Capstone Project II

Creating movie recommendation system

1.Introduction

A recommender system is a simple algorithm whose aim is to provide the most relevant information to a user by discovering patterns in a dataset. The algorithm rates the items and shows the user the items that they rate highly.
An example of recommendation in action is when you get movie recommendations by a movie company.

2.Objective

The objective of this project is to build a movie recommender system. Recommendation Systems are a type of information filtering systems as they improve the quality of search results and provides items that are more relevant to the search item or are related to the search history of the user.

3.Data Summary - General Information

We'll be building a baseline Movie Recommendation System using TMDB 5000 Movie Dataset.

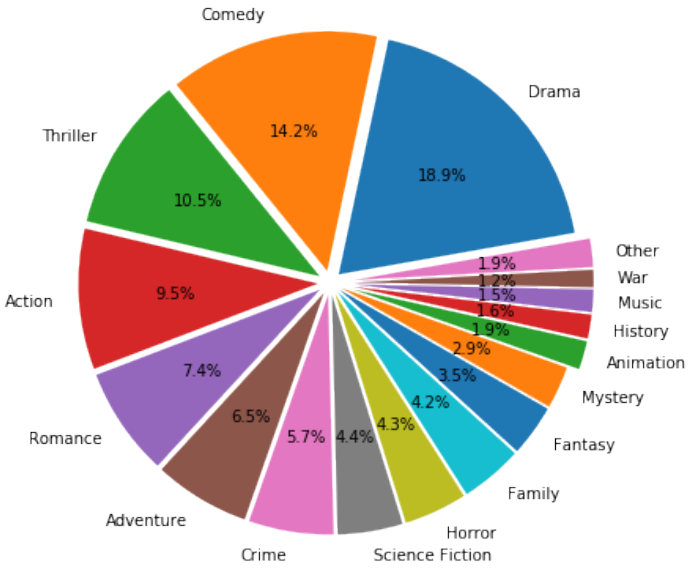
- There are 4803 movies in our datasets.
- A total of 3315117 ratings have been given.
- Every movie was assigned for some genres, among 20 genres in total.

4.Exploratory Data Analysis

In the following some data analysis are shown, like:

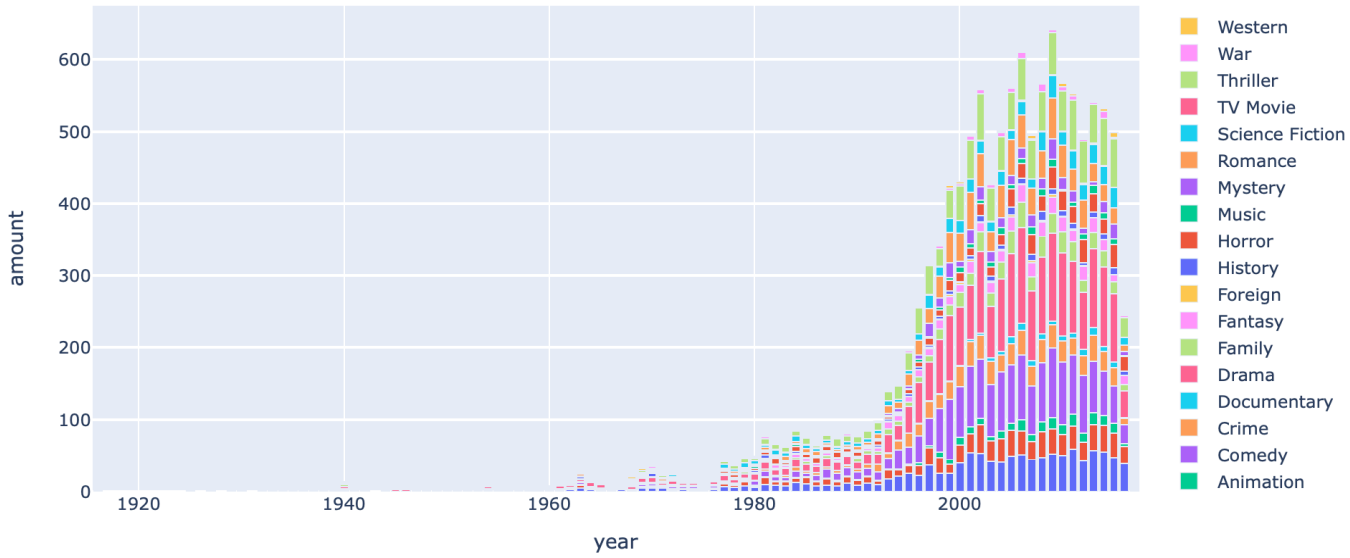
4.1 Proportion of Genres

This graph shows the proportion of the genres relatively.



