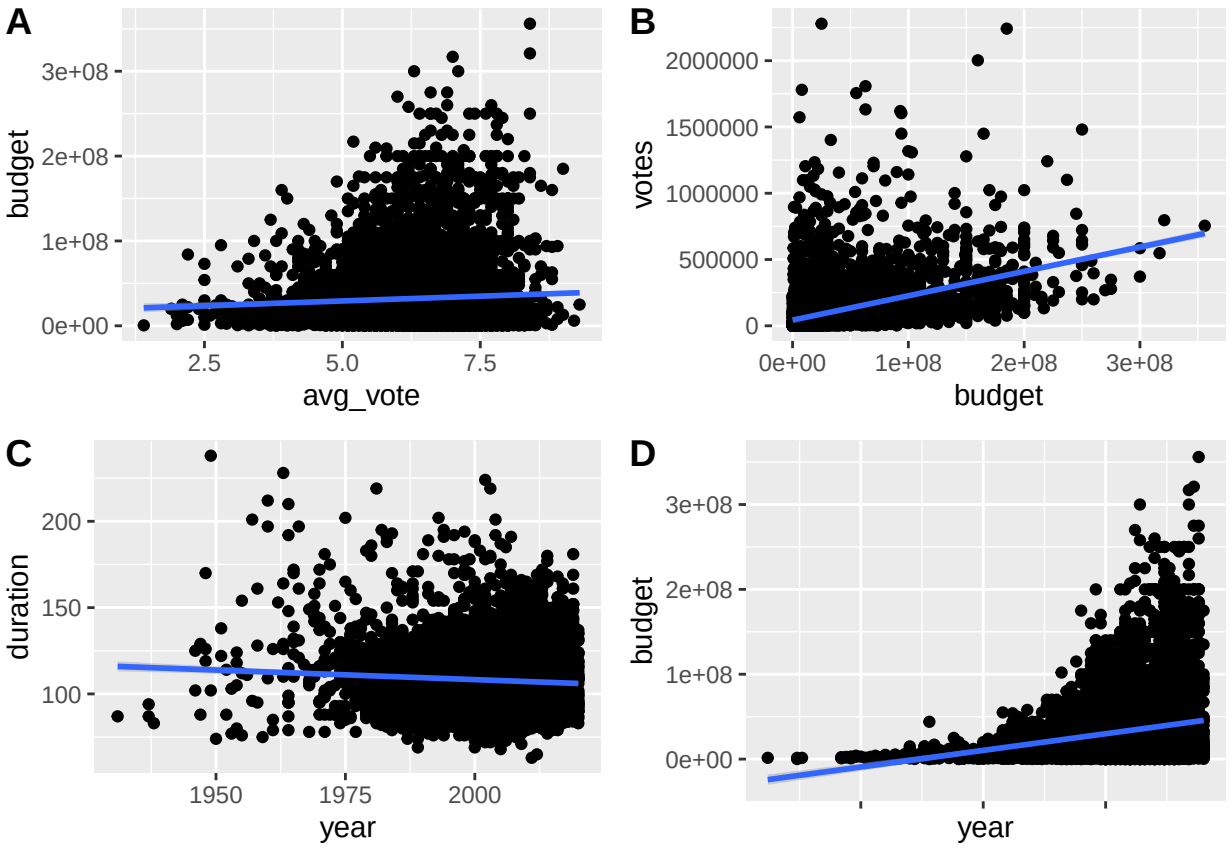
The histogram above shows a distribution of our response variable, IMDb score, encoded as avg_vote in our dataset. The distribution is unimodal, with its peak approximately at the mean. The distribution is fairly normal, with a long left tail. The mean of avg_vote is 6.4 and its median is 6.5.

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```




```
##                 [,1]
## year      -0.16815005
## duration   0.40803425
## avg_vote   1.00000000
## votes      0.45693599
## budget     0.05499919
## metascore  0.73346194
## profit     0.23017274
## oscar      0.38613264
## month             NA
```

This graphic is a correlation matrix between all the quantitative variables in the dataset. Focusing on our response variable, avg_vote, we can see that, there are weak correlations between a movie's imdb score and its duration, the number of votes it receives on IMDb, the number of reviews it receives, both from critics and users, the profit it makes and the year it was published. Both the appropriateness of these correlations, the direction of potential causality and the extent to which some of these variables are proxies for others should be examined further.



These graphics are scatterplots. They show the distribution of one numeric variable against the other, along with a smooth, flexible line of best fit. Understanding the shape of the distribution and the best fit line can help us get context behind the correlation coefficients plotted in the matrix above.

Results After running linear models on all the response variables - average vote, profit, Oscar - we find that all the quantitative variables are very statistically significant at the 5% significance level so we make use of them in the main prediction models. On the other hand, the p-values of the categorical variables tell us that only some of the levels are significant - for example, when we use oscars as a response we find that director Steven Spielberg has a significant p-value but we cannot just select only Steven Spielberg movies to analyze and make predictions on. The same thing happens when we try to perform best subset selection with the "genre" predictor. Since we need to keep our model as broad as possible without sacrificing usability to achieve the goal we set we cannot get rid of statistically insignificant levels.

