IMDB Data Mining Project

Richard Joerger
Rochester Institute of Technology
raj2348@g.rit.edu

Yifei Sun
Rochester Institute of Technology
ys8800@g.rit.edu

Ajeeta Khatri
Rochester Institute of Technology
ak6038@g.rit.edu

Zhuo Liu
Rochester Institute of Technology
zl9901@g.rit.edu

1. OVERVIEW

The purpose of this paper is to discuss our final implementation of our data mining component. We changed our data set, we used new features and incorporated some old ones, and we trained and analyzed our data. Our models preformed pretty well if one were to just look at accuracy but it doesn't tell the full picture. We preformed well in one category, the larger of the two subsets.

2. MOTIVATION

The rating of a movie can heavily affect the performance of said movie. If people hear a movie sucks, they're less likely to go and watch the movie and therefore the total revenue produced by that film will drop, meaning the studio makes less money. So by being able to determine if a movie will be rated well, and therefore will be more likely to preform well, before pouring millions of dollars into production can save studios from having movies flop and losing money on an investment. Obviously, like any form of art, risks are good so this wouldn't be the end all and be all of making a movie but it would be a huge benefit when it would come to estimating the performance of a very formulaic film.

3. DATA SET CHANGE

The original data set we had in mind was the same data set that we used for our data management section, the IMDb archives. We were reminded quite quickly about how messy that data set is. We were also recommended to go out and try to find more data which could be used in tandem with our existing data set to get a greater number of attributes to try and use as additional features. Instead we've decided to switch data sets entirely. The main reason for this is that the data set we're switching to, The Movie Database data set, has a greater number of attributes. Even though the data set is much smaller, with only 5000 entries, it is also readily apparent that the data is organized in a far more logical fashion which should also help us avoid getting stuck in a mess of cleaning and preprocessing our data and instead allow us to focus on the actual mining portion. The data set is very similar to the IMDb data set in that it has information about the title, the original language of the film, release date, run time, average rating, cast and production information, and vote count. What we believe makes this data set better is the keyword section, popularity, production company, budget, and revenue attributes. These added attributes provide us with a different domain to work with when it comes to actually classifying a movie with it's estimated rating.

4. DATA PREPARATION

In order to analyze and use the dataset we choose, we need to preprocess the data. We converted the original data into the way we want. Because in the original data, many attributes are stored in the form of json data.

4.1 Data Importing

At first, We need import the dataset in R from csv file.

```
setwd(" /tmdb-movie-metadata")
movie = read_csv("tmdb_5000_movies.csv",
col_names = TRUE, na = "NA")
credits = read_csv("tmdb_5000_credits.csv",
col_names = TRUE,na = "NA")
```

4.2 Data Cleaning and Arrangement

Since we observed that there are bad data in the database, data cleaning have to be done. The first part of data cleaning involves removal of spurious characters (Â) from a the movie title, genre and plot keyword columns. This might be because we have scrapped the data from the net.

```
movie$title <- (sapply(movie$title,gsub,pattern =
"\\Â",replacement = ""))
```

Next step included removing duplicate data. Duplicate data will skew our analysis hence needs to be removed.

```
movie <- movie[!duplicated(movie$title), ]
```

I tried creating different dataframes for extracting data from the json object. I use jsonlite library to extract the data. Then created comma seperated columns for 'keywords', 'genre', 'production_countries', 'production_companies', 'spoken_languages' for each movie object and combined them in the main movie data. You can check the code in our script file. Here shows how we create a genre dataframe.

```
genres <- movie %>% filter(nchar(genres) > 2) %>%
mutate( js = lapply(genres, fromJSON)) %>%
unnest(js,.names_repair = "check_unique") %>%
select(id, title, genres = name)
genres <- aggregate(genres  .,data = genres, paste,
collapse = ",")
```

In the end, we combine these columns in the main movie data.

```
movies <- subset(movie, select = -c(genres, keywords,
production_companies, production_countries, spoken_languages))
```

```
movies <- movies %>%
full_join(keywords, by = c("id", "title")) %>%
full_join(genres, by = c("id", "title")) %>%
full_join(production_companies, by = c("id", "title")) %>%
full_join(production_countries, by = c("id", "title")) %>%
full_join(spoken_languages, by = c("id", "title"))
```

5. DISCRETIZATION

Often data are given in the form of continuous values. If their number is huge, model building for such data can be difficult. Moreover, many data mining algorithms operate only in discrete search or variable space. For instance, decision trees typically divide the values of a variable into two parts according to an appropriate threshold value. The goal of discretization is to reduce the number of values a continuous variable assumes by grouping them into a number of intervals or bins. The discretization is an important process in Naive Bayes Model. Because the disadvantage of Naive Bayes is that if one of the conditional probability is zero, the the entire expression becomes zero.

5.1 Equal-frequency discretization algorithm

The algorithm determines the minimum and maximum values of the variable, sorts all values in ascending order, and divides the range into a user-defined number of intervals, in such a way that every interval contains the equal number of sorted values.

