FINAL REPORT

PROJECT TITLE: Predicting Movie Rating and Building a Recommender System

PROBLEM STATEMENT

During the last few decades, with the rise of Youtube, Amazon, Netflix and many other such web services, recommender systems have taken more and more place in our lives. From e-commerce to online advertisement recommender systems are today unavoidable in our daily online journeys. In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users (items being movies to watch, text to read, products to buy or anything else depending on industries).

Recommender systems are critical in some industries as they can generate a huge amount of income when they are efficient or also be a way to stand out significantly from competitors. One of these sectors is movie industry. We will use movie dataset so the client of our project could be movie products sellers such as Amazon, Netflix, Hulu and movie production companies.

The first goal of this project is understanding the important features of movie sector and predicting a movie rate. The second goal of this project is building a recommender system to provide recommendations for different users.

DATA DESCRIPTION

For this project, I will use two datasets of movies from Kaggle. One of them contains various details about movies for our analysis. It consists of two different files as movies and credits. The data can be reached from the following link, https://www.kaggle.com/tmdb/tmdb-movie-metadata. There are 4803 movie entries.

The first dataset 'credits' contains the following features:

- movie_id - a unique identifier for each movie
- title - the title of the movie
- cast - the name of lead and supporting actors
- crew - the name of director, editor, composer, writer etc.

The second dataset 'movies' has the following features:

- budget - The budget in which the movie was made.
- genre - The genre of the movie, action, comedy, thriller etc.
- homepage - A link to the homepage of the movie.
- id - This is infact the movie_id as in the first dataset.
- keywords - The keywords or tags related to the movie.
- original_language - The language in which the movie was made.
- original_title - The title of the movie before translation or adaptation.
- overview - A brief description of the movie.
- popularity - A numeric quantity specifying the movie popularity.
- production_companies - The production house of the movie.
- production_countries - The country in which it was produced.
- release_date - The date on which it was released.

- revenue - The worldwide revenue generated by the movie.
- runtime - The running time of the movie in minutes.
- spoken_languages - The languages spoken in the movie.
- status - "Released" or "Rumored".
- tagline - Movie's tagline.
- title - Title of the movie.
- vote_average - average ratings the movie recieved.
- vote_count - the count of votes recieved.

The second dataset from Kaggle is about users. It can be reached from the following link. https://www.kaggle.com/rounakbanik/undefined#ratings_small.csv. The user id in this data set is important for collaborative filtering. The user is recommended items that people with similar tastes and preferences liked in the past. The features of this data set are:

- userId - a unique identifier for each user
- movieId - a unique identifier for each movie
- rating – the rating users gave.
- timestamp – the time of given rating.

DATA WRANGLING

First, we merged credits and movies datasets on the common column, movie id. When merged these two datasets, we built a new dataset includes 24 columns and 4803 entries. Some columns have the same information so we will drop these same columns in the following steps.

Some columns using in our data analysis such as genres, cast, crew, production companies, and production countries are in json format, so we changed these columns to string format and selected necessary variables from these json formats. For example, from the crew column, we extract director information and create a new column for directors of the movies.

We compared title and original title columns to understand what the difference is between these two columns. We see that original title is the name of movies in original language and title is the English name of movies. We will use English title of movies for our analysis so we will delete original title column.

	title_x	**original_title**
97	Shin Godzilla	シン・ゴジラ
215	Fantastic 4: Rise of the Silver Surfer	4: Rise of the Silver Surfer
235	Asterix at the Olympic Games	Astérix aux Jeux Olympiques
317	The Flowers of War	金陵十三釵
474	Evolution	Évolution

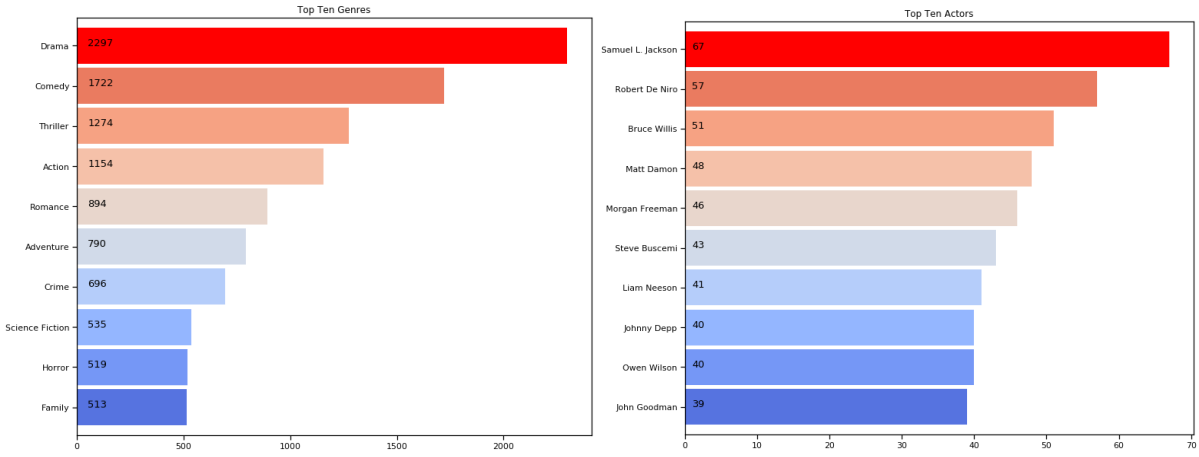
We dropped unnecessary columns such as duplicate columns. The dropped columns are 'movie_id', 'overview', 'status', 'tagline', 'homepage', 'spoken_languages', 'id', 'original_title', 'title_y'. While we merged two datasets, the duplicate name of columns changed to title_x. We again changed the title_x column name to title.