We build an ensemble model via stacks. In general, ensemble models are significantly more flexible and efficient than a single model. Additionally, it is more resilient against model building bias. However, ensemble models are also computationally expensive to make predictions with, while favoring models with low test time. Furthermore, we can observe diminishing returns on the number of models that can be incorporated in ensemble. Here, we use four models in ensemble, for regression:

Multiple linear regression, which is an extension of ordinary least squares regression (OLS) that involves one response and more than one predictors. OLS is a method for estimating the unknown coefficients in the linear regression model. It does this by minimizing the RSS to best fit a straight line to the data.

K nearest neighbours, for regression, stores all known cases and then predicts the response based a measure of similarity. For regression, this is a distance function.

LASSO, or least absolute shrinkage and selection operator, is used here as a method of penalized regression. One can think of LASSO as solving a restricted optimization for minimizing RSS.

Random Forests artificially increases the variety among trees by randomly restrict which predictors can be used at each split point. This increases accuracy of the ensemble by breaking correlations among the participant trees therefore reducing variance.

In addition, 10-fold cross validation was performed for each model–models were fit.

```
imdb_rmse_train
```

```
## # A tibble: 9 x 4
##   .metric .estimator .estimate member
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       0     avg_vote
## 2 rmse    standard       0.653 .pred
## 3 rmse    standard       0.100 knn_fit_1_01
## 4 rmse    standard       0.756 knn_fit_1_15
## 5 rmse    standard       0.763 knn_fit_1_16
## 6 rmse    standard       0.769 knn_fit_1_17
## 7 rmse    standard       0.847 lm_fit_1_1
## 8 rmse    standard       0.852 lasso_fit_1_01
## 9 rmse    standard       0.499 rf_fit_1_1
```

```
imdb_rmse_test
```

```
## # A tibble: 9 x 4
##   .metric .estimator .estimate member
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       0     avg_vote
## 2 rmse    standard       0.800 .pred
## 3 rmse    standard       1.13  knn_fit_1_01
## 4 rmse    standard       0.831 knn_fit_1_15
## 5 rmse    standard       0.831 knn_fit_1_16
## 6 rmse    standard       0.831 knn_fit_1_17
## 7 rmse    standard       0.827 lm_fit_1_1
## 8 rmse    standard       0.827 lasso_fit_1_01
## 9 rmse    standard       0.813 rf_fit_1_1
```

```
data.frame(imdb_new_preds$original_title,imdb_new_preds$.pred)
```

```
##    imdb_new_preds.original_title imdb_new_preds..pred
## 1         Spider-Man: No Way Home             7.081823
## 2       The Matrix Resurrections             7.235362
## 3                 The King's Man             6.881071
## 4                          Sing 2             7.233814
## 5            A Journal for Jordan             7.001626
## 6                          Scream             6.285981
## 7                 House of Gucci             7.705650
## 8                     The Humans             6.368667
## 9       Clifford the Big Red Dog             6.084420
## 10                         Cyrano             6.866720
```

The above data frame presents our prediction for the IMDb scores of new and upcoming movies. Furthermore, the ensemble model produced an RMSE of 0.653 on the training data, and an RMSE of 0.800 on the test data. Given that IMDb scores average ~7, this is not an ideal, but still an acceptable degree of error. The above dataframe call lists our predictions for upcoming movies' IMDb scores.

```
oscar_rmse_train
```

```
## # A tibble: 8 x 4
##   .metric .estimator .estimate member
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       0     oscar
## 2 rmse    standard       0.274 .pred
## 3 rmse    standard       0.318 knn_fit_1_19
## 4 rmse    standard       0.320 knn_fit_1_20
## 5 rmse    standard       0.321 knn_fit_1_21
## 6 rmse    standard       0.344 lm_fit_1_1
## 7 rmse    standard       0.351 lasso_fit_1_01
## 8 rmse    standard       0.208 rf_fit_1_1
```

```
oscar_rmse_test
```

```
## # A tibble: 9 x 4
##   .metric .estimator .estimate member
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard        0    oscar
## 2 rmse    standard        6.22 .pred
## 3 rmse    standard        6.25 knn_fit_1_01
## 4 rmse    standard        6.19 knn_fit_1_15
## 5 rmse    standard        6.19 knn_fit_1_16
## 6 rmse    standard        6.18 knn_fit_1_17
## 7 rmse    standard        6.22 lm_fit_1_1
## 8 rmse    standard        6.22 lasso_fit_1_01
## 9 rmse    standard        6.21 rf_fit_1_1
```

```
data.frame(oscar_new_preds_regression$original_title,oscar_new_preds_regression$.pred)
```

```
##    oscar_new_preds_regression.original_title oscar_new_preds_regression..pred
## 1                     Spider-Man: No Way Home                         0.44875859
## 2                    The Matrix Resurrections                         0.46098123
## 3                              The King's Man                         0.21874268
## 4                                    Sing 2                         0.36499084
## 5                        A Journal for Jordan                         0.32092190
## 6                                    Scream                         0.10274329
## 7                              House of Gucci                         0.56684802
## 8                                  The Humans                         0.13148537
## 9                    Clifford the Big Red Dog                         0.06823856
## 10                                   Cyrano                         0.23779270
```