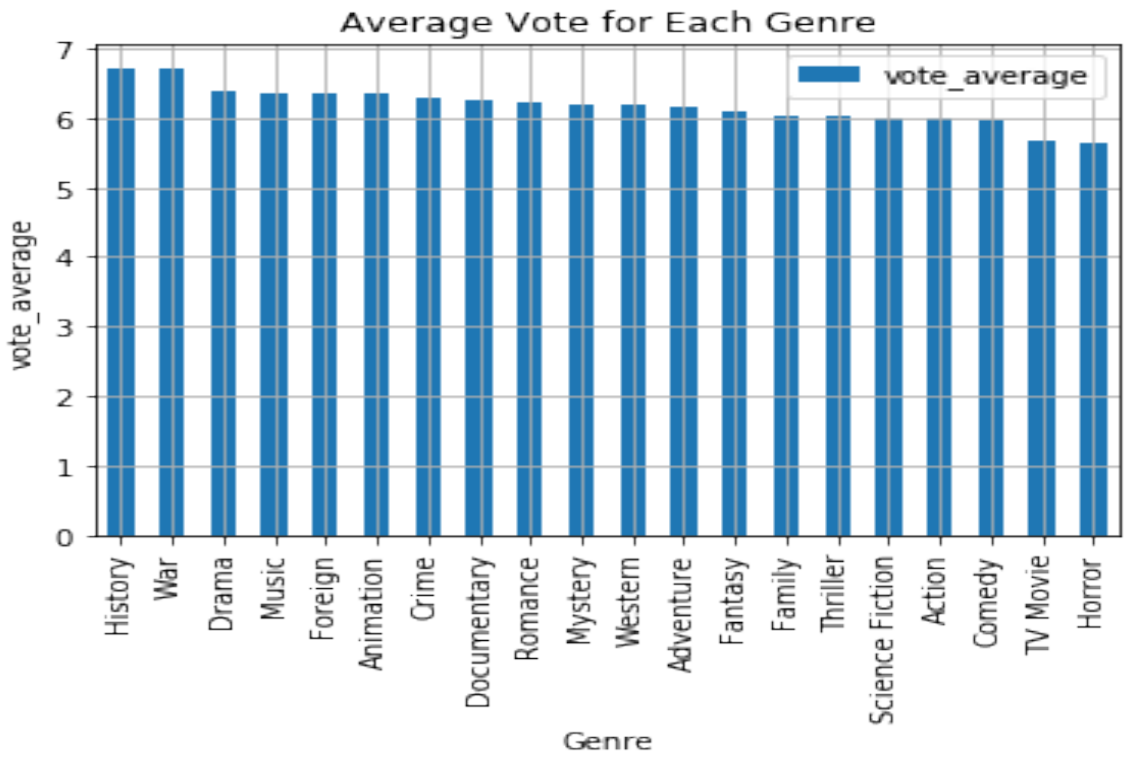
4.2 Movies amount per genres distribution by year

In this picture we made a bar plot for each year that shows the genre of the films made on the specific year.



4.3 Average Rating for Each Genre

Another good analysis was to see the average vote for each genre.
We can see which genre has the most ratings.



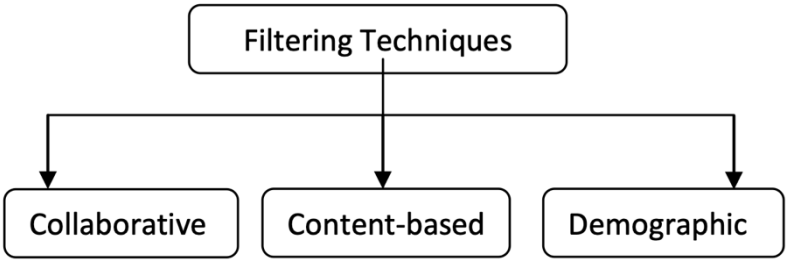
5.Different types of recommendation engines:

The most common types of recommendation systems are content-based and collaborative filtering recommender systems.
In collaborative filtering the behavior of a group of users is used to make recommendations to other users. The recommendation is based on the preference of other users. A simple example would be a recommending movie to a user based on the fact that their friend liked the movie.

There are basically three types of recommender systems:

- **Demographic Filtering**- They offer generalized recommendations to every user, based on movie popularity and/or genre. The System recommends the same movies to users with similar demographic features. Since each user is different, this approach is considered to be too simple
- **Content Based Filtering**- They suggest 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.

- **Collaborative Filtering**- This system matches persons with similar interests and provides recommendations based on this matching. Collaborative filters do not require item metadata like its content-based counterparts.