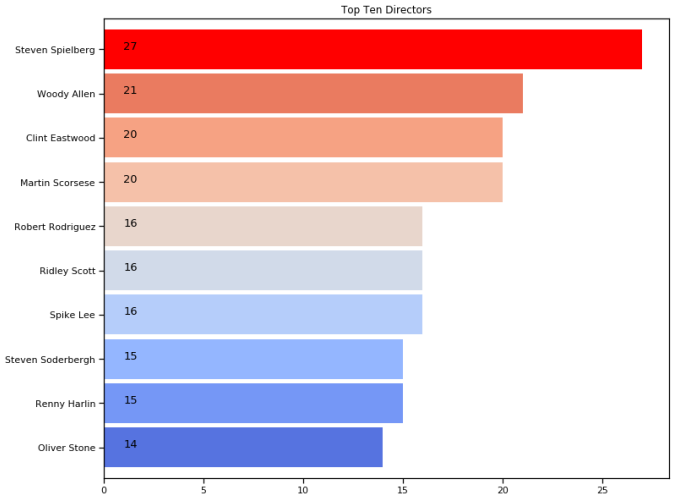
We also checked the duplicated rows and all rows are unique in the dataset. Every movie must be revenue, budget, and runtime so we checked the zero values in these columns. Revenue, budget and run time columns have 1427, 1037, and 35 zero values, respectively. This is not meaningful. However, we will keep these entries to avoid getting error while machine learning algorithms work.

After cleaning the data, we have 15 columns and 4803 rows. Title, Cast, Director, Genres, Keywords, Original Language, Production Companies and Production Countries and Release Date are categorical features. Budget, Popularity, Revenue, Runtime, Vote Average and Vote Count are numerical variables.

EXPLORATORY DATA ANALYSIS

Genre, Cast and Director



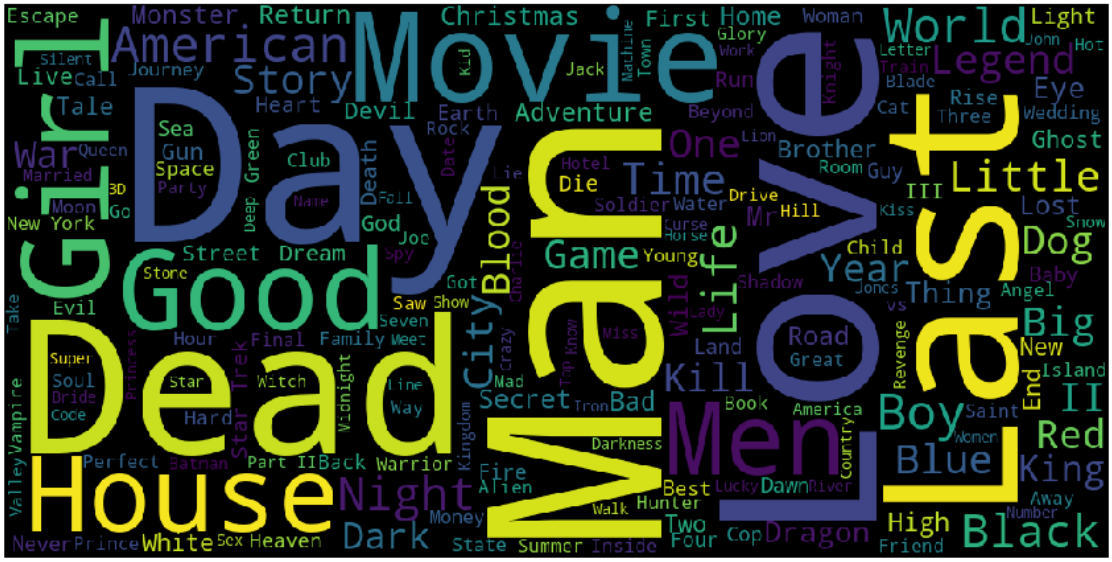


We counted the number of movies for each genre. From above graph, we can say that Drama is the most produced genre in movie sector with 2297 movies. Comedy and thriller genres follow drama.

