Using Machine Learning to Analyze the Highest Rated Films on IMDb

By: Maura Anish

Introduction:

The Internet Movie Database (IMDb), described on their website as "the world's most popular and authoritative source for movie, TV and celebrity content," has allowed users to access information about such media since 1990. In addition to viewing production information, trivia, trailers, and synopses of films, registered IMDb users can also rate them on a scale from 1-10, with "1 meaning the title was terrible and… 10 meaning you think it was excellent" (IMDb Help). From the various individual users' ratings, a "weighted average" is calculated and displayed as the film's rating on its page. IMDb states that "although we accept and consider all votes received by users, not all votes have the same impact (or 'weight') on the final rating" (IMDb Help). They remain purposefully vague about how the ratings are precisely calculated "to prevent abuse and minimize attempts to stuff the ballot" (IMDb Help). A title only needs five user ratings to earn a weighted average, but in order to appear in the Top 250 Movies list, it requires at least 25,000 (IMDb Help).

My primary goal of this project was to gain an understanding of the trends in the 1,000 highest rated films on IMDb using machine learning techniques. I began my analysis using multiple linear regression to predict IMDbRating for a training subset of the data. I then proceeded to use other methods, such as ridge and lasso regression, best-subset selection, and regression trees. The shrinkage methods, through standardizing the features, contributed to my understanding of which features were most strongly related to the label via their parameter estimates. Best-subset selection indicated a similar relationship through its determination of which features should be included in the models. By fitting regression trees, I saw which features

were most important through the splits and the importance plots. Overall, I was interested in learning about why these films in particular have earned such high ratings from IMDb users.

About the Dataset:

The original dataset on Kaggle provided information about the top 1,000 films' IMDbRating, Title, Director, TopFourStars, Genre, AgeRating, ReleaseYear, and Duration. After cleaning that data, I went through and added four other variables to the dataset – Language, LeadGender, LeadRace, and MetacriticScore – by consulting each title's page on IMDb.

Name	Type	Data Type	Unique Values	Description
IMDbRating	Label	Numeric	18	Weighted average of user ratings
Title	Feature	String	998	Title of the film
Director	Feature	String	577	Person/people who directed the film
TopFourStars	Feature	String	998	Four actors for the film in the order they're credited on IMDb
Genre	Feature	String	202	Genres (at most 3) the film is categorized as on IMDb
AgeRating	Feature	String	7	MPAA Rating given to the film (G, PG, PG-13, R, etc.)
Language	Feature	String	32	Primary language listed for the film on IMDb
LeadGender	Feature	String	2	Gender of lead actor in the film
LeadRace	Feature	String	5	Race of lead actor in the film
ReleaseYear	Feature	Numeric	102	Year the film was released
Duration	Feature	Numeric	142	How long the film is, in minutes
MetacriticScore	Feature	Numeric	62	Weighted average of critic scores

Table 1: Description of the label and 11 features included in the dataset.

Analyzing the Variables:

I summarized and visualized information about the eleven variables I tried to incorporate into my models (with Title being the only variable of the twelve not worth analyzing). Some of the predictors contained multiple pieces of information separated by commas (Director, TopFourStars, and Genre), so some string processing was required before any analysis on those variables could be conducted. Since there were so many unique combinations of directors, actors, and genres, only some of the information contained in these features could be included in the models. The other string variables (AgeRating, Language, LeadGender, and LeadRace) had far fewer unique values, so they could be used as factors in the models. The remaining predictors were numeric (ReleaseYear, Duration, and MetacriticScore), and didn't have to be altered except for ReleaseYear, which was used to create a categorical ReleaseDecade variable.

First, I summarized the label: IMDbRating. The label ranges from 7.5 to 9.3 and has a mean of 7.96. As the histogram (Figure 1) shows, IMDbRating is skewed right. Very few films are rated higher than 8.5. The non-normal distribution of the label could make some of the regression models' normal-based results less reliable.

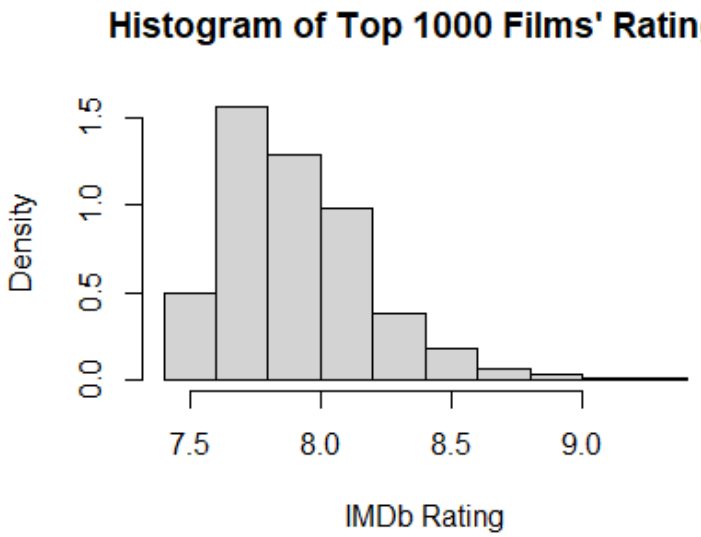

Figure 1: Histogram of IMDbRating.

The first two features that I summarized were Director and TopFourStars. The Director variable contains the names of the directors of each film. Most films only had one director, but some had more than one, and there was one film with twelve credited directors. In Table 2, the directors with nine or more films featured in the dataset are ranked according to how many films of theirs are included. The TopFourStars variable contains the names of four actors listed for a film, ordered primarily based on whoever was in the largest roles in the film. In Table 3, the actors who were featured in more than 10 of the films in the dataset are ranked according to how many films of theirs are included. When creating models later in the process, I tried to include a couple of Boolean variables that accounted for the presence or absence of these top directors or actors to see if their popularity correlated to a film being rated higher than others.

Director (9+ Films on List)	Films
Alfred Hitchcock	13
Steven Spielberg	13
Akira Kurosawa	11
Martin Scorsese	10
Hayao Miyazaki	10
Woody Allen	9
Stanley Kubrick	9
Billy Wilder	9

Table 2: Top Directors of Top 1,000 Films.

Actor (>10 Films on List)	Films
Robert De Niro	16
Tom Hanks	14
Al Pacino	13
Christian Bale	11
Matt Damon	11
Leonardo DiCaprio	11
Brad Pitt	11
James Stewart	11

Table 3: Top Actors of Top 1,000 Films.

Then, I summarized the Genre feature. There were 22 total genres that films were categorized as, but, as shown in Table 4, Drama was the most popular category by far, followed by Comedy and Crime. Most films were listed under multiple genres, with each film belonging to 2.5 genres on average. The maximum number of genres a film could be listed under is 3, and

those three genres are just the first three that are listed on IMDb, in alphabetical order. This negatively affects the number of films that are categorized as genres like Sci-Fi, Thriller, and War that are farther down in the alphabet.

Genre	Films
Drama	729
Comedy	236
Crime	206
Action	203
Adventure	191

Table 4: Most popular genres.

Since every film had at least 1 genre listed, but only some had more than 1, I created a new variable, Genre1, which contained the genres that were listed first for each film. Genre1 was originally a factor with 14 levels, but was later reduced to a factor of the 5 levels shown in Figure 2. The median IMDb rating for Crime films was slightly higher than that of the other categories.

First Genre	Films	First Genre	Films
Action	185	Family	1
Adventure	63	Fantasy	2
Animation	83	Film-Noir	2
Biography	87	Horror	13
Comedy	155	Mystery	9
Crime	110	Thriller	1
Drama	285	Western	4

Table 5: First-listed genres for each film.