5.2 Implementation in R

We use package infotheo to discretize features. In order to make sure the number of intervals for each variable should not be smaller than the number of classes, we set 10 bins/intervals by default.

```
feature <- discretize(data,"equalfreq",num_of_bins)
```

6. REVISED DATA SET CONJECTURES

One of the things that happens as you begin to understand a data set is you start connecting the dots between the different aspects of the data. For example, since time has gone on more people have probably engaged with newer entertainment products which means that more people will have a greater variety of opinions and therefore those entertainment products will have a score that is far closer to the middle. Additionally, we believe there will be a correlation between average rating and various other factors. Looking at figure 1 we can see the distribution that shows the relationship between different attributes of the dataset. Further in figure 2. we can observe that the Count of the movies is normally distributed across Average vote(rating) with a mean of around 6. In figure 4 we see a very interesting relation between the popularity and vote count for various titles. We can also observe the relationship between the avg ratings for the movies based on a particular original language. Finally we also see the summary of the entire dataset.

7. DATA MINING APPROACH

What we are aiming to mine from our data is the rating of a movie given some set of features. The features we're looking to use are: the number of movies the top 5 actors have acted in, the number of movies the director has directed, the number of movies the producer has produced, the budget of the film, the revenue of the film, the popularity of the film. If we're feeling particularly ambitious we may even try to quantize the vector attribute to better understand how genre plays a roll in the rating of the movie.

We will begin by quantizing all of the data which we believe will help us to better predict what the rating of the movie will be. From there, we will split up the data into three sets. The first and the largest will be the training data set. This will be used to train our model and do some early stage testing to verify that our model, and indirectly our features, are capable of predicting the rating of a movie. The next set of data will be the validation set. This will be used to fine tune our model with new data which it hasn't seen before. The last set of data will be the testing data. This is the data we will use to asses the quality of our model for data which it hasn't seen before. Our approach for assessing the performance of our model will be rather straight forward. We will calculate the accuracy of the model, the F1 score, precision, and recall. We believe these metrics will provide us with more than enough information to guarantee that our model not only predicts values that are right but that it does so consistently without making massive errors. Up to this point we haven't discussed which model we plan to use so we've

7.1 Feature Selection

The features we selected were: runtime, budget, revenue, popularity, the average experience of all directors which worked on the film, the average experience of all producers associated with the film, the average acting experience of all the actors in the film, the language of the film. We also tried training with a feature that was the ratio of revenue and budget as that would be a good indicator of the monetary success of the film but there wasn't a significant difference in the performance of our classifiers. We ended up picking these features as we believed they would be indicative of the success of the film. Movies produced by a popular producer, directed by a director who has directed many films, and has actors which have been in many films, would be more likely to preform better and therefore be more likely to have a consistent rating. Budget also plays an important role, the more money you have the more you can spend on actors, set design,

8. TRAINED MODELS

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resources costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Compared to other algorithms decision tree requires less effort for data preparation during pre-processing. A decision tree does not require normalization and scaling of data. Missing values in the data also does not affect the process of building decision tree to any considerable extent. A decision tree model is very intuitive an easy to explain to technical teams as well as stakeholders. But a small change in the data can cause a large change in the structure of the decision tree causing instability. For a decision tree sometimes calculation can go far more complex compared to logistic regression. It often involves higher time to train the model so it is relatively expensive as complexity and time taken is more. Decision tree is inadequate for applying regression and predicting continuous values.