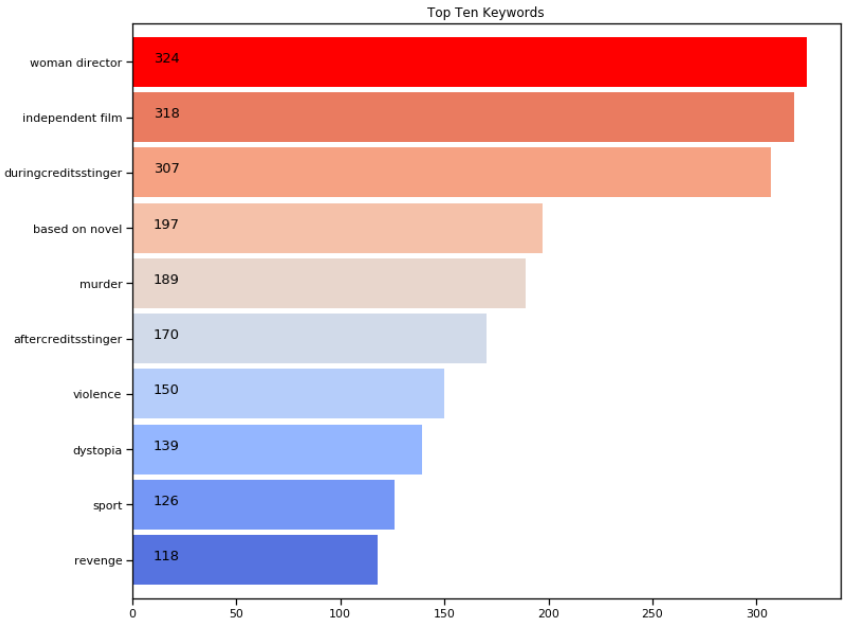
Samual L. Jackson is the first actor having roles in 67 movies. Then, Robert De Niro, Bruce Wills and Matt Damon follow him having 57, 51 and 48 roles, respectively.

From the above graph, we can say that Steven Spielberg is the most produced director with 27 movies. Then, Woody Allen follows him with 21 movies. Both Clien Esatwood and Martin Scorsese have 20 movies coming in the third place.

Movie Title and Keywords

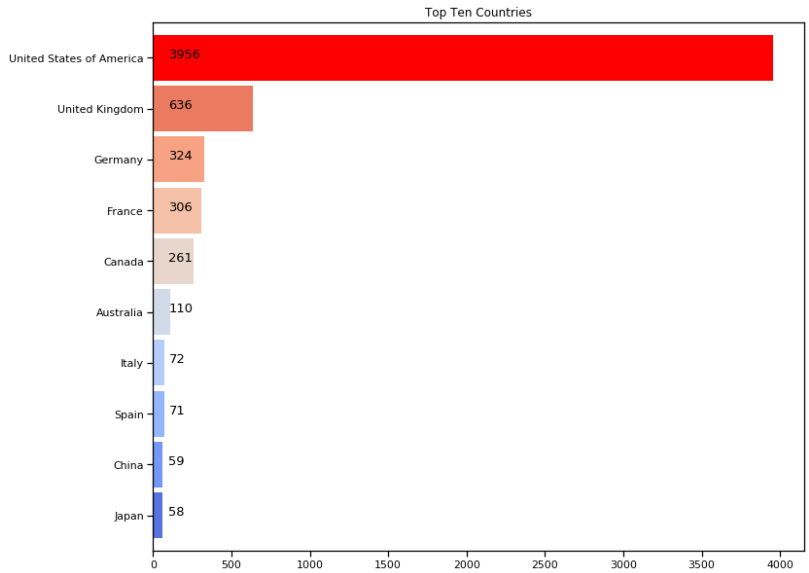


When we created a world cloud to see the frequency of words used in movie titles. It can be seen from the above word cloud that 'Man, Love, Day, Dead, Last, Girl, etc.' are the most used words in movie titles.



From above graph, we see that woman director, independent film, during credits stinger, based on novel and murder are respectively top five keywords that are used to describe movies.