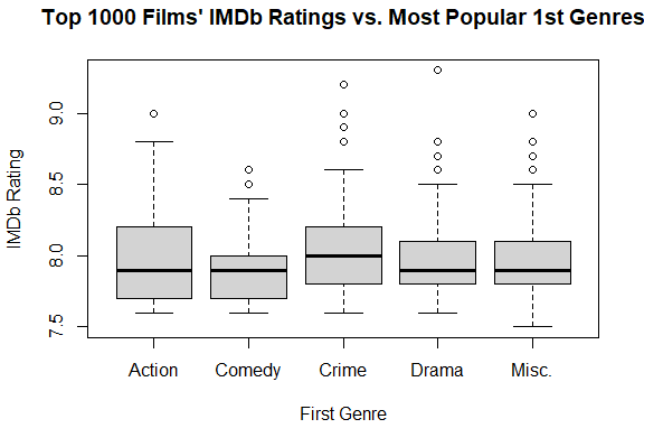

Figure 2: Boxplot of IMDbRating vs. Genre1.

Following Genre, I analyzed the Duration feature. The shortest film on the list was 45 minutes long and the longest film on the list was 321 minutes long, but the average duration was 123.7 minutes. Figure 3 shows the histogram of Duration to be skewed right, with very few films running longer than 200 minutes. Figure 4 shows the relationship between the films' IMDb ratings and their durations. There is a weak positive linear relationship between the two variables, as demonstrated by the correlation of 0.26. The lowess line on the graph also indicates this slight positive trend, but the 321 minute film represents an outlier which could be having a significant effect on these metrics.

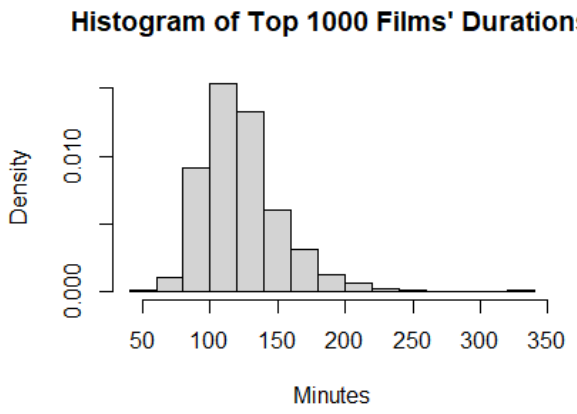

Figure 3: Histogram of Duration.

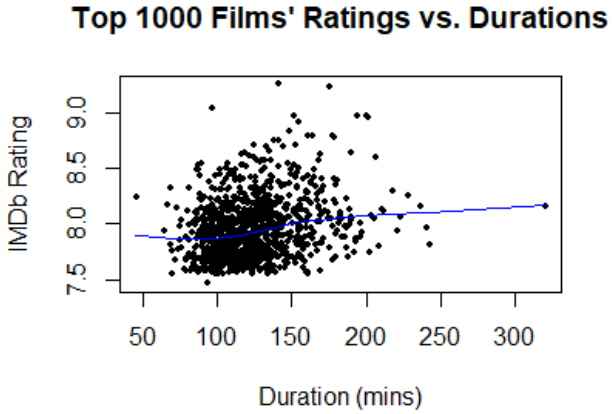

Figure 4: Scatterplot of IMDbRating vs. Duration.

After Duration, I summarized the AgeRating feature, which reports how the Motion Picture Association of America (MPAA) classified the film. "Passed" was used to rate films released prior to 1969, according to whether or not their content followed the Hays Production Code, and PG-13 was added to the G, PG, R, or NC-17 scale in 1984 (FilmRatings.com). A film is listed as NR if it was categorized as "Not Rated" or "Unrated" or it didn't have a rating listed on IMDb. As seen in Figure 5, the median IMDb rating of each group was close to 8.0, and films rated R had the widest spread, including the two highest outliers.

Age Rating	Films
Passed	78
G	39
PG	151
PG-13	171
R	367
NC-17	5
NR	189

Table 6: Films by Age Rating.

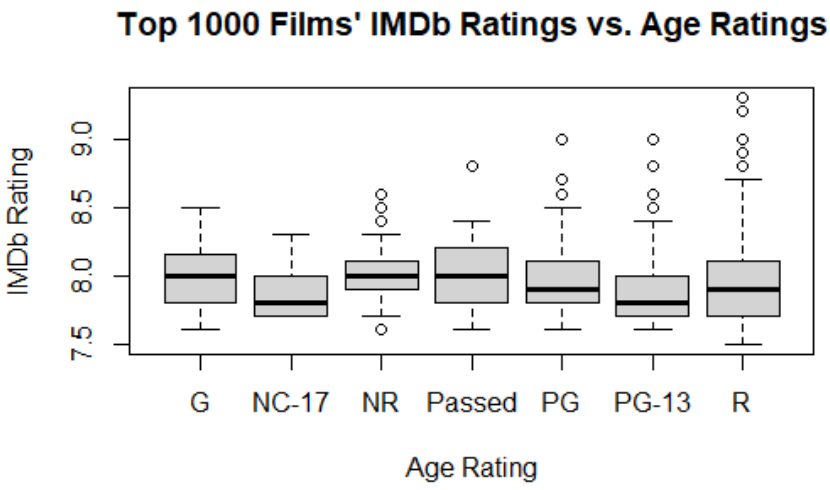

Figure 5: Boxplot of IMDbRating vs. AgeRating

Next, I summarized the ReleaseYear feature. The earliest film on the list was released in 1920, and the most recent film on the list was released in 2023. The median release year was 1999, and the mean release year was 1991. Figure 6 shows the histogram of ReleaseYear, which is skewed left and indicates a preference towards more recently released films – especially those released in the 2000's and 2010's. Figure 7 shows the relationship between a film's IMDb rating and its release year, but there doesn't seem to be any clear relationship between the two. The correlation coefficient was -0.1, which suggests a very weak negative linear relationship.

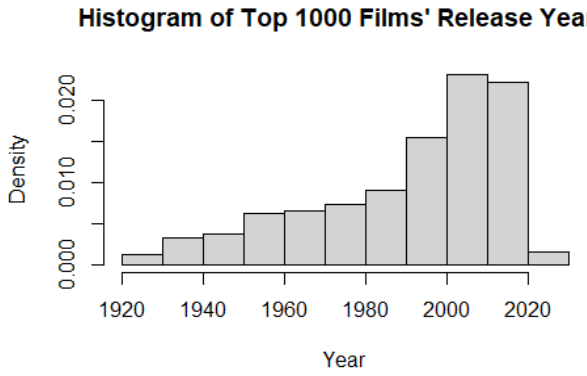

Figure 6: Histogram of ReleaseYear.

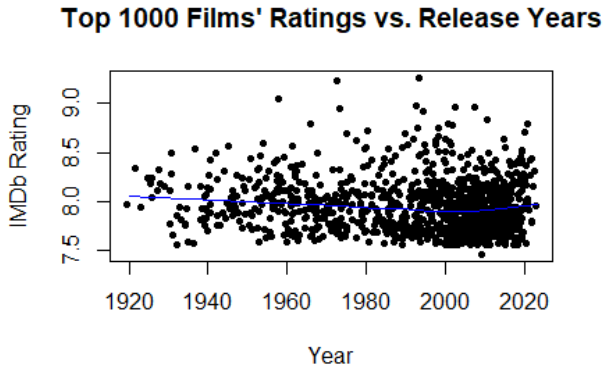

Figure 7: Scatterplot of IMDbRating vs. ReleaseYear.

Given that ReleaseYear isn't technically a numeric variable, I created ReleaseDecade, which grouped the release years together and examined their relationships to IMDbRating in a different way. Table 7 reflects the trends that were observed in the discussion of ReleaseYear – the 2000's and 2010's are certainly the decades with the most films in the dataset. Figure 8 shows some slight variation in the median IMDb rating for the decades, with the 1920's and the 2020's having the highest median ratings of over 8.0. This is likely due to the lower amount of films that are included in the dataset from these decades.

Decade	Films
1920s	11
1930s	26
1940s	38
1950s	58
1960s	75
1970s	68
1980s	89
1990s	145
2000s	229
2010s	237
2020s	24

Table 7: Films by Decade.

