The Movies Dataset: predicting movie rating

# Problem Identification

### Problem Statement

Before seeing a movie, the first question most people ask is, “is it good?” Everyone wants to know how good a movie is before seeing it. What if we could predict how a movie will be reviewed based on different features of that movie?

### Goal

Using a data driven analysis and machine learning modeling techniques, we will hopefully be able to predict the rating a movie will get in order to help users determine if they want to see that movie. I will combine movie metadata and create features to improve performance of the models and at the end, will have a model that performs well in predicting movie rating

# Data

The data comes from [Kaggle](https://www.kaggle.com/rounakbanik/the-movies-dataset) and contains seven csv files. These files contain data related to the movies, credits, and ratings. The dataset contains movies from the MovieLens Dataset and only has movies released through July 2017.

### Movie Data

There are four different files with movie data. There are three files that we are not interested in using for this project. They contain links to the IMDB and TMDB websites for each movie. This data is not needed to do our prediction. The fourth file is a keyword file that has different keywords describing movies. At this moment, this file is out of scope for this project.

The data that we will be using for our analysis is titled “movies\_metadata.csv.” This dataset contains features specific to each movie. Some notable features are:

* budget
* revenue
* collection
* genre
* runtime
* title
* overview
* popularity
* production studio
* language
* release date
* vote average
* vote count

### Credits

There is one file called “credits.csv.” This file contains cast and crew data. For each movie. There are only three features, a “cast” dictionary, “crew” dictionary, and id. This data will be valuable for our prediction model.

### Ratings

There are two files related to ratings. One is a full dataset of ratings, and the second is a smaller version with less rows. Since we have the vote\_average in the movie\_metadata file, we will use that instead of merging this file to our dataset.

# Data Wrangling

Data for each file was loaded into different dataframes using Pandas. The data was loaded into three different dataframes called movies, credits, and ratings. When I loaded the movies dataframe csv file, I noticed the movie ID field was “id” but in the other files it is “movieid.” Since I plan on merging these files, I changed the “id” column name to “movieid” for the movies dataframe. I then evaluated the data by looking at null values. I saw that the features that had null values made sense and they weren’t necessarily going to be an issue for our analysis. Not every movie belongs to a collection, so that makes sense.

A picture containing text

Description automatically generated

I then did a similar analysis for the credits dateframe. However, the credits dataframe had zero null values.

The next step was to merge the movies dataframe to the credits dataframe. Since we changed the “id” in the movies dataframe, we can merge to credits on “movieid.”

### Creating and Cleaning Features

After merging the dataframes the next step was to split out the values from the dictionaries in the different columns.

**“belongs\_to\_collection” and “production\_companies”**

* This feature contained a dictionary of the name of the collection and the collection’s ID
* Any movies that were not in a collection were NaN
* To split out the collection name, I did a for loop that used iterrows to loop through the dictionaries and stored the name to an empty list.
* The list with collection names was then added to the dataframe

**“genres”**

* This feature contained a dictionary of genres for each movie
* Some movie had multiple genres
* The first genre was the main genre and was where I focused
* I performed a for loop using iterrows like above.

“cast”

* The cast feature was a little more tricky
* This feature contained a dictionary with the actor name, character name, id, credit id, gender, order, and profile path
* I used the ast package to read this dictionary and identify the top 5 actors
  + Using the literal\_eval method from the ast package, I was able to read the dictionary
  + I am only interested in the top 5 actors since some movies may have a lot of actors and the top 5 actors would be the highest billed actors since they are listed in billing order
  + I used a for loop and iterrows with literal\_eval to identify the order of the actors and added a temporary dictionary to store the actor names and appended that dictionary to an empty list before the for loops.
    - This gave me a list of the top 5 actors for each movie
* To read this list better, I split the list and added a comma between each actor
* This was then added to the movies\_credits dataframe

“crew”

* The crew feature was similar to the “cast” feature
* It contained a dictionary with all of the crewmembers.
* I was only interested in the director name
* Using literal eval from the ast package, I performed a similar loop, except instead of looking for the order number, I used the “job” field in this dictionary and only stored names of people with the job of “Director.”
* This list was then added to the movies\_credits dataframe

A and B List Actors

* In order to use the actors in our modeling, I needed to find a way to make them numeric. I thought it might be interesting to see if the number of A list actors and B list actors has an effect on the rating.
* I found a list of the top 1000 actors and imported that as a new dataframe called top\_actor\_list
* This dataframe contains the rank, created, modified, name, known for, and birthdate fields
* I created two new lists that stored actors in position <= 500. This list was called a\_list\_actors
* Any actor in position greater than 500 was added to a list called b\_list\_actors

“a\_list\_sum” and “b\_list\_sum”

* Now that I have the A and B list actor lists, I need to count how many of these actors are in the lists within the movies\_credits dataframe
* I did a for loop using iterrows and added 1 to a list if there was a match. I think summed up the numbers for each movie and that number was stored in another empty list as the sum of the actors
  + I did this for both A List and B List actor lists
* These new features were added to the dataframe

Now that I have all of my features created and cleaned up, I am able to delete the old features with the dictionaries and any other fields I didn’t find necessary.

# Exploratory Data Analysis

## Outliers

After looking at the box plots (below) there seemed to be a wide range of data and some outliers. I calculated the zscore for each data column and deleted anything that was less than 3 standard deviations away. This data was saved in a new dataframe called movies\_credits\_actors\_no.

A picture containing shoji, window, building

Description automatically generated

## Correlations

I saw in the correlation plot below that there was some data that had correlations and that I would want to remove all or some of these features from my data during modeling. I noticed that the revenue feature has a correlation with vote\_count and budget. There is also a slight correlation between budget and popularity.

# Graphical user interface Description automatically generated with medium confidence

## Distribution

The distribution of the data in the vote\_average feature is spread pretty evenly, however there is a large amount of very low scores.

Chart, histogram

Description automatically generated

# Modeling and Outcome

## Modeling

I plan to use three modeling methods: linear regression, random forests regressor, and SVM. I started with logistic regression. The features I focused on for modeling was to use the popularity, vote\_count, budget, revenue, a\_list\_sum, and b\_list\_sum for the X and vote\_average for the y.

The data was split 80/20. After running the linear regression model, the score was very low. The training score was 0.0352 and the score after running on the test data was 0.2996. Since there was some correlation with revenue and budget/vote\_count, I removed the feature and reran the data. The score was even lower. The training score was 0.0347 and when run on the test data, the score was 0.0296.

The coefficient scores showed that there is a slight relationship between popularity and a\_list\_sum and b\_list\_sum and that those features may be good predictors of the movie score.

Next I ran the random forest regressor, the score on the training data without revenue was 0.588 and the score on the test data was 0.58. This model performed significantly better, but still not great.

The score for the SVR model was -0.0378. So the random forest regressor definitely was the better performing model in this situation. When the prediction was done and plotted, the predications didn’t not turn out great.

Chart, line chart

Description automatically generated

## Outcome

The modeling scores were very low and the predictions did not seem to be very accurate. This can be for a few different reasons:

* A lot of zeros outside of NaN.
* Linear approach wasn’t a great predictor because of the zeros and not a large range within the features.
* Random forest was better because it was able to group things into the right ranges, but was not specific enough.

# Future Considerations

There are some things that I would like to do to improve the predictions. I would first start by looking at the zeros. There are a lot of zero values outside of NaNs. I would want to see if these are true zeros or if the data is just not clean. Also, adding the keywords file to the dataset and bringing in that data might give more useful features that might improve the modeling. I would also look at the current features again and see if there is anywhere to create more features.