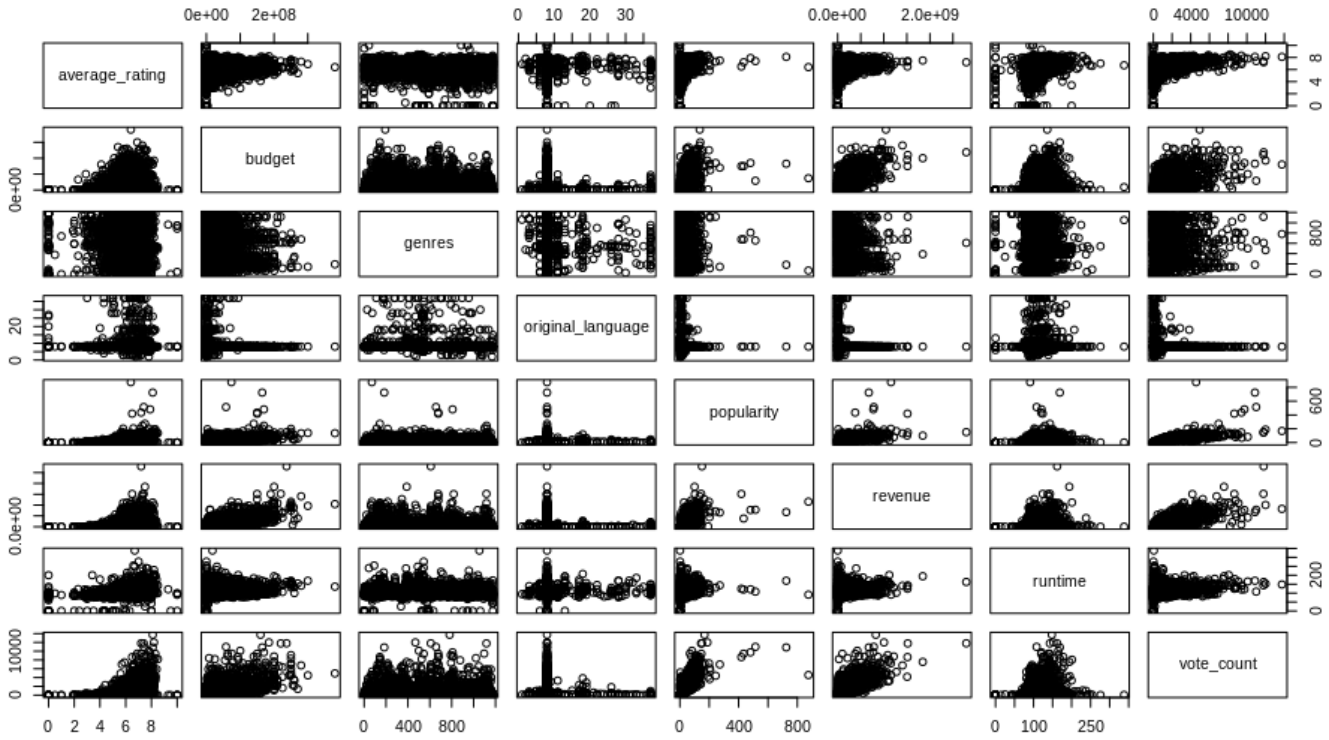

Figure 1: Pairs of comparison

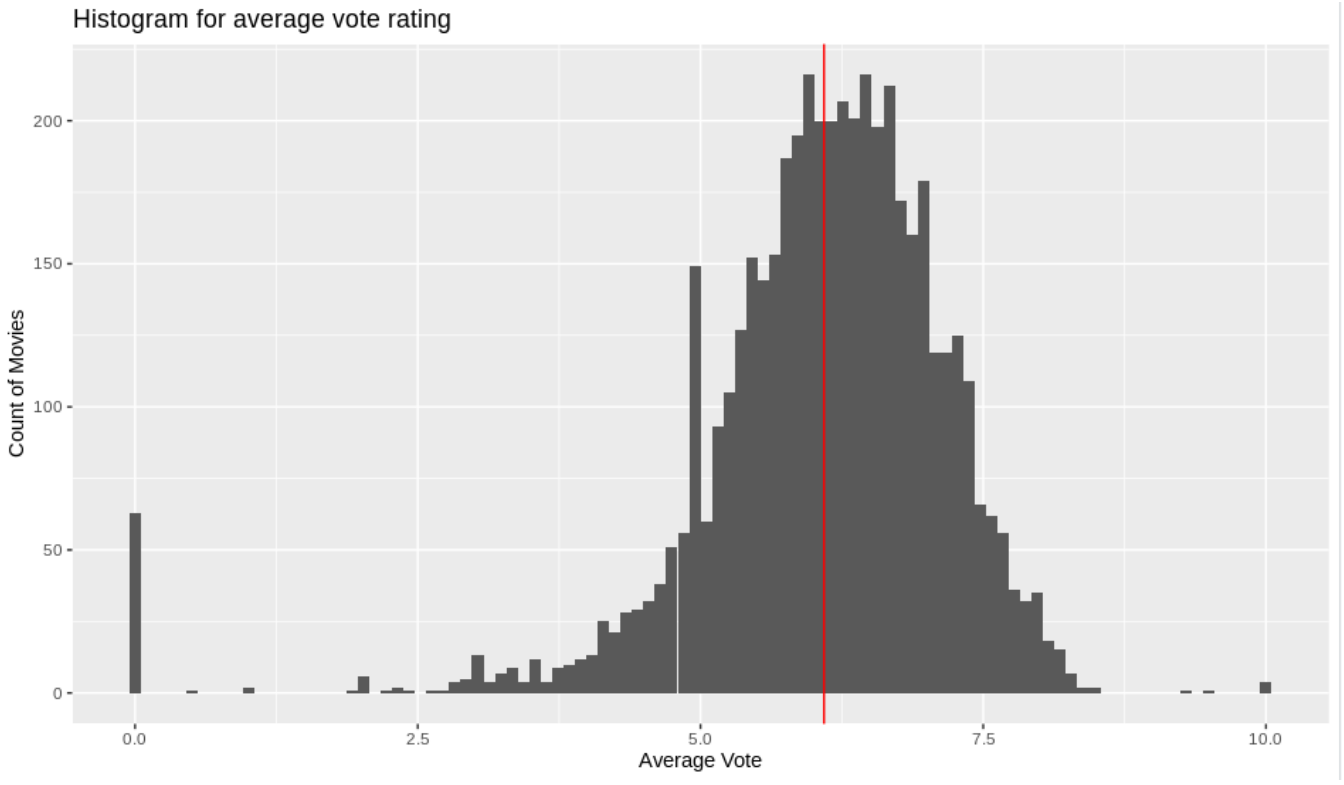

Figure 2: Histogram for average vote rating

A Naive Bayes is a simple probabilistic classifier which classifies using A Posteriori decision rule in a Bayesian set-