The above data frame presents our prediction for the probability that each of these new and upcoming movies wins an oscar in any category. Furthermore, the ensemble model produced an RMSE of 0.274 on the training data, and an RMSE of 6.22 on the test data. Given that the probabilities range from 0 - 1, this is, quite frankly, horrible! And given that the training rmse is so much lower than the test rmse, our model

is extremely overfit on the oscars data. This is strange, given that we attempted cross-validation, but also given the limited size of the oscar dataset, along with our faliure to use a classification model (instead we used regression,) this is not overly surprising. Although likely highly flawed, the probablity of winning an oscar (or something akin to this, maybe best interpreted in a relative sense) for upcoming movies is displayed in the above dataframe.

```
profit_rmse_train
```

```
## # A tibble: 7 x 4
##   .metric .estimator  .estimate member
##   <chr>   <chr>           <dbl> <chr>
## 1 rmse    standard           0  profit
## 2 rmse    standard    62229765. .pred
## 3 rmse    standard    38584570. knn_fit_1_02
## 4 rmse    standard    56994836. knn_fit_1_03
## 5 rmse    standard   110949690. lm_fit_1_1
## 6 rmse    standard   119531550. lasso_fit_1_47
## 7 rmse    standard    61848040. rf_fit_1_1
```

```
profit_rmse_test
```

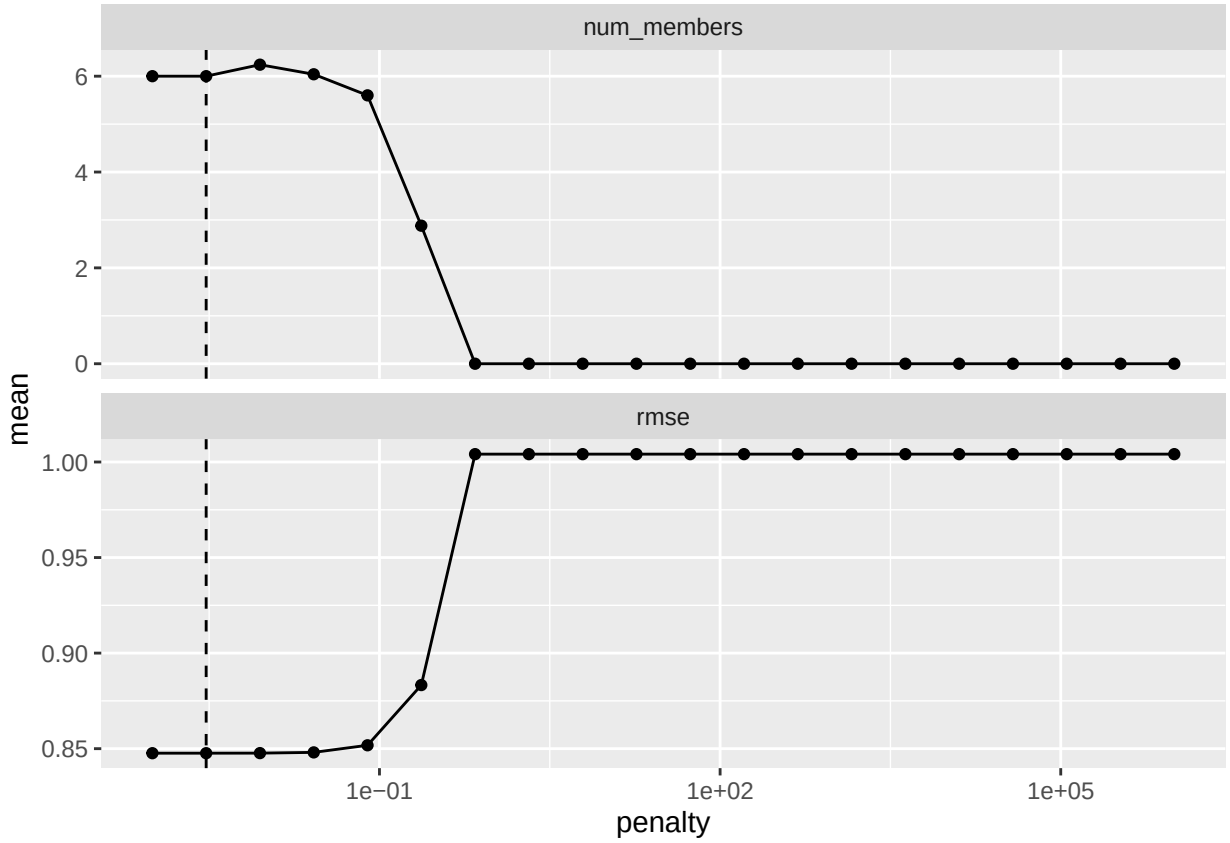
```
## # A tibble: 7 x 4
##   .metric .estimator  .estimate member
##   <chr>   <chr>           <dbl> <chr>
## 1 rmse    standard           0  profit
## 2 rmse    standard   128330648. .pred
## 3 rmse    standard   149330379. knn_fit_1_02
## 4 rmse    standard   143331847. knn_fit_1_03
## 5 rmse    standard   132136783. lm_fit_1_1
## 6 rmse    standard   136102508. lasso_fit_1_47
## 7 rmse    standard   128968644. rf_fit_1_1
```

```
data.frame(profit_new_preds$original_title,profit_new_preds$.pred)
```