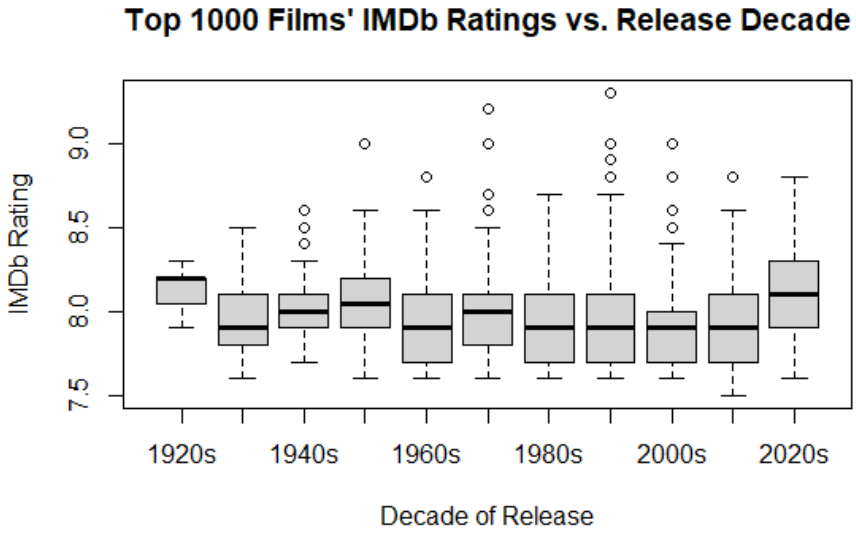

Figure 8: Boxplot of IMDbRating vs. ReleaseDecade.

Following ReleaseYear, I summarized Language, which represents the first-listed and primarily-spoken language for the films. As shown in Table 8, English films were by far the most popular on the Top 1,000 Films list. Although there were 32 languages that had at least one film on the list, only 30% of the films weren't primarily in English. Figure 9 shows how varied the

IMDb rating spreads were for the most popular languages in the table (those which had more than 15 films on the list). The highest rated films were in English, yet the films in Hindi and Japanese had the highest median ratings. The Hindi category also had the smallest spread, while the English category had the widest spread.

Top 5 Languages	Films
English	700
Hindi	61
Japanese	49
French	43
German	17
Spanish	17

Table 8: Most popular languages.

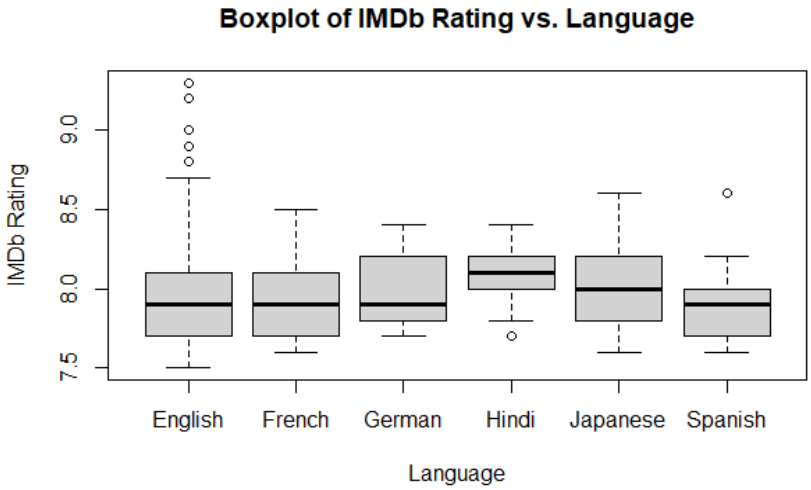

Figure 9: Boxplot of IMDbRating vs. Language.

Then, I analyzed the MetacriticScore feature. According to the Metacritic website, they "carefully curate a large group of the world's most respected critics, assign scores to their reviews, and apply a weighted average to summarize the range of their opinions." These averages range from 0 to 100 and are made using at least 4 critics' scores (Metacritic.com). MetacriticScore is the only predictor that has some NA values, because 149 films in the dataset didn't have a rating available on Metacritic. The films that did have Metacritic scores ranged from 28 to 100, with a mean score of 78.79. Figure 10 shows the Metacritic scores are skewed left, but not so much as the IMDb ratings are. The scatterplot of IMDbRating vs. MetacriticScore shows a slightly positive linear relationship, with a correlation of 0.25.

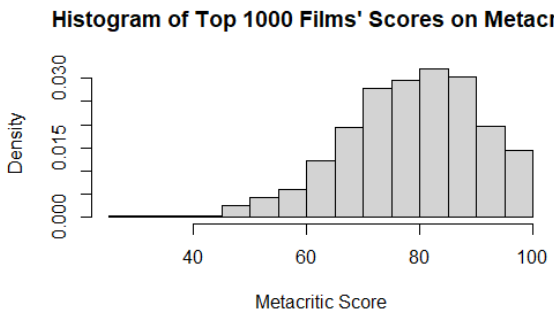

Figure 10: Histogram of MetacriticScore.

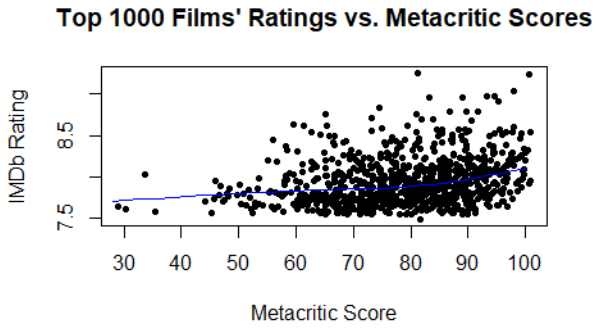

Figure 11: Scatterplot of IMDbRating vs. MCScore.

The last two variables I analyzed were LeadGender and LeadRace. 831 of the Top 1,000 films had a male actor listed as the first in the cast, and only 169 had a female actor listed first. According to Figure 12, though, the median score for films led by men and films led by women is approximately the same. Films led by men generally had a larger spread and higher outliers. The analysis of LeadRace showed that 770 of the Top 1,000 films were led by a white actor, with 170 led by Asian actors, 31 led by Black actors, 27 led by Latino actors, and 2 led by Pacific Islander actors. Figure 13 shows that films led by Asian or Latino actors had slightly higher median ratings than those led by others. The white-led films had the largest spread of ratings and included many high outliers, as well.

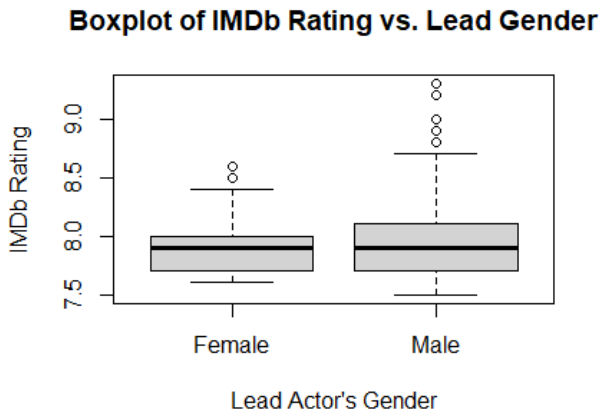

Figure 12: Boxplot of IMDbRating vs. LeadGender.

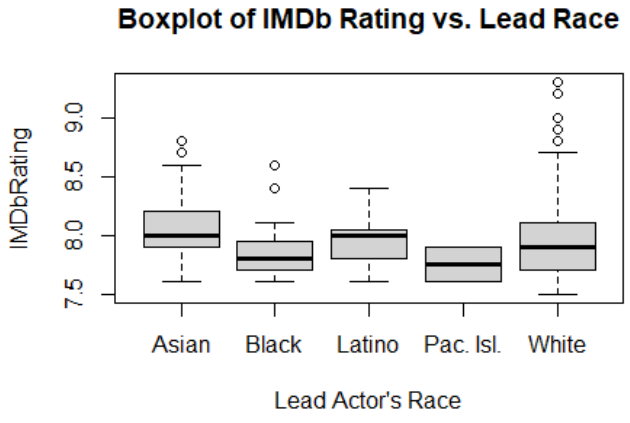

Figure 13: Boxplot of IMDbRating vs. LeadRace.