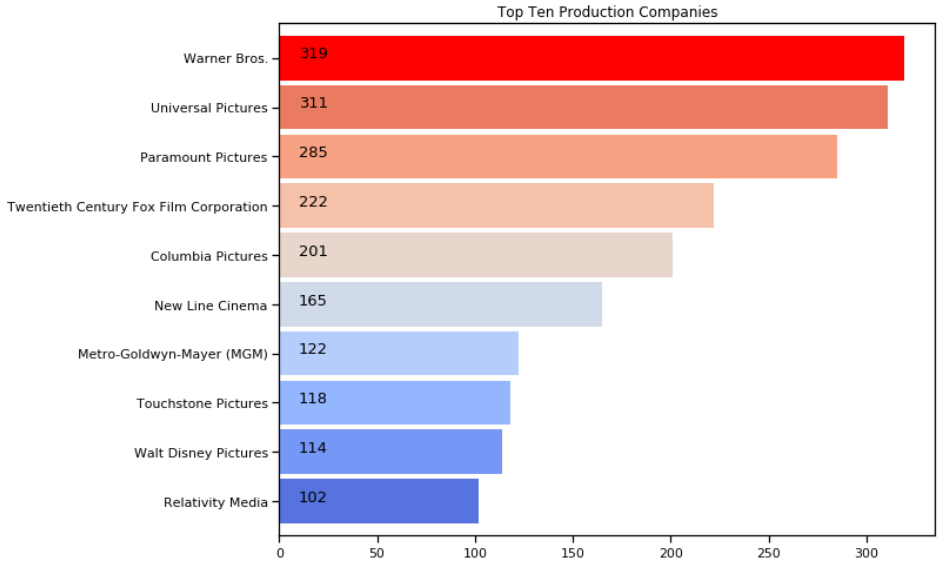
Production Countries, Companies and Original Language



Percentage of Languages	
en	0.94
fr	0.01
es	0.01
zh	0.01
de	0.01

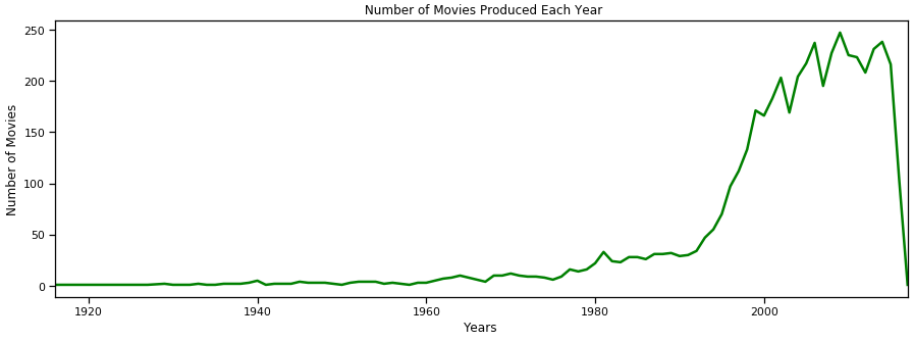
From the above graph, we can say that %61 of the movies is produced by the United States. It is the top producer country. United Kingdom is the second country producing %10 of the movies. France and Germany follow the United Kingdom having the %5 of the movies.

Most of movies are produced by United States and England that speaks English. Thus, %94 of languages of movies are English.

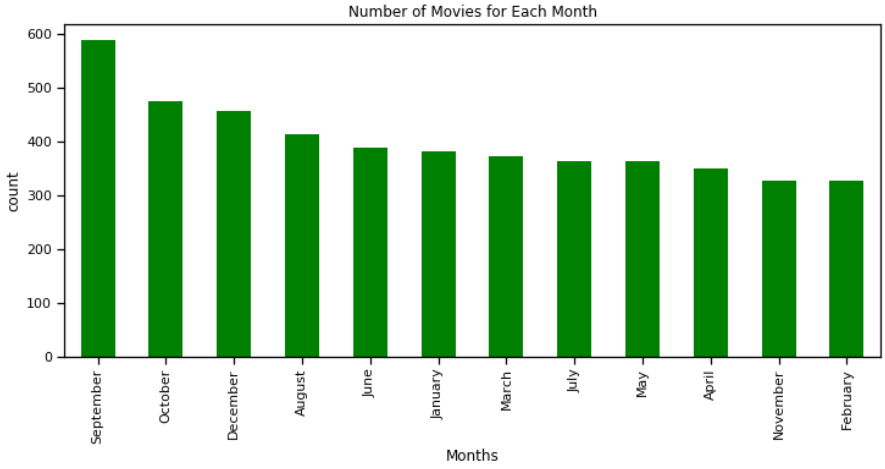


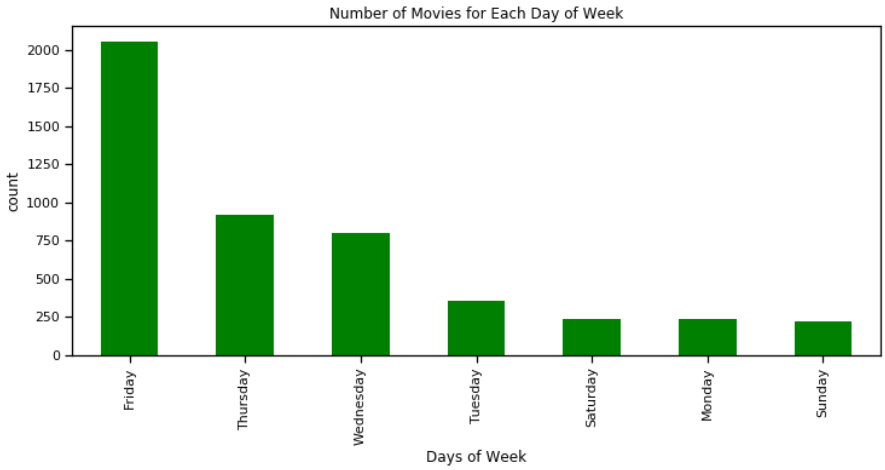
As we see from the above graph, Warner Bros., Universal Pictures, and Paramount Pictures are the top three production companies. The first most produced company, Warner Bros., produced 319 movies. The second company, Universal Pictures produced 311 number of movies.