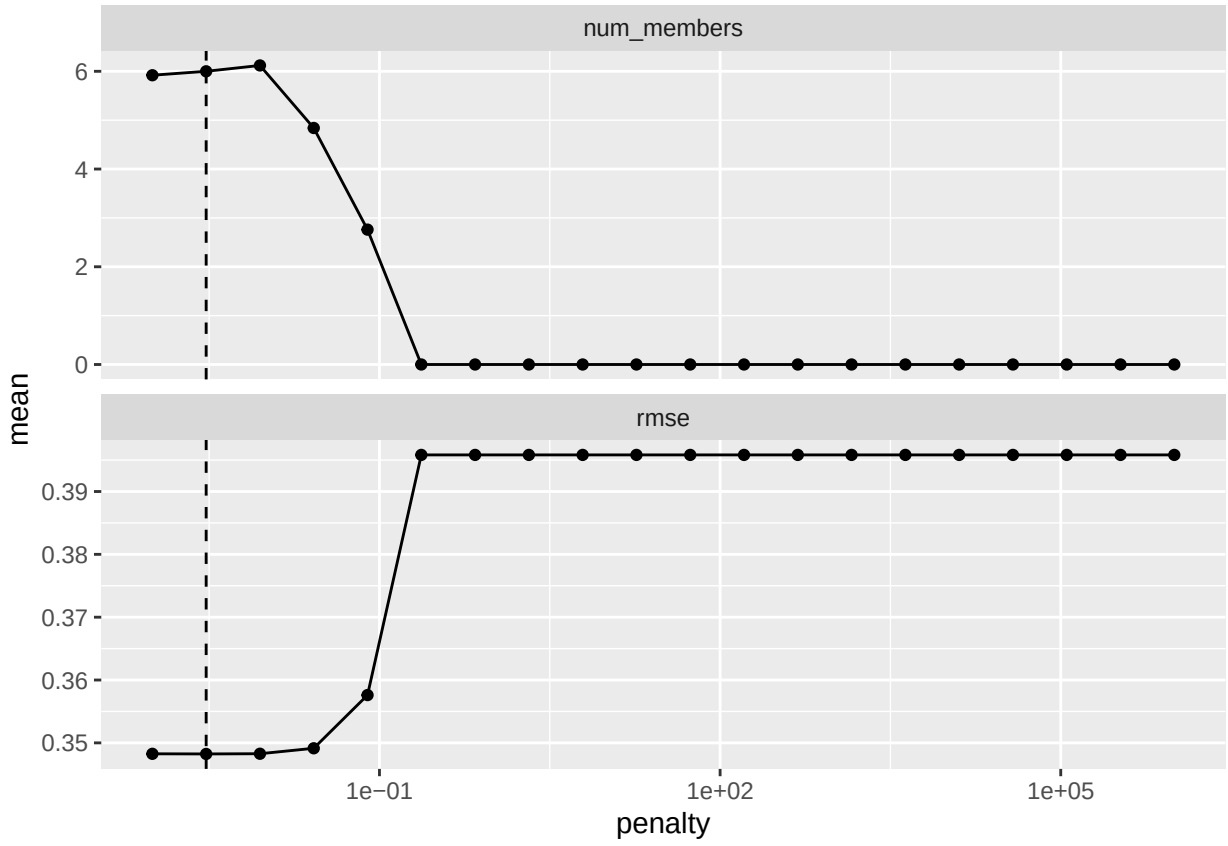
```
##    profit_new_preds.original_title profit_new_preds..pred
## 1          Spider-Man: No Way Home              580948278
## 2         The Matrix Resurrections              594762339
## 3                   The King's Man              333766306
## 4                            Sing 2              322812167
## 5             A Journal for Jordan              163465743
## 6                            Scream              137474371
## 7                   House of Gucci              325235214
## 8                       The Humans               78457994
## 9         Clifford the Big Red Dog              162732623
## 10                           Cyrano               97323801
```

For the profit response variable, the ensemble model produced an RMSE of 62229765 on the training data, and an RMSE of 128330648 on the test data. Again, givent he much higher test RMSE, this indicates that our model is highly variable and is overfit. The predicted profit for 10 unreleased movies are displayed above.