Fitting Linear Regression Models:

Having completed my analysis of the label and features, I proceeded to fit multiple linear regression models on a training dataset with 75% of the data (750 films) to see how the variables were being used to make predictions for IMDbRating.

The first model I fit used Genre1, Duration, AgeRating, ReleaseDecade, Language, MetacriticScore, LeadGender, and LeadRace to predict IMDbRating, because these variables could all be incorporated into the model as is, due to their being numeric or factors with at most 32 levels. The model had a total of 73 parameters, plus the intercept. None of the age ratings or lead races were significant (i.e. none had p-values below alpha=.05). In Genre1, whether the film was a mystery was significant, in ReleaseDecade, whether the film was released in the 1960's or 2010's was significant, and in Language, whether the film was in English, French, Italian, or Russian was significant. Duration, MetacriticScore, and LeadGender were also significant. Duration and MetacriticScore had the lowest p-values by far. I was hesitant to include MetacriticScore in the model because of its NA values, but this model convinced me to keep it.

When I tried to apply this first model to the test dataset, however, I was unable to do so because there were certain values in the Language variable which were present in the test data that weren't in the training data. To combat this, I created a new variable, EnFrItRu, which was 1 if the film was primarily in English, French, Italian, or Russian and 0 otherwise, because those four languages were shown to be the most significant in the training model's outcome. I replaced Language with this indicator variable, and then I was able to use the model to predict IMDbRating from the test data. The mean-squared error for the test data was 0.0770.

Next, I tried to add two other variables in to see how they affected the model's results. In the second model I fit, I added a Boolean variable, HasTopDir, which was 1 if any of the top 8

most popular directors listed in Table 2 directed the film and 0 otherwise. With the inclusion of HasTopDir, the MSE for the test data was 0.0769, indicating a hardly noticeable decrease in the error. For the third model I fit, I made a Boolean variable, HasTopActor, to check for the presence of the top 3 most popular actors listed in Table 3, which was 1 if any of them were in the film and 0 otherwise. The inclusion of this variable along with the other variables in the previous model produced a slightly higher MSE of 0.0771.

On this first pass through the data, the models I came up with suggested that longer films that are led by a male actor, have higher scores on Metacritic, and aren't in English, French, Italian, or Russian are likely to be rated higher on IMDb than others. I learned that including variables like HasTopDir or HasTopActor weren't very helpful to understand why certain films on IMDb are rated so highly, because the personnel that they checked for were only involved in a fraction of the 1,000 total highest rated films. For the other algorithms that I used to fit models to the data, I stuck with the set of variables that I used for the initial linear regression model: Genre1, Duration, AgeRating, ReleaseDecade, Language/EnFrItRu, MetacriticScore, LeadGender, and LeadRace.

Fitting Ridge and Lasso Regression Models:

With the same training and testing set and features that I used to fit the linear regression models, I fit a ridge regression model. The best lambda was chosen to be 0.074, and the test MSE was 0.075, which was slightly lower than that of the linear regression model. When examining the parameter estimates from the ridge model, I discovered that the most significant variables for predicting IMDbRating were whether the film was categorized as a Fantasy, Mystery, or Western or if it was led by a Pacific Islander actor. It was interesting that MetacriticScore, Duration, and LeadGender weren't nearly as important here as they were to the linear regression model.

Next, with the same training set, testing set, and subset of features, I fit a lasso regression model. The best lambda was chosen to be 0.006, and the test MSE was 0.0741, which was lower than all of the previous measures from the other algorithms. The parameter estimates for the lasso model indicated that the same features identified by the ridge model (Genre1Fantasy, Genre1Mystery, Genre1Western, and LeadRacePacificIslander) were the most important to predict IMDbRating. With standardized variables, these models allowed me to gain a better understanding of which parameters were having the most impact on the predictive results.

Running Best-Subset Selection:

After running best-subset selection on the same training set, testing set, and subset of features, I got some different results. I allowed a maximum of 10 variables to be used in the models, and I found that the top five most important variables were, in this order: MetacriticScore, Duration, ReleaseDecade1990s, LeadMale, and Genre1Mystery. This was the first time that anything from the ReleaseDecade feature seemed to be important to the model's predictive power for IMDbRating.

Fitting Regression Trees:

When I fit a tree on the same training data set, using the same predictors, it seemed as though MetacriticScore, Duration, and ReleaseDecade were the more important features, as shown in Figure 14. I used the "ANOVA" method to produce numeric results, chose an alpha of 0.01, and pruned the tree with the same value of alpha. The test MSE from the tree was 0.079.

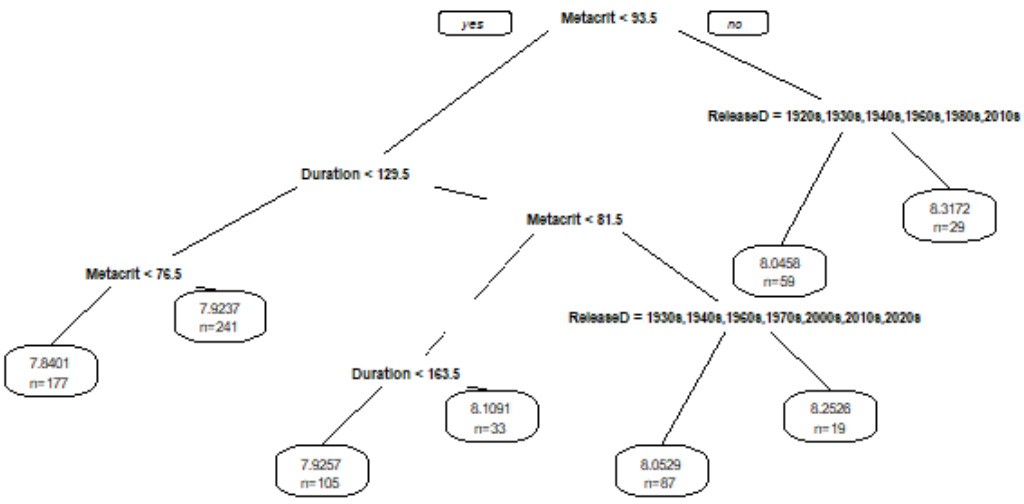

Figure 14: A regression tree fit on the training data with 7 splits.

Next, I used random forest to generate importance plots and examine the features' effects on IMDbRating that way. With the choice of five randomly selected variables at each split in the trees, the test MSE was 0.0708, which was the lowest MSE so far of any of the algorithms. The most important variables, according to the importance plots shown in Figure 15 below, were MetacriticScore, Duration, and ReleaseDecade, followed by either Genre1, Language/EnFrItRu, or AgeRating.

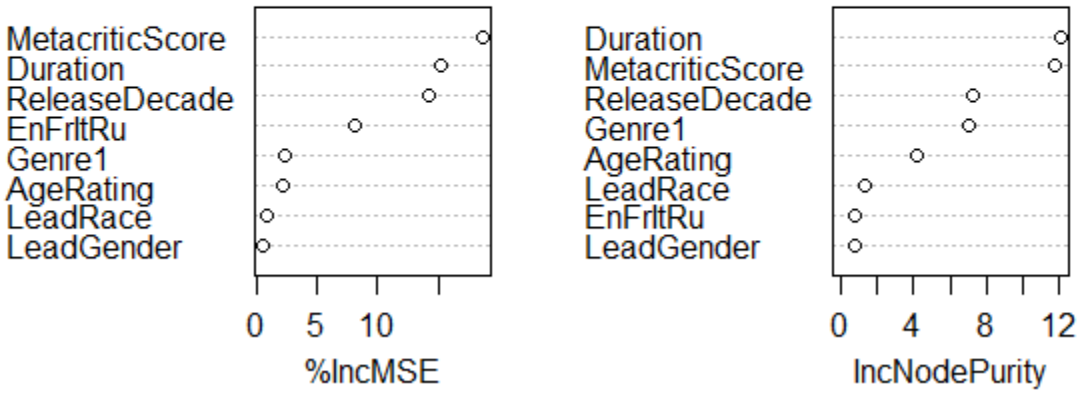

Figure 15: The random forest trees' importance plots.