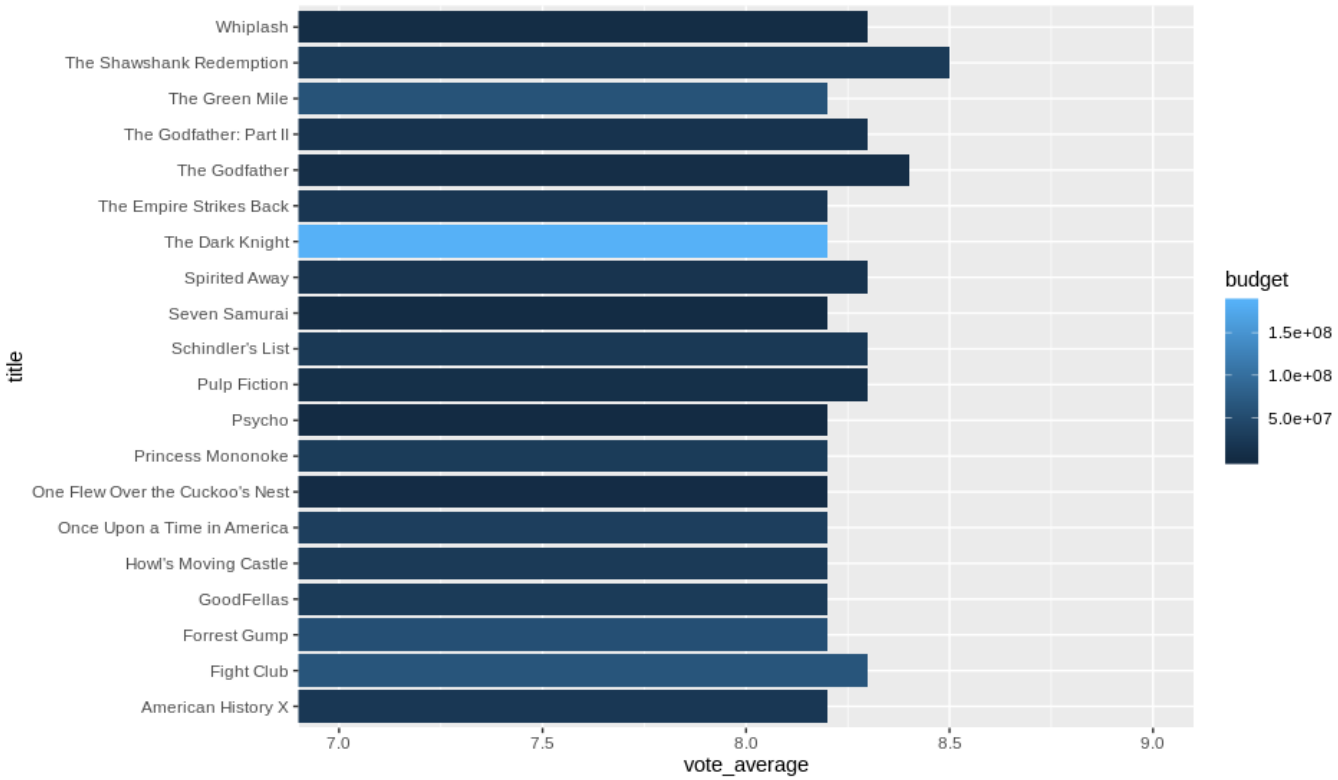

Figure 3: Plot for average vote rating and budget for titles

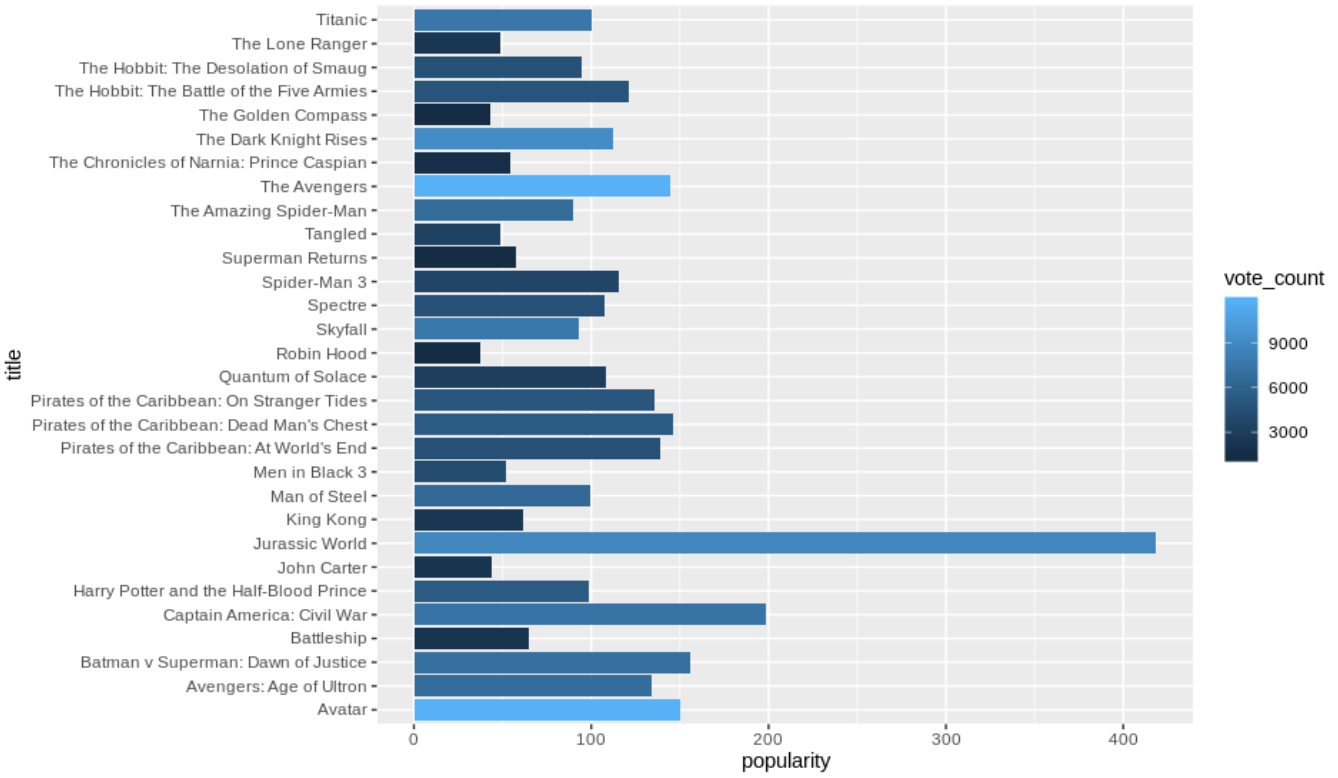

Figure 4: Plot for vote count and popularity for titles

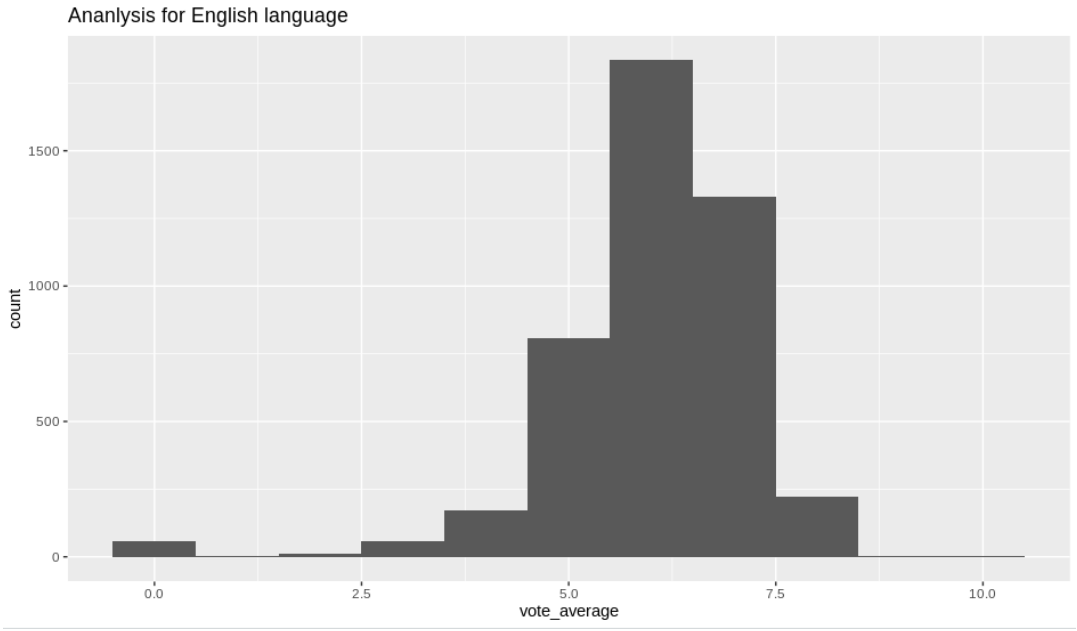

Figure 5: Plot for average vote and for titles in English language

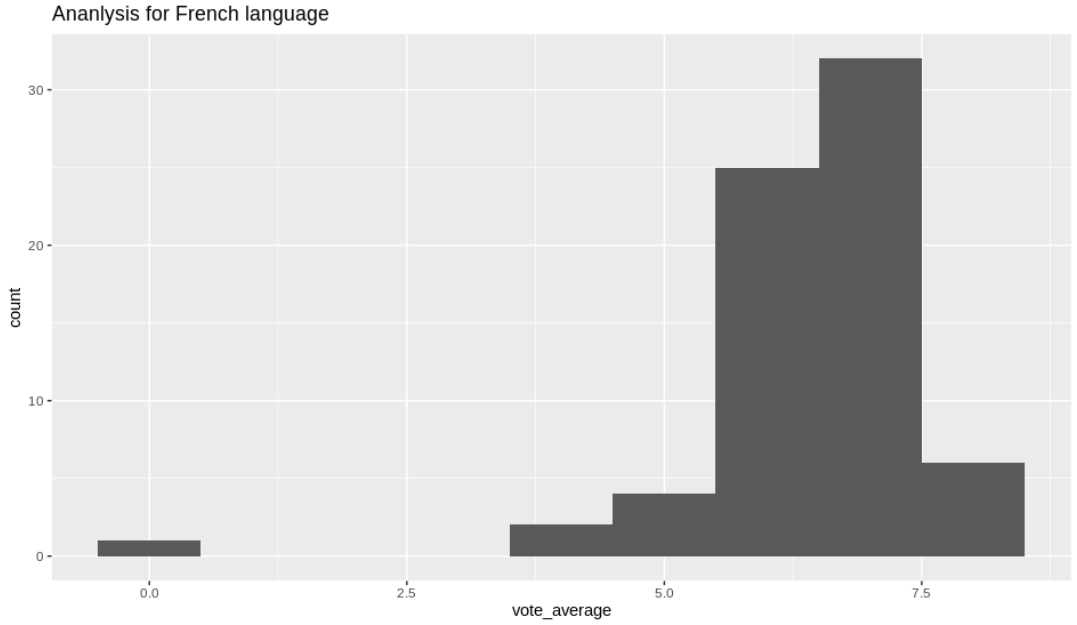

Figure 6: Plot for average vote and for titles in French language

ting. A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

K-Nearest Neighbors works by taking the euclidean distance between K neighbors of your current node and then clusters that node into a particular group. The label for the new point is the cluster which it is closest to. KNN is great in that is rather simple but is very powerful. By grouping data together by using the distance between features it is rather easy to visualize where the clusters are, assuming the feature space is in a dimension a human can comprehend.

Naive Bayes classifiers, KNN, and decision trees all operate by using different mechanisms. Naive Bayes is all about using the knowledge derived from existing data to predict what the label of the next value will be. KNN works by taking the distances and then grouping them based on their closest cluster. Decision trees work by looking at how much information can be gained from a particular feature and then splitting based on the hierarchy of information gain. We opted to use multiple models for testing as in the end we can pick the highest preforming model and use that model to make our decision.