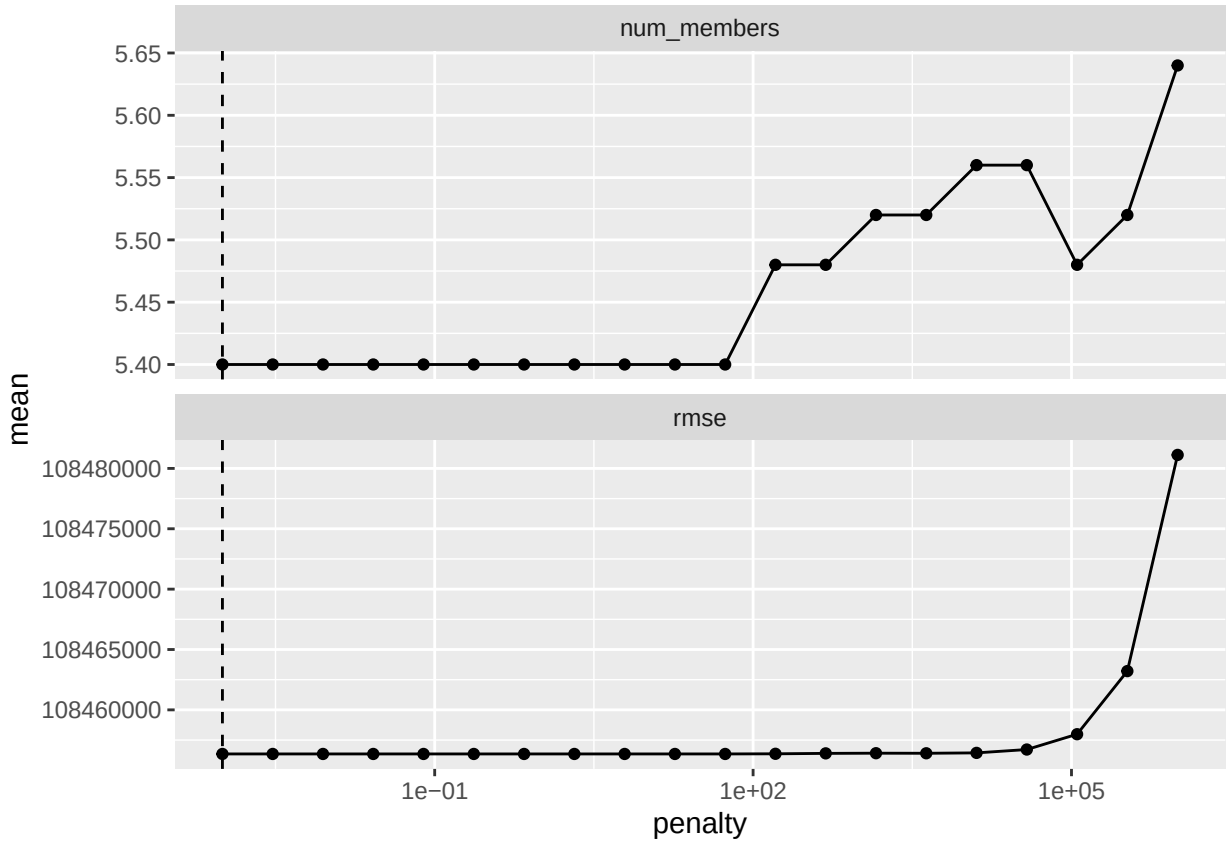
```
imdb_penalty
```




```
oscar_penalty
```




```
profit_penalty
```



The above plot depicts the effects of LASSO penalty on the mean prediction value, for all three models that we produced.

Discussion Our main objective, to perform inference and prediction on both IMDb scores, Oscars and profits, yielded us the above results. H O W E V E R.

There are some very concerning limitations. As mentioned above, the test rmse for the Oscars prediction is over an order of magnitude worse than that of the training rmse. This suggests overfitting to a massive degree, even though we performed cross validation, which–in theory–should have helped minimize this. Overfitting was an issue for all models as evidenced by the train rmse being significantly lower than and test rmse in all cases. In short, our models have low bias and high variance, to different degrees for each model. It is quite possible that this is a result of the high flexibility of our ensemble model. Possibly something was incorrect with how we implemented cross-validation. We also could have used additional training data–certainly for the oscars set, or attempted to reduce the number of features in our model. Reducing features could have been done by clustering directors, production companies, and so on.

We have also not considered the presence of outliers and leverage points. These may bias the training model which worsens performance on the test set.

Another limitation is that we didn't have as many predictors we would've liked. Information such as actors, features of the plot, social media interaction and whether or not the movie is a part of a series or franchise is all likely to have predictive power. All of this information is missing due to challenges in acquiring or wrangling data for suitable use in our models. This is reflected in the relatively weak correlations in our data exploration section.

Additionally, our `oscar` variable is such that each movie is coded as having won an oscar regardless of the category it won the award for. This was done mostly because selecting any one category would've drastically reduced the number of winners in our dataset since there is only one winner in each category each year. It makes sense that different traits make a movie more likely to win different categories of Oscars, so the way

the data is coded is likely creating noise and weakening the predictive power of our model. Furthermore, Oscars are even more hard to predict because of changing tastes and modern Oscar marketing campaigns that have large effects on outcomes.

We also likely lost valuable data while adding the Oscar data to the IMDb set, since anything other than an exact match in titles would've resulted in either a misclassifaction as an oscar loser, or the removal of the row from the dataset.

Finally, our profit data is definitely skewed by inflation, as evidenced by the extremely strong predictive power `year` has over it.