Applying Techniques to Other Training Sets:

After I established how different the results from different algorithms could be for just one training dataset, I created three other training and testing sets (of 750 films and 250 films each) to see how much the results from each algorithm varied from one set to another.

Set #	Linear Reg.	Best Subset	Reg. Trees	Ridge Reg.	Lasso Reg.
1	1. MCScore 2. Duration 3. EnFrItRu MSE: .0770	1. MCScore 2. Duration 3. RD1990s MSE: .0732	1. MCScore 2. Duration 3. ReleaseDec MSE: .0708	1. G1Mystery 2. G1Fantasy 3. LRPacIsland MSE: .0754	1. G1Mystery 2. LRPacIsland 3. G1Fantasy MSE: .0741
2	1. Duration 2. MCScore 3. LGMale MSE: .0582	1. Duration 2. MCScore 3. RD1990s MSE: .0556	1. Duration 2. MCScore 3. ReleaseDec MSE: .0576	1. G1Fantasy 2. LRPacIsland 3. G1Mystery MSE: .0548	1. G1Mystery 2. Duration 3. MCScore MSE: .0528
3	1. Duration 2. MCScore 3. G1Mystery MSE: .0618	1. Duration 2. MCScore 3. RD1990s MSE: .0626	1. Duration 2. MCScore 3. ReleaseDec MSE: .0624	1. G1Mystery 2. G1Fantasy 3. LRPacIsland MSE: .0611	1. G1Mystery 2. Duration 3. RD1950s MSE: .0607
4	1. Duration 2. MCScore 3. LGMale MSE: .0605	1. MCScore 2. Duration 3. RD1990s MSE: .0607	1. Duration 2. MCScore 3. ReleaseDec MSE: .0637	1. G1Fantasy 2. G1Mystery 3. G1Thriller MSE: .0617	1. G1Mystery 2. G1Fantasy 3. MCScore MSE: .0605

Table 9: Summary of the most important variables according to each method for four run-throughs.

Regardless of the algorithm used to analyze the data or the training set that was chosen, it appeared that Duration and MetacriticScore had the most impact on the predictions of IMDbRating. ReleaseDecade and Genre1 seemed to play important roles, as well, while LeadRace and LeadGender had some significance and Language/EnFrItRu had a small influence. Evidently, AgeRating had a very minimal impact on the IMDb ratings, as it was never featured in the table. MetacriticScore and Duration always had positive coefficients, indicating that longer films which are more highly rated by critics are more likely to be highly rated by users on IMDb.

Some features which only had a few films in certain categories had a disproportionate effect on the outcomes. Particularly, in the Genre1 feature, the Mystery and Fantasy categories seemed to have a strong influence on the films' ratings, but there were only 9 films with Mystery as their first genre and 2 films with Fantasy as their first genre in the whole dataset. Similarly, with LeadRace, there were only 2 films out of the 1,000 that were led by a Pacific Islander, yet this aspect also had a large influence on the films' ratings. This isn't surprising, as when there are less films in a certain group, the spread of IMDbRating for that group will be much narrower than it would be for a group with a lot of films in it. However, the degree to which the ridge and lasso methods were affected by these small groups was surprising.

Given this imbalance, I decided to create new versions of the factors that had levels with very few films. This involved first reducing Genre1 from a factor of 14 levels to a factor of 5 levels: "Action," "Comedy," "Crime," "Drama," or "Misc." for the rest. Next, I changed the Language variable from EnFrItRu, which was a Boolean indicator variable for those four languages, to a Language factor with 5 levels: "English," "Hindi," "Japanese," "French," or "Misc." for the rest, since these four languages had the most films, as shown in Table 8. Finally, I changed LeadRace to only have 4 levels: "Asian," "Black," "White," and "Latino/Pac.Isl." With these new factors, no category had less than 10 films in it.

I re-ran the five algorithms on the four training sets of 750 and test sets of 250 that I used previously. This time, Genre1 was hardly present in the table at all, LeadRace was present less, and Language was present a lot more. Some of the categories with fewer films in them still seemed to be affecting the results strongly (there were only 49 films in Japanese, 61 films in Hindi, 29 films led by Latino/Pacific Islander actors, and 31 films led by Black actors), but these weren't the categories with the fewest films in them overall. For example, AgeRatingNC-17 only

had 5 films in it, and there were only 11 films from the 1920s, 24 from the 2020s, and 26 from the 1930s, yet none of those categories were explicitly included in Table 10.

Set #	Linear Reg.	Best Subset	Reg. Trees	Ridge Reg.	Lasso Reg.
1	1. MCScore 2. Duration 3. LaJapanese MSE: .0767	1. MCScore 2. Duration 3. LaJapanese MSE: .0755	1. MCScore 2. Duration 3. ReleaseDec MSE: .0708	1. LaJapanese 2. LRLatPacIs 3. LaHindi MSE: .0751	1. LaJapanese 2. MCScore 3. Duration MSE: .0738
2	1. Duration 2. MCScore 3. LGMale MSE: .0576	1. Duration 2. MCScore 3. RD1990s MSE: .0556	1. Duration 2. MCScore 3. ReleaseDec MSE: .0575	1. LaJapanese 2. Duration 3. RD1990s MSE: .0542	1. Duration 2. MCScore 3. G1Crime MSE: .0528
3	1. Duration 2. MCScore 3. LaJapanese MSE: .0619	1. Duration 2. MCScore 3. RD1990s MSE: .0626	1. Duration 2. MCScore 3. ReleaseDec MSE: .0636	1. LaHindi 2. LaJapanese 3. RD1950s MSE: .0608	1. RD1950s 2. Duration 3. MCScore MSE: .0601
4	1. MCScore 2. Duration 3. LGMale MSE: .0598	1. MCScore 2. Duration 3. RD1990s MSE: .0607	1. Duration 2. MCScore 3. ReleaseDec MSE: .0649	1. RD1950s 2. LaHindi 3. LRBlack MSE: .0616	1. MCScore 2. Duration 3. RD1990s MSE: .0608

Table 10: Summary of the most important variables according to each method for four new run-throughs.

Conclusion:

Overwhelmingly, the models indicated that longer films which are rated higher on Metacritic are more likely to be rated higher on IMDb. The decade in which the film was released and the language the film was primarily in also seemed to have an effect on its IMDb rating, while the race and gender of the film's lead actor was somewhat important as well.

It is difficult to know, however, how much of MetacriticScore and Duration's impact resulted from them being the only two numeric variables included in the models. If I were to modify the dataset further and re-conduct the analyses with these algorithms, I would try to add more numeric variables to the dataset and see if those had as strong as an impact as

MetacriticScore and Duration did. Some numeric variables I could add, for example, would be OscarsWon (the number of Oscars each film won), BoxOffice (the amount each film made at the international box office), and NumRatings (the number of user ratings that were submitted for each film on IMDb which were used to calculate the weighted average rating).

These results have a limited scope of application, due to the small range of the label and the many levels that each factor had. I created a new test dataset with 100 films that weren't included in the Top 1,000 list and tried to use the models trained on all 1,000 films in the original dataset to predict IMDbRating for these new films. The regression, best subset selection, and regression tree models were able to predict IMDbRating for the new films, but, unsurprisingly, as shown in Figure 16, the predicted ratings were much higher than the actual ratings of the test data (which ranged from 5.4 to 7.8). Instead of the test MSEs of around 0.05 and 0.07 which I found earlier, these test MSEs were closer to 1.09 and 1.18.