TRADEOFFS BETWEEN RECOMMENDATION TECHNIQUES

Technique	**Pluses**	**Minuses**
Collaborative Filtering (CF)	A. Can identify cross-genre niches. B. Domain knowledge nc needed. C. Adaptive: quality improves over time. D. Implicit feedback sufficient	I. New user ramp-up problem. J. New item ramp-up problem. K. "Gray Sheep" problem. L. Quality dependent on large historical dataset. M. Stability vs. plasticity problem.
Content-based Filtering (CBF)	B, C, D	I, L, M
Demographic Filtering (DF)	A, B, C	I, K, L, M N. Must gather demographic information

5.1 Demographic Filtering

This type of recommender systems suggests items based on the demographic profile of users. It can be used to identify the taste of users that belong to a certain community. Therefore, to design these systems, we need some information about users to categorize them into groups. Then, if some users in a particular group like or order an item, it is possible that the other users of this group tend to do the same. It should be noted that although it might be better to use more structured information about users, there is a trade- off between the computational complexity and the quality of demographic filtering.

The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience.
We are using the IMDb weighted rating formula.
This formula provides a true 'Bayesian estimate', which takes into account the number of votes each title has received, minimum votes required to be on the list, and the mean vote for all titles:

weighted rating (WR) = (v ÷ (v+m)) × R + (m ÷ (v+m)) × C

Where:

- R = average for the movie (mean) = (Rating)
- v = number of votes for the movie = (votes)
- m = minimum votes required
- C = the mean vote across the whole report

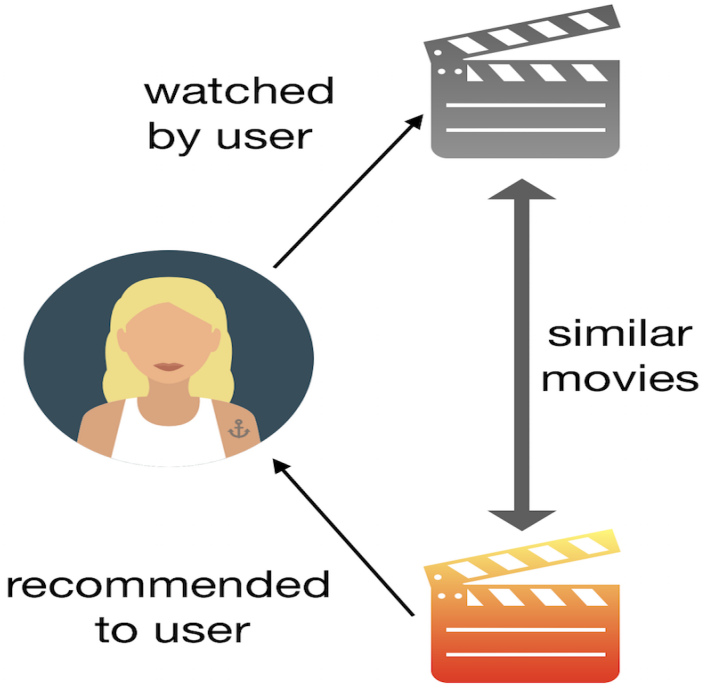
We already have v(**vote_count**) and R (**vote_average**) and C can be calculated.
The next step is to determine an appropriate value for m, the minimum votes required to be listed in the chart. We will use 90th percentile as our cutoff. In other words, for a movie to feature in the charts, it must have more votes than at least 90% of the movies in the list.
Now, we need to calculate our metric for each qualified movie. To do this, we will define a function, **weighted_rating()** and define a new feature **score**, of which we'll calculate the value by applying this function to our Data Frame of qualified movies.

Finally, let's sort the Data Frame based on the score feature and output the title, vote count, vote average and weighted rating or score of the top 10 movies.
We need to keep in mind is that these demographic recommenders provide a general chart of recommended movies to all the users.

	title	vote_count	vote_average	score
1881	The Shawshank Redemption	8205	8.5	8.059258
662	Fight Club	9413	8.3	7.939256
65	The Dark Knight	12002	8.2	7.920020
3232	Pulp Fiction	8428	8.3	7.904645
96	Inception	13752	8.1	7.863239
3337	The Godfather	5893	8.4	7.851236
95	Interstellar	10867	8.1	7.809479
809	Forrest Gump	7927	8.2	7.803188
329	The Lord of the Rings: The Return of the King	8064	8.1	7.727243
1990	The Empire Strikes Back	5879	8.2	7.697884

5.2 Content Based Filtering

Content-based methods make recommendations based on the description of the items. Nowadays, it is combined with other methods and use more information about items and users. However, several algorithms have been proposed to analyze the content of a document.