Refrences @Manual{, title = {stacks: Tidy Model Stacking}, author = {Simon Couch and Max Kuhn}, year = {2021}, note = {R package version 0.2.1}, url = {https://CRAN.R-project.org/package=stacks}, } https://CRAN.R-project.org/package=stacks https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset https://www.kaggle.com/unanimad/the-oscar-award

Code Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(parsnip)
library(recipes)
library(stacks)
library(rsample)
library(yardstick)
library(tune)
library(tidymodels)
library(workflows)
library(ggpubr)
library(GGally)
```

#Process IMDb data, add in Oscars column

```
# imdb_na is the complete data set
imdb_na <- read_csv("./IMDb/IMDb movies.csv")

# imdb is the data set without NAs
imdb <- na.omit(imdb_na)

# removing "$" from USA gross income, Worldwide gross income, budget to make them quantitative again
imdb$usa_gross_income <- as.numeric(gsub("\\$", "", imdb$usa_gross_income))
imdb$worlwide_gross_income <- as.numeric(gsub("\\$", "", imdb$worlwide_gross_income))
imdb$budget <- as.numeric(gsub("\\$", "", imdb$budget))
imdb <- na.omit(imdb)

#change varible names
imdb <- rename(imdb, world_income = worlwide_gross_income,
         usa_income = usa_gross_income)

# Adding column "profit" to the data set
profit <- imdb$world_income - imdb$budget
imdb <- data.frame(imdb, profit)

# Encode all non-top 30 (by frequency) writers, directors, and production companies
df1 <- data.frame(table(imdb$writer))
df1 <- head(df1[order(-df1$Freq),] ,30)
```

```
for(j in 1:length(imdb$writer)) {
  if(!(imdb$writer[j] %in% df1$Var1)) {
     imdb$writer[j] = 'other'}
  #else{print(imdb$writer[j])  }
  }

df1 <- data.frame(table(imdb$director))
df1 <- head(df1[order(-df1$Freq),] ,30)
for(j in 1:length(imdb$director)) {
  if(!(imdb$director[j] %in% df1$Var1)) {
     imdb$director[j] = 'other'}}

df1 <- data.frame(table(imdb$production_company))
df1 <- head(df1[order(-df1$Freq),] ,30)
for(j in 1:length(imdb$production_company)) {
  if(!(imdb$production_company[j] %in% df1$Var1)) {
     imdb$production_company[j] = 'other'}}

#read in oscars
oscars_na <- read_csv("./oscars/the_oscar_award.csv")
imdb['oscar'] <- 0
imdb$genre <- gsub(",.*", "",imdb$genre)
imdb$country <- gsub(",.*", "",imdb$country)
imdb$language <- gsub(",.*", "",imdb$language)

#match movies in oscar dataset to movies in imdb dataset
for(i in 1:length(imdb$original_title)){
    if(imdb$original_title[i] %in% oscars_na$film){
      imdb[['oscar']][i] <- 1
    }
}

#split date_published to year and month columns
imdb['year'] <- NA
imdb['month'] <- NA

for (x in 1:length(imdb$date_published)){
  imdb$year[x] <- as.numeric(substr(imdb[['date_published']][[x]],0,4))
  imdb$month[x] <- as.numeric(substr(imdb[['date_published']][[x]],6,7))
}

imdb_dropped <- imdb[ -c(1, 2, 3, 5, 13, 14, 18, 19, 21 ,22) ]
```

```
dist <- ggplot(imdb_dropped, aes(x = avg_vote, color = "white")) +
  geom_histogram(bins = 30) +
  geom_vline(aes(xintercept = mean(avg_vote), color = "red"))

mean(imdb_dropped$avg_vote)
median(imdb_dropped$avg_vote)

corr1 <- ggcorr(imdb_dropped, hjust = 1, size = 3.5)

corr2 <- imdb_dropped %>%
```

```
  select_if(is.numeric) %>%
  cor(imdb$avg_vote)

sct1 <- ggplot(imdb_dropped, aes(x = avg_vote, y = budget)) +
  geom_point() + geom_smooth(method = "lm")

sct2 <- ggplot(imdb_dropped, aes(x = budget, y = votes)) +
  geom_point() + geom_smooth(method = "lm")

sct3 <- ggplot(imdb_dropped, aes(x = year, y = duration)) +
  geom_point() + geom_smooth(method = "lm")

sct4 <- ggplot(imdb_dropped, aes(x = year, y = budget)) +
  geom_point() + geom_smooth(method = "lm")

scatter <- ggarrange(sct1, sct2, sct3, sct4 + rremove("x.text"),
          labels = c("A", "B", "C", "D"),
          ncol = 2, nrow = 2)
```

```
#test set, training set
set.seed('333')
imdb_init <- initial_split(imdb, prop = 3/4)
imdb_train <- training(imdb_init)
imdb_test <- testing(imdb_init)
```

Logistic Regression for oscar score

```
# build logistic model for Oscars
imdb_log <- glm(oscar~country+genre+budget+month+production_company+director+year+language,data=imdb_tra
```

###IMDb Model Building w/ stacks ################################## Recipe for IMDb

```
imdb_vote_rec <-
  recipe(avg_vote~writer+genre+budget+duration+director+year+production_company,data = imdb_train) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  #step_unknown()%>%
  step_zv(all_predictors())%>%
  step_normalize(all_numeric(), -all_outcomes())
```

10-fold CV

```
folds <- vfold_cv(imdb_train, v = 10)
metric <- metric_set(rmse)
ctrl_grid <- control_stack_grid()
ctrl_res <- control_stack_resamples()
```

KNN

```
knn_mod <- nearest_neighbor(
  mode = "regression",
  neighbors = tune("k")) %>%
  set_engine("kknn")


knn_wf<- workflow() %>%
  add_model(knn_mod) %>%
  add_recipe(imdb_vote_rec)

#remember, k values should be odd
knn_grid <- data.frame(k = 1+2*0:20)

knn_fit<- knn_wf %>% tune_grid(
  resamples = folds,
  metrics = metric,
  grid = knn_grid,
  control = ctrl_grid)
```

Linear Model