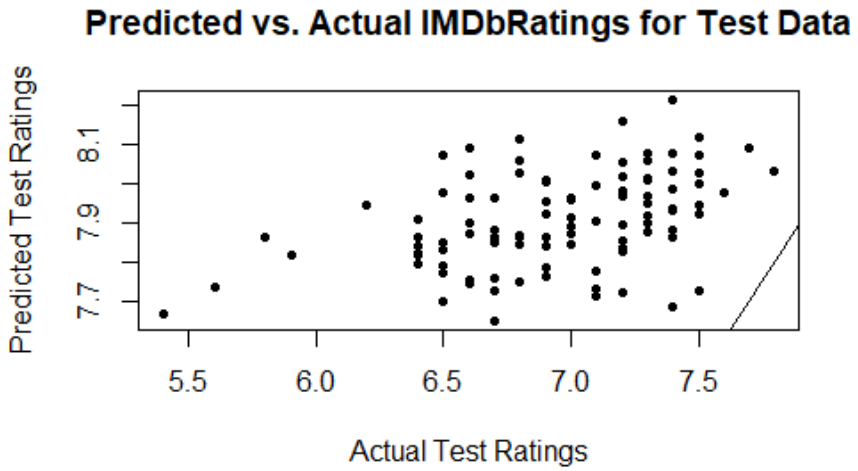

Figure 16: Scatterplot of standard linear regression's predictions of IMDbRating vs. the actual IMDbRating values for the new test dataset of 100 films.

The ridge, lasso, and random forest models trained on the original data weren't even able to run on the new data, because I hadn't included films with every level of every factor in the new test data. Because I didn't have any films in Japanese or Hindi, any films before the 1960s,

or any films rated “Passed” or “G” in the new dataset I created, these three models weren’t able to predict IMDbRating for the new data. Although the goal of this project wasn’t to apply the predictive models trained on these data to other films that weren’t in the dataset, it is worth noting how finely tuned each of the models are to these data.

In future analysis on the highest-rated films on IMDb, it could be interesting to look solely at the Top 250 Movies list that’s publicly displayed on IMDb. Only 15 of those films don’t have a Metacritic score, so there would be less data lost when omitting those than there was in omitting the 149 films without them in this dataset. Moreover, this list of 250 films could be compared to the Top 250 Narrative Feature Films list on Letterboxd – another popular, newer website that creates a “weighted calculation” from users’ ratings of films on a scale of 0.5 to 5 stars (Letterboxd.com). If one were to fit models to both sets of the 250 films and then compare which features the models considered to be the most important in predicting the films’ weighted average scores, one could get a better sense of whether the features that are important in the IMDb models are unique to IMDb.