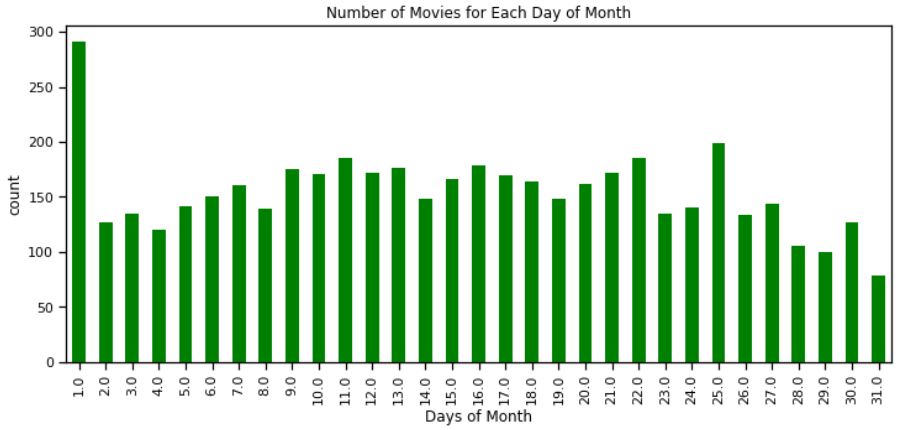
Release Date



First, we changed the release date column from object to datetime to analyze the feature. We can state from the above time series that the number of movies shows a very slightly increase after 90's. After 20's, at least about 150 movies were produced every year. As we see, the last movie in this dataset released in 2017-02-03. Also, the maximum number of movies produced in 2006.

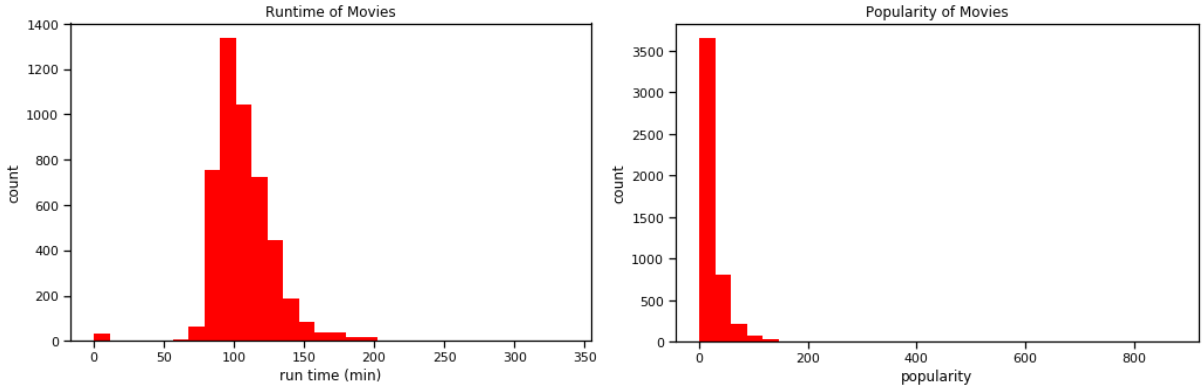






We separated months, days of weeks and the days of month of release date column to see the detailed number of movies produced. September is the top month to release a movie. Then, October and December follow September. Most of the people do not prefer to go to a movie in the summer so we can say that the movies release after the summer holiday. Another important point is about the day of the week. As we see from the days of week graph, Friday has the greatest number of movies comparing to other days because people prefer to go to a movie at weekends. The last important point is that the number of movies is greater than the other days in the first day of the month. Probably, the movies release in the beginning of the months.