```
lm_mod <- linear_reg() %>% set_engine("lm")

lm_wf<-workflow() %>%
  add_model(lm_mod) %>%
  add_recipe(imdb_vote_rec)

lm_fit <- lm_wf %>%
  fit_resamples(resamples = folds,
  metrics = metric,
  control = ctrl_res)
```

lasso

```
lasso_mod <- linear_reg(
  mode = "regression",
  penalty = tune("lambda")) %>%
  set_engine("glmnet")

lasso_wf<- workflow() %>%
  add_model(lasso_mod) %>%
  add_recipe(imdb_vote_rec)

lasso_grid <- data.frame(lambda = 10^seq(-2, 8, length = 50))

lasso_fit<- lasso_wf %>% tune_grid(
  resamples = folds,
  grid = lasso_grid,
  control = ctrl_grid)
```

random forest

```
rf_mod <- rand_forest(mode = "regression", trees = 30) %>%
  set_engine("randomForest")

rf_wf <- workflow() %>%
  add_model(rf_mod) %>%
  add_recipe(imdb_vote_rec)

rf_fit <- rf_wf %>%
  fit_resamples(resamples = folds,
    metrics = metric,
    control = ctrl_res)
```

stack the stacks-IMDb

```
imdb_st <- stacks() %>%
  add_candidates(knn_fit) %>%
  add_candidates(lm_fit) %>%
  add_candidates(lasso_fit) %>%
  add_candidates(rf_fit)
```

stack the stacks

```
stacks_grid <- 10^seq(-3, 6, length = 20)
imdb_st_blend <- imdb_st %>%
  blend_predictions(penalty = stacks_grid, metric = metric)
imdb_en_fit<- imdb_st_blend %>% fit_members()
imdb_preds<- imdb_train %>% bind_cols(predict(imdb_en_fit, .))
member_preds <- imdb_preds %>% select(avg_vote) %>%
  bind_cols(predict(imdb_en_fit, imdb_train, members = T))
imdb_rmse_train <- map_dfr(member_preds, rmse, truth = avg_vote, data = member_preds) %>%
mutate(member = colnames(member_preds))
```

```
imdb_preds<- imdb_test %>% bind_cols(predict(imdb_en_fit, .))
member_preds <- imdb_preds %>% select(avg_vote) %>%
  bind_cols(predict(imdb_en_fit, imdb_test, members = T))
imdb_rmse_test <- map_dfr(member_preds, rmse, truth = avg_vote, data = member_preds) %>%
mutate(member = colnames(member_preds))
```

```
imdb_new <- read.csv('new_releases.csv')
imdb_new_preds <- imdb_new %>% bind_cols(predict(imdb_en_fit, .))
```

```
imdb_penalty <- autoplot(imdb_st_blend)
```

#Profit model building ##

```
profit_vote_rec <-
  recipe(profit~writer+genre+budget+duration+director+year+production_company,data = imdb_train) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  #step_unknown()%>%
  step_zv(all_predictors())%>%
  step_normalize(all_numeric(), -all_outcomes())
```

10-fold CV

```
folds <- vfold_cv(imdb_train, v = 10)
metric <- metric_set(rmse)
ctrl_grid <- control_stack_grid()
ctrl_res <- control_stack_resamples()
```

KNN

```
knn_mod <- nearest_neighbor(
  mode = "regression",
  neighbors = tune("k")) %>%
  set_engine("kknn")


knn_wf<- workflow() %>%
  add_model(knn_mod) %>%
  add_recipe(profit_vote_rec)

#remember, k values should be odd
knn_grid <- data.frame(k = 1+2*0:20)

knn_fit<- knn_wf %>% tune_grid(
  resamples = folds,
  metrics = metric,
  grid = knn_grid,
  control = ctrl_grid)
```

Linear Model

```
lm_mod <- linear_reg() %>% set_engine("lm")

lm_wf<-workflow() %>%
  add_model(lm_mod) %>%
  add_recipe(profit_vote_rec)

lm_fit <- lm_wf %>%
  fit_resamples(resamples = folds,
  metrics = metric,
  control = ctrl_res)
```

LASSO

```
lasso_mod <- linear_reg(
  mode = "regression",
  penalty = tune("lambda")) %>%
  set_engine("glmnet")

lasso_wf<- workflow() %>%
  add_model(lasso_mod) %>%
  add_recipe(profit_vote_rec)
```

```
lasso_grid <- data.frame(lambda = 10^seq(-2, 8, length = 50))

lasso_fit<- lasso_wf %>% tune_grid(
  resamples = folds,
  grid = lasso_grid,
  control = ctrl_grid)
```

Random Forest

```
rf_mod <- rand_forest(mode = "regression", trees=30) %>%
  set_engine("randomForest")

rf_wf <- workflow() %>%
  add_model(rf_mod) %>%
  add_recipe(profit_vote_rec)

rf_fit <- rf_wf %>%
  fit_resamples(resamples = folds,
    metrics = metric,
    control = ctrl_res)
```

stack the stacks-profit

```
profit_st <- stacks() %>%
  add_candidates(knn_fit) %>%
  add_candidates(lm_fit) %>%
  add_candidates(lasso_fit) %>%
  add_candidates(rf_fit)
```

stack the stacks