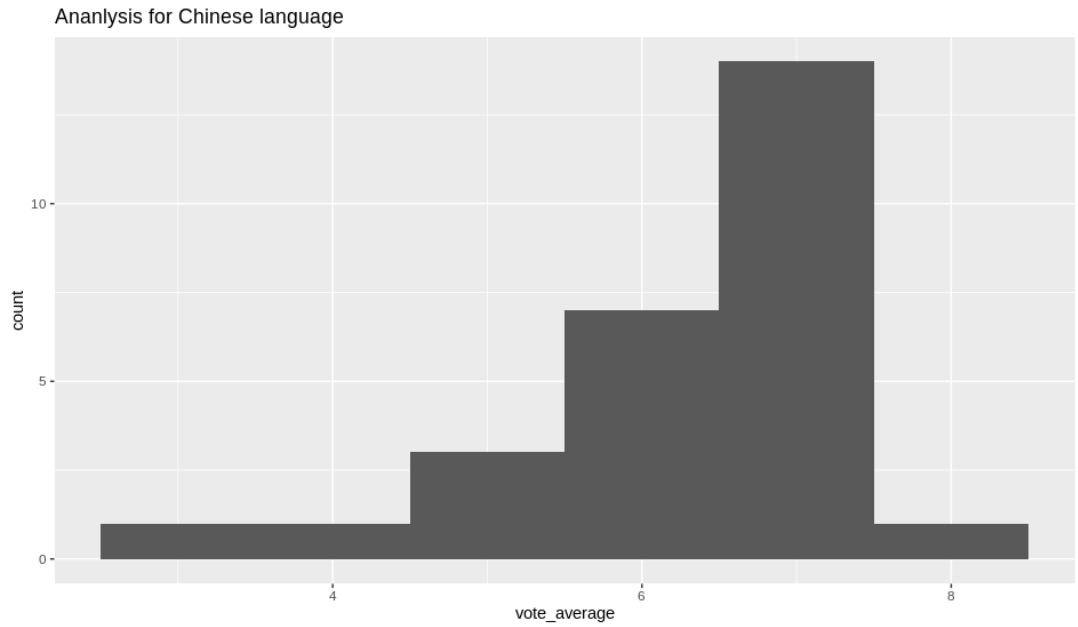

Figure 7: Plot for average vote and for titles in Chinese language

```
    budget             homepage              id         original_language  original_title       overview
Min.   :        0   Length:4800        Min.   :     5   Length:4800        Length:4800        Length:4800
1st Qu.:   795000   Class :character   1st Qu.:  9020   Class :character   Class :character   Class :character
Median : 15000000   Mode  :character   Median : 14633   Mode  :character   Mode  :character   Mode  :character
Mean   : 29060614                      Mean   : 57198
3rd Qu.: 40000000                      3rd Qu.: 58640
Max.   :380000000                      Max.   :459488

  popularity       release_date           revenue             runtime         status            tagline
Min.   :  0.000   Min.   :1916-09-04   Min.   :0.000e+00   Min.   :  0.0   Length:4800        Length:4800
1st Qu.:  4.668   1st Qu.:1999-07-14   1st Qu.:0.000e+00   1st Qu.: 94.0   Class :character   Class :character
Median : 12.925   Median :2005-10-01   Median :1.917e+07   Median :103.0   Mode  :character   Mode  :character
Mean   : 21.498   Mean   :2002-12-30   Mean   :8.229e+07   Mean   :106.9
3rd Qu.: 28.351   3rd Qu.:2011-02-17   3rd Qu.:9.294e+07   3rd Qu.:117.8
Max.   :875.581   Max.   :2017-02-03   Max.   :2.788e+09   Max.   :338.0
                  NA's   :1                                NA's   :2
   title            vote_average      vote_count        keywords           genres          production_companies