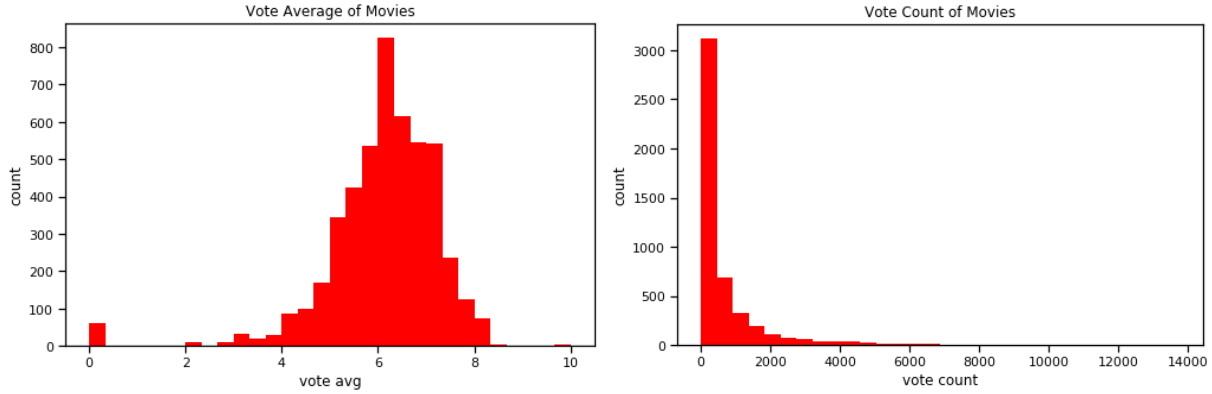
Run Time, Popularity



The shortest movie is Vessel with 14 minutes released in 2012. The longest movie is Carlos with 338 minutes released in 2010. The average run time of movies is 107.6 minutes. Most of movies run between 75 and 150 minutes.

The first popular movie is Minions with 875.58 points. Interstellar and Deadpool follow Minions with 724.24 and 514.56 points, respectively. When we analyze the distribution of the popularity, 21.49 is the average of it. %75 of the movies has popularity less than 28.31.

Vote Average and Vote Count

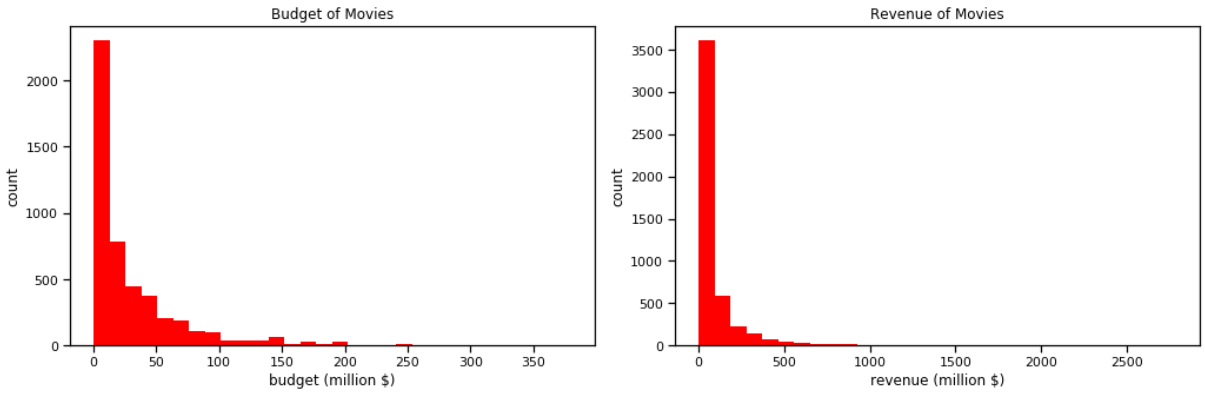


The mean of the vote average of movies is 6.09. It looks like a normal distribution. The vote average changes between 0 to 10. There are four movies having 10 vote average. These movies are Stiff Upper Lips, Me You and Five Bucks, Dancer, Texas Pop.81 and Little Big Top. However, these movies have only one or two votes, so these higher vote averages do not make any sense. If we filter the movies having more than 2000 votes, The Shawshank Redemption rating is maximum with 8.5 vote average. The Godfather comes in the second place with 8.4 rating.

The vote count feature has an exponential distribution. 62 of movies have no vote and %50 of the vote count is less than 235. Inception is the first movie having 13752 vote count. The second movie having 12002 votes is The Dark Knight. Then, Avatar follows them with 11800 votes.