```
stacks_grid <- 10^seq(-3, 6, length = 20)
profit_st_blend <- profit_st %>%
  blend_predictions(penalty = stacks_grid, metric = metric)
profit_en_fit<- profit_st_blend %>% fit_members()
profit_preds<- imdb_train %>% bind_cols(predict(profit_en_fit, .))
member_preds <- profit_preds %>% select(profit) %>%
  bind_cols(predict(profit_en_fit, imdb_train, members = T))
profit_rmse_train <- map_dfr(member_preds, rmse, truth = profit, data = member_preds) %>%
mutate(member = colnames(member_preds))
```

```
profit_preds<- imdb_test %>% bind_cols(predict(profit_en_fit, .))
member_preds <- profit_preds %>% select(profit) %>%
  bind_cols(predict(profit_en_fit, imdb_test, members = T))
profit_rmse_test <- map_dfr(member_preds, rmse, truth = profit, data = member_preds) %>%
mutate(member = colnames(member_preds))
```

```
profit_imdb_new <- read.csv('new_releases.csv')
profit_new_preds_regression <- profit_imdb_new %>% bind_cols(predict(profit_en_fit, .))
```

```
profit_penalty <- autoplot(profit_st_blend)
```

#Oscars model building ##

```
oscar_vote_rec <-
  recipe(oscar~writer+genre+budget+duration+director+year+production_company,data = imdb_train) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  #step_unknown()%>%
  step_zv(all_predictors())%>%
  step_normalize(all_numeric(), -all_outcomes())
```

10-fold CV

```
folds <- vfold_cv(imdb_train, v = 10)
metric <- metric_set(rmse)
ctrl_grid <- control_stack_grid()
ctrl_res <- control_stack_resamples()
```

KNN

```
knn_mod <- nearest_neighbor(
  mode = "regression",
  neighbors = tune("k")) %>%
  set_engine("kknn")


knn_wf<- workflow() %>%
  add_model(knn_mod) %>%
  add_recipe(oscar_vote_rec)

#remember, k values should be odd
knn_grid <- data.frame(k = 1+2*0:20)

knn_fit<- knn_wf %>% tune_grid(
  resamples = folds,
  metrics = metric,
  grid = knn_grid,
  control = ctrl_grid)
```

Linear Model

```
lm_mod <- linear_reg() %>% set_engine("lm")

lm_wf<-workflow() %>%
  add_model(lm_mod) %>%
  add_recipe(oscar_vote_rec)

lm_fit <- lm_wf %>%
  fit_resamples(resamples = folds,
  metrics = metric,
  control = ctrl_res)
```

LASSO

```
lasso_mod <- linear_reg(
  mode = "regression",
  penalty = tune("lambda")) %>%
  set_engine("glmnet")

lasso_wf<- workflow() %>%
  add_model(lasso_mod) %>%
  add_recipe(oscar_vote_rec)

lasso_grid <- data.frame(lambda = 10^seq(-2, 8, length = 50))

lasso_fit<- lasso_wf %>% tune_grid(
  resamples = folds,
  grid = lasso_grid,
  control = ctrl_grid)
```

Random Forest

```
rf_mod <- rand_forest(mode = "regression", trees=30) %>%
  set_engine("randomForest")

rf_wf <- workflow() %>%
  add_model(rf_mod) %>%
  add_recipe(oscar_vote_rec)

rf_fit <- rf_wf %>%
  fit_resamples(resamples = folds,
    metrics = metric,
    control = ctrl_res)
```

stack the stacks-IMDb

```
oscar_st <- stacks() %>%
  add_candidates(knn_fit) %>%
  add_candidates(lm_fit) %>%
  add_candidates(lasso_fit) %>%
  add_candidates(rf_fit)
```

stack the stacks

```
stacks_grid <- 10^seq(-3, 6, length = 20)
oscar_st_blend <- oscar_st %>%
  blend_predictions(penalty = stacks_grid, metric = metric)
oscar_en_fit<- oscar_st_blend %>% fit_members()
oscar_preds<- imdb_train %>% bind_cols(predict(oscar_en_fit, .))
member_preds <- oscar_preds %>% select(oscar) %>%
  bind_cols(predict(oscar_en_fit, imdb_train, members = T))
oscar_rmse_train <- map_dfr(member_preds, rmse, truth = oscar, data = member_preds) %>%
mutate(member = colnames(member_preds))
```

```
oscar_preds<- imdb_test %>% bind_cols(predict(oscar_en_fit, .))
member_preds <- oscar_preds %>% select(oscar) %>%
  bind_cols(predict(imdb_en_fit, imdb_test, members = T))
oscar_rmse_test <- map_dfr(member_preds, rmse, truth = oscar, data = member_preds) %>%
mutate(member = colnames(member_preds))
```

```
oscar_imdb_new <- read.csv('new_releases.csv')
oscar_new_preds_regression <- oscar_imdb_new %>% bind_cols(predict(oscar_en_fit, .))
```

```
oscar_penalty <- autoplot(oscar_st_blend)
```

```
#forward_select<-regsubsets(avg_vote~writer+genre+budget+duration+director+year+production_company,
#                           data = imdb, nvmax = 7, method = "forward")
```