Length:4800        Min.   : 0.000   Min.   :    0.0   Length:4800        Length:4800        Length:4800
Class :character   1st Qu.: 5.600   1st Qu.:   54.0   Class :character   Class :character   Class :character
Mode  :character   Median : 6.200   Median :  235.5   Mode  :character   Mode  :character   Mode  :character
                   Mean   : 6.092   Mean   :  690.5
                   3rd Qu.: 6.800   3rd Qu.:  737.2
                   Max.   :10.000   Max.   :13752.0

production_countries spoken_languages
Length:4800          Length:4800
Class :character     Class :character
Mode  :character     Mode  :character
```

Figure 8: Data Summary

9. DISCUSSION

9.1 Decision Tree

Our decision tree performed quite well when using a simple binary classifier where the groups were greater than 5 or less than 5 for the rating. One of the things we noted was that when training the decision tree, it only used a subset of the features we created as the library determined that

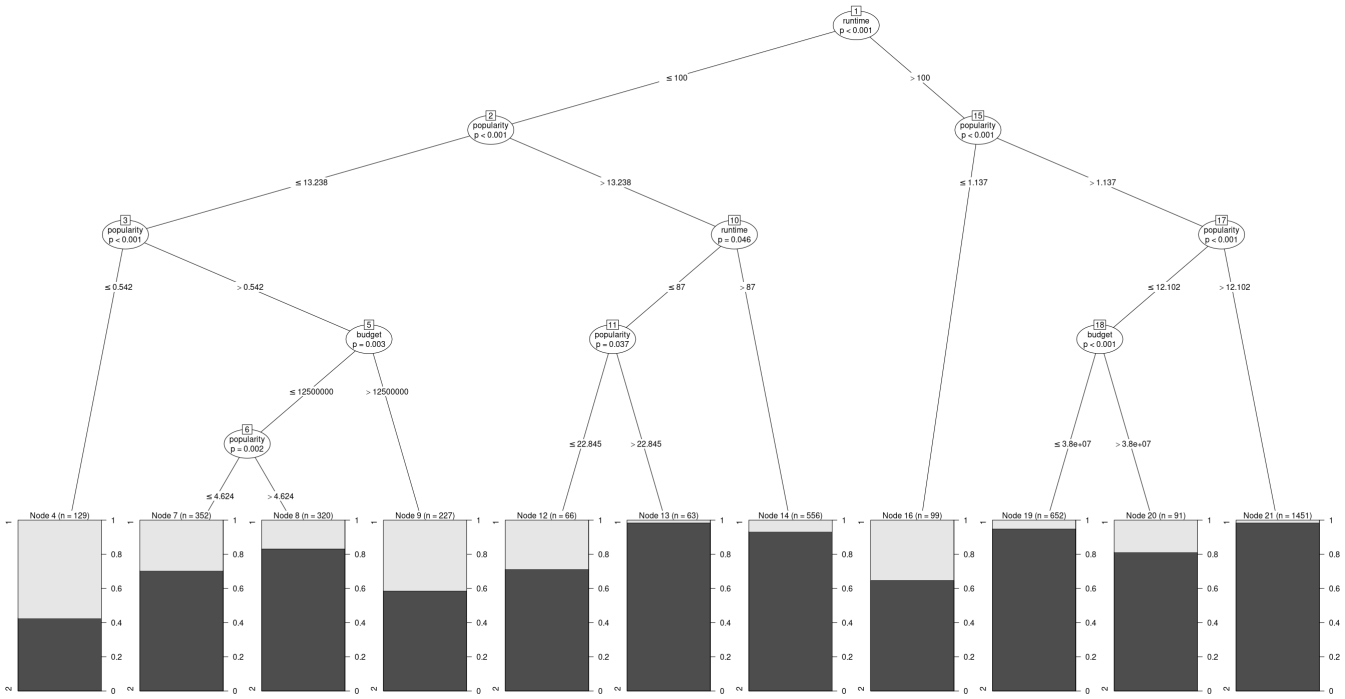

Figure 9: Decision Tree

```
           actual
predicted    1   2
        1    6   8
        2   40 392
> DT_accuracy
[1] 0.8923767
```

Figure 10: Confusion matrix and accuracy

those features were all that were needed to actual make the determination. Despite this, our accuracy is quite high at 89.3%, shown in figure 10. But, if we take a look at our F1 scores, figure 11, we see that for class 1, the less than 5 group, we preformed really poorly with an F1 score 0.2. For our second group, the greater than 5 group, our F1 score was great at 0.94.