R Code

```
# read data in from an Excel spreadsheet
library(readxl)
imdb.data <- read_excel("C:/Users/maura/Downloads/IMDb_Data.xlsx")

#####################################################################
# look at summaries of eleven variables
# 1. IMDbRating
# look at numerical summary
summary(imdb.data$IMDbRating)
# Min.   1st Qu.  Median  Mean   3rd Qu.   Max.
# 7.500   7.700   7.900   7.957   8.100   9.300
# make a histogram to look at the spread (Figure 1)
hist(imdb.data$IMDbRating, freq=F, xlab="IMDb Rating",
     main="Histogram of Top 1000 Films' Ratings")

# 2. Director
# sort the directors by number of films they have in the dataset
sort(table(imdb.data$Director))

# 3. TopFourStars
# get all actors' names into a list
four.thousand.actors <- NULL
actor1s <- rep(NA,1000); actor2s <- rep(NA,1000)
actor3s <- rep(NA,1000); actor4s <- rep(NA,1000)
for(i in 1:1000){
  four.actors <- strsplit(imdb.data$TopFourStars[i], split=", ")
  actor1s[i] <- four.actors[[1]][1] # store first actors
  actor2s[i] <- four.actors[[1]][2] # store second actors
  actor3s[i] <- four.actors[[1]][3] # store third actors
  actor4s[i] <- four.actors[[1]][4] # store fourth actors
}
imdb.data$Actor1 <- actor1s # add 1st actors to dataframe
imdb.data$Actor2 <- actor2s # add 2nd actors to dataframe
imdb.data$Actor3 <- actor3s # add 3rd actors to dataframe
imdb.data$Actor4 <- actor4s # add 4th actors to dataframe
four.thousand.actors <- c(actor1s, actor2s, actor3s, actor4s)
# determine which actors are in more than 10 movies
big.actors <- table(four.thousand.actors)>10 # T if in more than 10
big.actors.names <- NULL # store names of those in more than 10 here
for(i in 1:2689){ # for each of the 2689 unique actors
  if(big.actors[i]==T){ # if the actor has been in more than 10
    big.actors.names <- c(big.actors.names, big.actors[i]) # store
  }
```

```
}
big.actors.names # look at the actors that were in more than 10
sum(four.thousand.actors=="AlPacino") # Al Pacino in 13
sum(four.thousand.actors=="BradPitt") # Brad Pitt in 11
sum(four.thousand.actors=="ChristianBale") # Christian Bale in 11
sum(four.thousand.actors=="JamesStewart") # James Stewart in 11
sum(four.thousand.actors=="LeonardoDiCaprio") # Leo DiCaprio in 11
sum(four.thousand.actors=="MattDamon") # Matt Damon in 11
sum(four.thousand.actors=="RobertDeNiro") # Rob De Niro in 16
sum(four.thousand.actors=="TomHanks") # Tom Hanks in 14

# 4. Genre
# put all genres into one list
all.genres <- NULL
genre1s <- rep(NA, 1000); genre2s <- rep(NA, 1000)
genre3s <- rep(NA, 1000) # store up to 3 genres for each film
for(i in 1:1000){
  genres <- strsplit(imdb.data$Genre[i],split=",")
  genre1s[i] <- genres[[1]][1] # first listed genre
  genre2s[i] <- genres[[1]][2] # second listed genre
  genre3s[i] <- genres[[1]][3] # third listed genre
}
all.genres <- c(genre1s, genre2s, genre3s) # make list of all genres
all.genres <- na.omit(all.genres) # get rid of NAs
length(all.genres)/1000 # 2539/1000 = average of 2.5 genres/film
# look at how many films were in each genre (Table 4)
table(all.genres)
# change genre to factor of Action, Comedy, Crime, Drama, or Misc.
genre1 <- rep("Misc.", 1000)
for(i in 1:1000){
  if     (genre1s[i]=="Action"){genre1[i] <- "Action"}
  else if(genre1s[i]=="Comedy"){genre1[i] <- "Comedy"}
  else if(genre1s[i]=="Crime"){genre1[i] <- "Crime"}
  else if(genre1s[i]=="Drama"){genre1[i] <- "Drama"}
}
imdb.data$Genre1 <- as.factor(genre1) # add 1st genre to dataframe
# look at table of first-listed genres (Table 5)
table(imdb.data$Genre1)
# make boxplot to look at IMDbRating spread for each genre (Figure 2)
plot(imdb.data$IMDbRating ~ imdb.data$Genre1,
     xlab = "First Genre", ylab = "IMDb Rating",
     main="Top 1000 Films' IMDb Ratings vs. Most Popular 1st Genres")

# 5. Duration
# look at numerical summary
```

```
summary(imdb.data$Duration)
# Min.  1st Qu.  Median  Mean   3rd Qu.   Max.
# 45.0   103.0   120.0   123.7   138.0   321.0
# make a histogram to look at the spread (Figure 3)
hist(imdb.data$Duration, freq=F, xlab="Minutes",
     main="Histogram of Top 1000 Films' Durations")
# scatterplot to look at relationship between rating and runtime
# with slightly jittered points (Figure 4)
set.seed(12)
dur.jit <- runif(1000, -1, 1) # for Duration
set.seed(12)
rat.jit <- runif(1000, -.05, .05) # for IMDbRating
plot((imdb.data$Duration+dur.jit), (imdb.data$IMDbRating+rat.jit),
     xlab="Duration (mins)", ylab = "IMDb Rating", pch=20,
     main = "Top 1000 Films' Ratings vs. Durations", cex=.8)
# add a lowess line to visualize the trend
lines(lowess(imdb.data$Duration, imdb.data$IMDbRating), col="blue")
# look at correlation between rating and runtime
cor(imdb.data$Duration, imdb.data$IMDbRating) # 0.26
# scale Duration to standardize it
imdb.data$Duration <- scale(imdb.data$Duration)

# 6. AgeRating
# see how many films were in each rating category (Table 6)
table(imdb.data$AgeRating)
# G   NC-17   NR   Passed  PG    PG-13    R
# 39    5    189     78    151    171    367
# look at IMDb rating spread for each category (Figure 5)
boxplot(imdb.data$IMDbRating ~ imdb.data$AgeRating,
        xlab="Age Rating", ylab="IMDb Rating",
        main = "Top 1000 Films' IMDb Ratings vs. Age Ratings")
# convert the variable to a factor in the dataframe
imdb.data$AgeRating <- as.factor(imdb.data$AgeRating)

# 7. ReleaseYear
# look at numerical summary
summary(imdb.data$ReleaseYear)
# Min.   1st Qu. Median   Mean   3rd Qu.  Max.
# 1920    1975    1999    1991    2010    2023
# make histogram to look at the spread (Figure 6)
hist(imdb.data$ReleaseYear, freq=F, xlab="Year",
     main = "Histogram of Top 1000 Films' Release Years")
# make scatterplot to look at rating and year, w/ jitter (Figure 7)
set.seed(12)
year.jit <- runif(1000,-1,1)
```

```
plot((imdb.data$ReleaseYear+year.jit),(imdb.data$IMDbRating+rat.jit),
     xlab="Year", ylab = "IMDb Rating", pch=20,
     main = "Top 1000 Films' Ratings vs. Release Years")
# add a lowess line to the plot to visualize the trend
lines(lowess(imdb.data$ReleaseYear,imdb.data$IMDbRating), col="blue")
# look at the correlation between rating and release year
cor(imdb.data$ReleaseYear, imdb.data$IMDbRating) # -0.10
# use ReleaseYear to make ReleaseDecade and add to dataframe
decade <- rep(NA, 1000)
for(i in 1:1000){
  if     (imdb.data$ReleaseYear[i]<1930){decade[i]<-"1920s"}
  else if(imdb.data$ReleaseYear[i]<1940){decade[i]<-"1930s"}
  else if(imdb.data$ReleaseYear[i]<1950){decade[i]<-"1940s"}
  else if(imdb.data$ReleaseYear[i]<1960){decade[i]<-"1950s"}
  else if(imdb.data$ReleaseYear[i]<1970){decade[i]<-"1960s"}
  else if(imdb.data$ReleaseYear[i]<1980){decade[i]<-"1970s"}
  else if(imdb.data$ReleaseYear[i]<1990){decade[i]<-"1980s"}
  else if(imdb.data$ReleaseYear[i]<2000){decade[i]<-"1990s"}
  else if(imdb.data$ReleaseYear[i]<2010){decade[i]<-"2000s"}
  else if(imdb.data$ReleaseYear[i]<2020){decade[i]<-"2010s"}
  else if(imdb.data$ReleaseYear[i]<2024){decade[i]<-"2020s"}
}
imdb.data$ReleaseDecade <- as.factor(decade)
table(decade) # look at summary of decades (Table 7)
# make boxplot of IMDbRating vs. ReleaseDecade (Figure 8)
boxplot(imdb.data$IMDbRating~imdb.data$ReleaseDecade,
        xlab = "Decade of Release", ylab = "IMDb Rating",
        main = "Top 1000 Films' IMDb Ratings vs. Release Decade")

# 8. Language
# see how many films were in each language (Table 8)
table(imdb.data$Language)
# see spread of rating for languages with > 15 films (Figure 9)
pop.langs <- ifelse( imdb.data$Language=="English" |
                     imdb.data$Language=="French" |
                     imdb.data$Language=="German" |
                     imdb.data$Language=="Hindi" |
                     imdb.data$Language=="Japanese" |
                     imdb.data$Language=="Spanish", 1, 0)
boxplot(imdb.data$IMDbRating[pop.langs==1] ~
        imdb.data$Language[pop.langs==1],
        xlab="Language", ylab="IMDb Rating",
        main="Boxplot of IMDb Rating vs. Language")
# modify Language to have English, Hindi, Japanese, French, or Misc.
language <- rep("Misc.", 1000)
```

```
for(i in 1:1000){
  if     (imdb.data$Language[i]=="English"){language[i]<-"English"}
  else if(imdb.data$Language[i]=="Hindi"){language[i]<-"Hindi"}
  else if(imdb.data$Language[i]=="Japanese"){language[i]<-"Japanese"}
  else if(imdb.data$Language[i]=="French"){language[i]<-"French"}
}
# store languages as a factor in the dataframe
imdb.data$Language <- as.factor(language)

# 9. Metacritic Score
# store Metacritic Score as an int, not a char, in dataframe
imdb.data$MetacriticScore <- as.integer(imdb.data$MetacriticScore)
# look at numerical summary
summary(imdb.data$MetacriticScore)
#  Min.  1st Qu.  Median  Mean   3rd Qu.  Max.    NA's
# 28.00   71.00   80.00   78.79   88.00  100.00   149
# make separate dataset with films that have MC scores
MC.score.films <- na.omit(imdb.data)
# make histogram to look at spread (Figure 10)
hist(MC.score.films$MetacriticScore,
     xlab = "Metacritic Score", freq=F,
     main = "Histogram of Top 1000 Films' Scores on Metacritic")
# make scatterplot for IMDb score and Metacritic score (Figure 11)
set.seed(12) # jitter Metacritic Scores
mc.jit <- runif(851, -1, 1)
set.seed(12) # jitter IMDb Ratings
rat.jit <- runif(851, -.05, .05)
plot((MC.score.films$MetacriticScore+mc.jit),
     (MC.score.films$IMDbRating+rat.jit), pch=20,
     xlab = "Metacritic Score", ylab = "IMDb Rating",
     main = "Top 1000 Films' Ratings vs. Metacritic Scores")
# add lowess line to plot to visualize trend
lines(lowess(MC.score.films$MetacriticScore,
      MC.score.films$IMDbRating), col="blue")
# correlation between critics' ratings and audiences' ratings
cor(MC.score.films$MetacriticScore, MC.score.films$IMDbRating)#0.25
# scale MetacriticScore to standardize it
imdb.data$MetacriticScore <- scale(imdb.data$MetacriticScore)

# 10. Lead Gender
# see how many films had a lead actor of different genders
table(imdb.data$LeadGender)
#   F   M
# 169 831
# look at spread of ratings for different genders (Figure 12)
```

```
boxplot(imdb.data$IMDbRating ~ imdb.data$LeadGender,
        xlab="Lead Actor's Gender", names=c("Female","Male"),
        ylab = "IMDb Rating",
        main="Boxplot of IMDb Rating vs. Lead Gender")
# store variable as a factor in dataframe
imdb.data$LeadGender <- as.factor(imdb.data$LeadGender)

# 11. Lead Race
# see how many films had a lead actor of different races
table(imdb.data$LeadRace)
# Asian    Black     Latino   Pacific Islander    White
# 170       31         27            2             770
# look at spread of ratings for different races (Figure 13)
boxplot(imdb.data$IMDbRating ~ imdb.data$LeadRace,
        xlab="Lead Actor's Race",
        names=c("Asian","Black","Latino","Pac. Isl.","White"),
        main="Boxplot of IMDb Rating vs. Lead Race")
# change LeadRace to White, Asian, Black, or Latino/Pac.Isl.
race <- rep(NA, 1000)
for(i in 1:1000){
  if     (imdb.data$LeadRace[i]=="White"){race[i]<-"White"}
  else if(imdb.data$LeadRace[i]=="Asian"){race[i]<-"Asian"}
  else if(imdb.data$LeadRace[i]=="Black"){race[i]<-"Black"}
  else if(imdb.data$LeadRace[i]=="Latino" |
          imdb.data$LeadRace[i]=="Pacific Islander")
          {race[i]<-"Latino/Pac.Isl."}
}
# store variable as a factor in dataframe
imdb.data$LeadRace <- as.factor(race)

#####################################################################
# fit models to predict IMDbRating on training data
# start at seed 12, then use seed 24, then seed 42, then seed 22
set.seed(12)
set.seed(24)
set.seed(42)
set.seed(22)
train.rows <- sample(1:1000, 750)
train.data <- imdb.data[train.rows,]
test.data <- imdb.data[-train.rows,]

# 1. Linear Regression
# fit linear regression model
lm.mod <- lm(IMDbRating ~ Genre1 + Duration + AgeRating + Language +
             ReleaseDecade + MetacriticScore + LeadGender + LeadRace,
```

```
             data=train.data)
summary(lm.mod) # look at parameter estimates and p-values
lm.mod.preds <- predict(lm.mod,na.omit(test.data)) # predict for test
mean((lm.mod.preds - na.omit(test.data)$IMDbRating)^2) # test MSE

# 2. Ridge Regression
library(glmnet)
# set up x and y matrices for training data
train.x <- model.matrix(IMDbRating ~ Genre1 + Duration + AgeRating +
                          ReleaseDecade + Language + MetacriticScore +
                          LeadGender + LeadRace, data=train.data)
train.y <- na.omit(train.data)$IMDbRating
set.seed(12)
set.seed(24)
set.seed(42)
set.seed(22)
grid <- 10^seq(1,-5,length=500) # grid of possible lambda values
# determine best value of lambda for this dataset
ridge.cv.out <- cv.glmnet(train.x, train.y, alpha=0, lambda=grid)
ridge.bestlam <- ridge.cv.out$lambda.min
# run ridge model on training set with best lambda
ridge.mod <- glmnet(train.x, train.y, alpha=0, lambda=ridge.bestlam,
                    thresh = 1e-15)
# set up x and y matrices for test data
test.x <- model.matrix(IMDbRating ~ Genre1 + Duration + AgeRating +
                       ReleaseDecade + Language + MetacriticScore +
                       LeadGender + LeadRace, data=test.data)
test.y <- na.omit(test.data)$IMDbRating
# predict IMDbRating and calculate MSE for test data
ridge.pred <- predict(ridge.mod, s=ridge.bestlam, newx=test.x)
mean((ridge.pred - test.y)^2)
ridge.mod$beta[order(ridge.mod$beta),] # parameter estimates

# 3. Lasso Regression
set.seed(12)
set.seed(24)
set.seed(42)
set.seed(22)
# determine best value of lambda for this dataset
lasso.cv.out <- cv.glmnet(train.x, train.y, alpha=1, lambda=grid)
lasso.bestlam <- lasso.cv.out$lambda.min
# run lasso model on training data
lasso.mod <- glmnet(train.x, train.y, alpha=1, lambda=lasso.bestlam,
                    thresh = 1e-15)
# predict IMDbRating and calculate MSE for test data
```

```
lasso.pred <- predict(lasso.mod, s=lasso.bestlam, newx=test.x)
mean((lasso.pred - test.y)^2)
lasso.mod$beta[order(lasso.mod$beta),] # parameter estimates

# 4. Best Subset Selection
library(leaps)
set.seed(12)
set.seed(24)
set.seed(42)
set.seed(22)
# see which 10 variables are determined to be most important
bss.mod <- regsubsets(IMDbRating ~ Genre1 + Duration + AgeRating +
                      ReleaseDecade + Language + MetacriticScore +
                      LeadGender + LeadRace, data=train.data,
                      nvmax = 10)
summary(bss.mod)
# fit the best 3 variable model and make predictions for test data
bss.lm.mod <- lm(IMDbRating ~ MetacriticScore + Duration +
                 ReleaseDecade, data=train.data)
summary(bss.lm.mod)
bss.lm.mod.preds <- predict(bss.lm.mod, na.omit(test.data))
mean((bss.lm.mod.preds - na.omit(test.data)$IMDbRating)^2) #test MSE

# 5. Regression Trees and Random Forest
library(rpart)
library(rpart.plot)
library(randomForest)
set.seed(12)
set.seed(24)
set.seed(42)
set.seed(22)
# make a tree from training data, prune it, and plot it (Figure 14)
tree <- rpart(IMDbRating ~ Genre1 + Duration + AgeRating +
              ReleaseDecade + Language + MetacriticScore + LeadGender
              + LeadRace, data=train.data, method='anova',
              control=rpart.control(minsplit=50,cp=.01,xval=10))
tree.pruned <- prune(tree, cp=.01)
prp(tree.pruned, faclen=0, extra=1, roundint=F, digits=5)
# predict IMDbRating for test data and find test MSE
tree.preds <- predict(tree.pruned, test.data)
mean((tree.preds - test.data$IMDbRating)^2)

set.seed(12)
set.seed(24)
set.seed(42)
```

```
set.seed(22)
# run randomForest, trying 5 variables at each split
tree.RF <- randomForest(IMDbRating ~ Genre1 + Duration + AgeRating +
                        ReleaseDecade + Language + MetacriticScore +
                        LeadGender + LeadRace, mtry = 5,
                        data=na.omit(train.data), importance=TRUE)
# see which variables were deemed most important (Figure 15)
importance(tree.RF); varImpPlot(tree.RF)
# predict IMDbRating for test data and find test MSE
tree.RF.preds <- predict(tree.RF, na.omit(test.data))
mean((tree.RF.preds - na.omit(test.data)$IMDbRating)^2)

#################################################################
# try to fit models to new test data
train.data <- imdb.data
test.data<-read_excel("C:/Users/maura/Documents/IMDb_Data_New.xlsx")
test.data$AgeRating <- as.factor(test.data$AgeRating)
test.data$ReleaseDecade <- as.factor(test.data$ReleaseDecade)
test.data$Genre1 <- as.factor(test.data$Genre1)
test.data$Language <- as.factor(test.data$Language)
test.data$LeadGender <- as.factor(test.data$LeadGender)
test.data$LeadRace <- as.factor(test.data$LeadRace)
test.data$Duration <- scale(test.data$Duration)
test.data$MetacriticScore <- scale(test.data$MetacriticScore)

# re-run linear regression, best subset selection, and regression
# trees using the above code
# get errors when trying to re-run ridge, lasso, and random forest
# using the above code

# visualize standard linear reg. model's predictions (Figure 16)
plot(test.data$IMDbRating, lm.mod.preds, xlab="Actual Test Ratings",
     ylab = "Predicted Test Ratings", pch=20,
     main = "Predicted vs. Actual IMDbRatings for Test Data")
abline(0,1) # add the line y=x to the plot
```