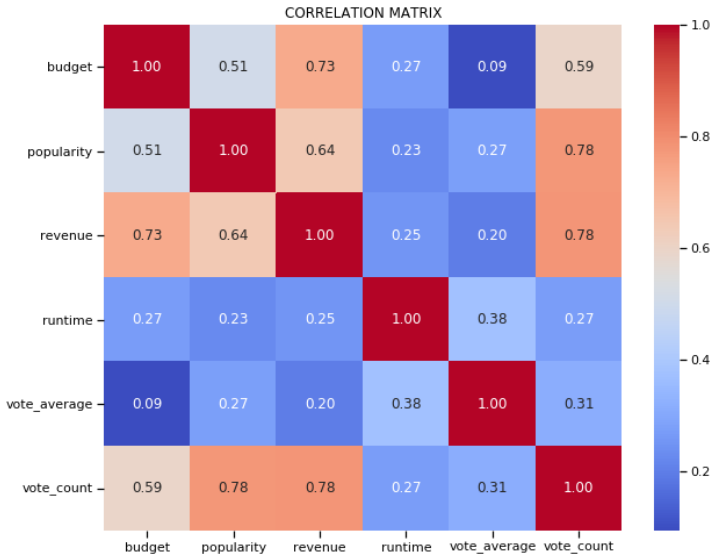
title	vote_average	vote_count	release_date
The Shawshank Redemption	8.5	8205	1994-09-23
The Godfather	8.4	5893	1972-03-14
Pulp Fiction	8.3	8428	1994-10-08
Fight Club	8.3	9413	1999-10-15
Spirited Away	8.3	3840	2001-07-20
The Godfather: Part II	8.3	3338	1974-12-20
Schindler's List	8.3	4329	1993-11-29
Whiplash	8.3	4254	2014-10-10

Budget and Revenue



Both budget and revenue of movies have an exponential distribution. They have zero values that are not meaningful, but we will not drop these values to apply machine learning algorithms. The %50 of movies' budget are less than 15 million $. Pirates of the Caribbean: On Stranger Tides, Pirates of the Caribbean: At World's End and Avengers: Age of Ultron have 380 million $, 300 million $ and 280 million $ budget, respectively. These movies are top three movies having more budget comparing other movies. %50 of movies' revenue is less than 19 million $. Avatar, Titanic, The Avengers are top three movies having more than 1.5 billion $ revenue.

Correlation of Numerical Features



We built a correlation matrix to understand the relationship between numerical features. All numerical features have positive relationship between each other. There is a strong correlation between popularity and vote count. Also, the same relationship is valid between revenue and budget. We can say that if the budget is higher for a movie, we can expect higher revenue too. In addition, if a movie received many votes, its popularity increases in terms of the number of votes.

PREDICTING VOTE AVERAGE

Before releasing a movie on the theater, the most common questions in mind is what the rating of the movie will be. Will the users like it or not? In this part, we built three machine learning algorithms to predict the average rating of a movie. These algorithms are linear regression, KNN (K nearest neighbors) and random forest regression. Also, we only used popularity, vote count, revenue, and budget features to predict vote average.

Before built our models, we split our dataset into two part as a training and test set. Also, we standardized the feature values to get meaningful result. We applied these algorithms in Scikit-Learn and found Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. A smaller RMSE means that the performance of the model is better, so we compared the RMSE of the models we built. These are the RMSE result of each model we built.

```
Linear Model RMSE: 1.142936955228411
KNN Model RMSE for k = 5 : 0.9170384581476931
KNN Model RMSE for k = 6 : 0.9071766730598588
KNN Model RMSE for k = 7 : 0.9077793127184054
KNN Model RMSE for k = 8 : 0.8989416654146611
KNN Model RMSE for k = 9 : 0.8864749464110114
KNN Model RMSE for k = 10 : 0.8819050607745806
RF RMSE for n_estimator = 50 : 0.8691340834153295
RF RMSE for n_estimator = 70 : 0.867550015139498
RF RMSE for n_estimator = 90 : 0.8644038008844245
RF RMSE for n_estimator = 110 : 0.8726479830822365
RF RMSE for n_estimator = 130 : 0.8703479095143601
RF RMSE for n_estimator = 150 : 0.8677051501449795
RF RMSE for n_estimator = 170 : 0.8650161643075953
RF RMSE for n_estimator = 190 : 0.8650259057546114
```

As a result, we got minimum RMSE of 0.864 with the performance of Random Forest with 90 number of estimators.

RECOMMENDER SYSTEM

Recommender systems are one of the most common used and easily understandable applications of data science. Lots of work has been done on this topic, the interest and demand in this area remains very high because of the rapid growth of the internet and the information overload problem. It has become necessary for online businesses to help users to deal with information overload and provide personalized recommendations, content, and services to them.

Two of the most popular ways to approach recommender systems are collaborative filtering and content-based recommendations. We applied these two methods to create our recommendation models.

Content Based Filtering

This type of filter does not involve other users. Based on what we like, the algorithm will simply pick items with similar content to recommend us. In this case, this method suggests similar items based on a particular item. This system uses item metadata, such as genre, director, description, actors, etc. for movies, to make these recommendations. The general idea behind these recommender systems is that if a person liked a particular item, he or she will also like an item that is similar to it.

What the meaning of similarity is an important question before applying the method. It does not really seem something we can quantify, but it can be measured. Cosine similarity is one of the metrics that we can use when calculating similarity, between users or contents.