9.2 Naive Bayes

Our Naive Bayes classifier preformed rather mediocre with an accuracy of 77.8%. Looking at our F1 scores, figure 12, we see that we again have a bias toward group 2 with an F1 score of 0.86 where as for group 1 we had an F1 score of 0.39. This means that this classifier was less accurate over all it was actually more consistent between the two groups. We made the assumption by using a Naive Bayes classifier that all of our features were independent of each other but it would seem that is not the case. Realistically though, it isn't too surprising. Most things in the "real world" aren't independent so it isn't all that surprising that in a complex system like movie making that we're running into the problem of dependency between various features.

9.3 10 Nearest Neighbors

Our final model was the 10 nearest neighbors classifier. We settled with 10 as the number of nearest neighbors after trying 3, 5, 7, and 10. We found that a K value of 10 gave us the best F1 scores even though they are not that impressive. Our accuracy was pretty good at 89.5% but again, like all of our other models, our F1 scores were very one sided. For class 1, our F1 score was 0.25 and for class 2 it was 0.94. Honestly placing it quite similar to our decision tree.

We believe a big part of the reason we're seeing such poor performance for the second group is because our data is quite heavily skewed toward class 2. Due to time limitations we couldn't get to balancing out our data set but we do believe that would have greatly helped the performance of all of our classifiers. Something which we found interesting about our classifiers is that all preformed more or less the same which we think may mean the limitation is not with the classifiers

```
   y
pre   1   2
  1   6   8
  2  40 392
[1] "-------------precision--------------------"
        1         2
0.4285714 0.9074074
[1] "-------------recall---------------------"
        1         2
0.1304348 0.9800000
[1] "-------------f1---------------------"
        1         2
0.2000000 0.9423077
[1] "--------------mean(f1)-------------------"
[1] 0.5711538
```

Figure 11: Tree Precision, Recall, and F1 Score

```
pre   1   2
  1  31  15
  2  84 316
[1] "-------------precision--------------------"
       1        2
0.673913 0.790000
[1] "-------------recall---------------------"
        1         2
0.2695652 0.9546828
[1] "-------------f1---------------------"
        1         2
0.3850932 0.8645691
[1] "--------------mean(f1)-------------------"
[1] 0.6248311
```

Figure 12: Naive Bayes Precision, Recall, and F1 Score

```
   y
pre   1   2
  1   8   9
  2  38 391
[1] "-------------precision--------------------"
        1         2
0.4705882 0.9114219
[1] "-------------recall--------------------"
       1        2
0.173913 0.977500
[1] "-------------f1--------------------"
        1         2
0.2539683 0.9433052
[1] "--------------mean(f1)-------------------"
[1] 0.5986367
```

Figure 13: 10-nearest neighbor Precision, Recall, and F1 Score

but with the data. If we had found a better way to evenly distribute the data, we think we would have ended up with far better results.

10. CHALLENGES AND LESSONS

The biggest challenge for us as a team was learning how to use R. It feels like a different programming paradigm due to the way in which objects are structured, and without the inclusion of the separate libraries, it isn't immediately obvious how you can mutate elements. So for us, learning this new paradigm was a big hurdle. As a group we spent a lot of time trying to figure out how to convert our typical iterative approach into a vector/column oriented approach. Another challenge we faced, but ultimately solved, was the lack of information in our original data set. As stated in the previous update, we had a limited number of attributes which only really encompassed a subset of the total information that constitutes making a movie. The main problem with switching data sets was having to go back and spend time to understand this new data set.

In terms of the lessons we've learned, I think we've all learned about how important API information and Google can be when it comes to solving problems. Between reading documentation and some strong Google skills we were able to solve most of the problems we ran into as a team. I think we all are a bit more comfortable with R, I think we'd all agree that it isn't our favorite language but it's nice to say we all have some experience with it. I think one of the other lesson's we've learned is how important having good data is. Our original data set would have been very hard to work with but after making the switch to the TMDB data set we realized just how bad it could have been. Plus, by switching we gained a greater number of features to work with and we were able to get an idea of how we can take raw information, like the credits of a film, and make new features which summarize them in a convenient way.

11. WORK LOAD DISTRIBUTION

The work load was split rather interestingly for this project. Richard did the initial summary with our original dataset. Once we changed, Ajeeta took over and re-did the summary for our new data set. From there, feature selection and model choices were all done as a team. We split the work load up, Yifei and Zhou handled data cleaning as well as training the Naive Bayes classifier. Ajeeta and Richard handled the decision tree and k Nearest neighbor training. Yifei and Zhou worked really hard and wrote a few functions which Ajeeta and Richard were able to use to better complete their part of the assignment. We all came together and contributed to the paper.