Works Cited

"About Metascores: How We Create the Metascore Magic." *Metacritic.com*, https://www.metacritic.com/about-metascores. Accessed 20 April 2023.

"Frequent questions · Letterboxd." *Letterboxd.com*, https://letterboxd.com/about/faq/. Accessed 7 May 2023.

Gohil, Digvijaysinh. "IMDb Dataset Top-Rated films 1898-2022." *Kaggle.com*, 14 June 2022, https://www.kaggle.com/datasets/digvijaysinhgohil/imdb-dataset-toprated-films-18982022?resource=download. Accessed 2 April 2023.

"History of Ratings." *FilmRatings.com*, https://www.filmratings.com/History. Accessed 20 April 2023.

"IMDb Help Center." *IMDb.com*, https://help.imdb.com/imdb.

– "How do I submit my rating on IMDb?" Accessed 7 April 2023. https://help.imdb.com/article/imdb/track-movies-tv/how-do-i-submit-my-rating-on-imdb/G9R8NF943K39DQDT?ref_=helpsect_cons_2_4#.

– "Ratings FAQ." Accessed 7 April 2023. https://help.imdb.com/article/imdb/track-movies-tv/ratings-faq/G67Y87TFYYP6TWAV?ref_=helpart_nav_5#.

– "What is IMDb?" Accessed 7 April 2023. https://help.imdb.com/article/imdb/general-information/what-is-imdb/G836CY29Z4SGNMK5?ref_=helpsect_cons_1_1#.