We all are familiar with vectors: they can be 2D, 3D or more. We can consider 2D because it is easier to picture in our mind. The dot product between two vectors is equal to the projection of one of them on the other. Therefore, the dot product between two identical vectors (i.e. with identical components) is equal to their squared module, while if the two are perpendicular (i.e. they do not share any directions), the dot product is zero. Generally, for *n*-dimensional vectors, the dot product can be calculated as shown below.

$$\boldsymbol{u} \cdot \boldsymbol{v} = [u_1 \; u_2 \; \ldots \; u_n] \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n = \sum_{i=1}^{n} u_i v_i$$

The dot product is important when defining the similarity, as it is directly connected to it. The definition of similarity between two vectors **u** and **v** is, in fact, the ratio between their dot product and the product of their magnitudes.

$$similarity = cos(\theta) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \|\boldsymbol{v}\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \sqrt{\sum_{i=1}^{n} v_i^2}}$$

By applying the definition of similarity, this will be in fact equal to 1 if the two vectors are identical, and it will be 0 if the two are orthogonal. In other words, the similarity is a number bounded between 0 and 1 that tells us how much the two vectors are similar.

In this recommender system, the content of the movie (cast, director, keywords, genres, production companies) was used to find its similarity with other movies. Then, the movies that are most likely to be similar are recommended. We created a metadata soup, which is a string that contains all the metadata that we want to feed to our vectorizer. We used Count Vectorizer to create a count matrix and calculated the cosine similarity. The function we created below returned the most similar ten movies.

```
get_recommendation('Interstellar', cosine_sim)
```

```
4332                Silent Running
96                       Inception
239                        Gravity
635                      Apollo 13
643                   Space Cowboys
270                     The Martian
3405    Stargate: The Ark of Truth
3624                           Moon
539                     Titan A.E.
720                      Contagion
Name: title, dtype: object
```

```
get_recommendation('Frozen', cosine_sim)
```

```
89                  Wreck-It Ralph
88                      Big Hero 6
391                      Enchanted
6                          Tangled
1695                       Aladdin
1426                       Valiant
289      The Emperor's New Groove
2114         Return to Never Land
269     The Princess and the Frog
1620              Winnie the Pooh
Name: title, dtype: object
```

For example, if a user like the movie Interstellar, he or she can like Silent Running, Inception, Gravity and so on. As we see from the above table, other movies are also related with space. Their keywords and genres are very similar.

Collaborative Filtering

Collaborative filtering is a technique that can filter out items that a user might like based on reactions by similar users. It works by searching a large group of people and finding a smaller set of users with tastes like a particular user. It looks at the items they like and combines them to create a ranked list of suggestions.

One way to handle the scalability and sparsity issue created by Collaborative filtering is to leverage a latent factor model to capture the similarity between users and items. We turned the recommendation problem into an optimization problem. We can view it as how good we are in predicting the rating for items given a user. One common metric is Root Mean Square Error (RMSE). The lower the RMSE, the better the performance.

We used ‘users’ dataset that includes user id, movie id, rating, and timestamp. Also, we applied the Singular Value Decomposition (SVD) model, is a matrix decomposition method for reducing a matrix to its constituent parts to make certain subsequent matrix calculations simpler. We used the Surprise library to implement SVD. We got RMSE of about 0.90 which is more than good enough for our case.

	userId	movieId	rating	timestamp
15273	100	1	4.0	854193977
15274	100	3	4.0	854194024
15275	100	6	3.0	854194023
15276	100	7	3.0	854194024

```
svd.predict(100,1)
```

```
Prediction(uid=100, iid=1, r_ui=None, est=3.553598167371615, details={'was_impossible': False})
```

```
svd.predict(100,2)
```

```
Prediction(uid=100, iid=2, r_ui=None, est=3.2558176728417916, details={'was_impossible': False})
```

For example, we selected user with user id 100 and filtered his or her ratings for movies. For movie with id 1, we got an estimated prediction of 3.55. This rating is closed to his or her real rating for this movie. We also repeat the same process for movie with id 2 which is not voted by the user before. We got an estimated prediction of 3.25 for the movie by the same user. This recommender system does not consider what the movie is (or what it contains). It works purely based on an assigned movie id and tries to predict ratings based on how the other users predicted the movie.

Conclusion

In this project, data wrangling, data visualization, feature engineering, predictive modeling for movie vote average and recommendation engines are performed. To predictive regression modelling, linear regression, KNN and Random Forest algorithms is used. The minimum RMSE is obtained from the Random Forest model with 0.864 score.

In addition to predictive modeling, recommendation engines are built based on two different algorithms. The first method we used is content based filtering regarding with the features cast, director, keywords, genres, and production companies. We found the similarities between movies in terms of these features. The second method is collaborative filtering based on ratings by similar users. The singular value decomposition algorithm is applied for this filtering engine and about 0.90 RMSE score is obtained.

For further projects, other filtering models like hybrid can be applied to our dataset. Also, other features can be added to our content-based filtering model additionally.