In this system for movies the content of the movie (overview, cast, crew, keyword) is used to find its similarity with other movies. Then the movies that are most likely to be similar are recommended

We need to do some text processing on the contents that we have.

TF-IDF stands for **“Term Frequency — Inverse Document Frequency”**. This is a technique to quantify a word in documents, we generally compute a weight to each word which signifies the importance of the word in the document and corpus. This method is a widely used technique in Information Retrieval and Text Processing.

By vectorizing the documents, we can further perform multiple tasks such as finding the relevant documents, ranking, clustering and so on.

TF-IDF = Term Frequency (TF) * Inverse Document Frequency (IDF)

Term Frequency measures the frequency of a word in a document. This highly depends on the length of the document and the generality of word, for example a very common word such as “was” can appear multiple times in a document. but if we take two documents one which have 100 words and other which have 10,000 words.
IDF is the inverse of the document frequency which measures the informativeness of term t. When we calculate IDF, it will be very low for the most occurring words such as stop words (because stop words such as “is” is present in almost all of the documents, and N/df will give a very low value to that word).

This will give you a matrix where each column represents a word in the overview vocabulary (all the words that appear in at least one document) and each row represents a movie, as before. This is done to reduce the importance of words that occur frequently in plot overviews and therefore, their significance in computing the final similarity score.

We use the built-in TF-IDFVectorizer class of scikit-learn to produces the TF-IDF matrix.

We will be using the cosine similarity to calculate a numeric quantity that denotes the similarity between two movies. We use the cosine similarity score since it is independent of magnitude and is relatively easy and fast to calculate. Mathematically, it is defined as follows:

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

We can see below that system has found movies with similar plot descriptions.

```
65                                The Dark Knight
299                                Batman Forever
428                                Batman Returns
1359                                       Batman
3854    Batman: The Dark Knight Returns, Part 2
119                                 Batman Begins
2507                                    Slow Burn
9           Batman v Superman: Dawn of Justice
1181                                         JFK
210                                Batman & Robin
Name: title, dtype: object
```

It goes without saying that the quality of our recommender would be increased with the usage of better metadata. That is exactly what we are going to do in this section. We are going to build a recommender based on the following metadata: the 3 top actors, the director, related genres and the movie plot keywords.

From the cast, crew and keywords features, we need to extract the three most important actors, the director and the keywords associated with that movie.

```
65                The Dark Knight
119                 Batman Begins
4638    Amidst the Devil's Wings
1196                The Prestige
3073           Romeo Is Bleeding
3326              Black November
1503                      Takers
1986                      Faster
303                     Catwoman
747                Gangster Squad
Name: title, dtype: object
```

We see that our recommender has been successful in capturing more information due to more metadata and has given better recommendations.

Results shown that there are certain disadvantages of employing content-based filtering.

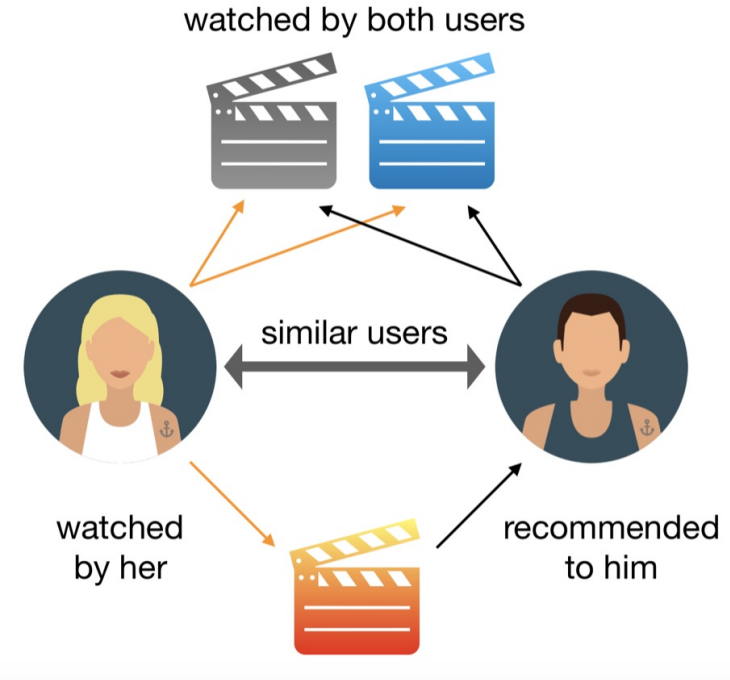
- content-based recommender systems tend to **over-specialize**. They will recommend items similar to those already consumed, with a tendency to create a "filter bubble", leaving less possibility for expanding a user's interests.
- The issue of limited content analysis: If the content doesn't contain enough information to discriminate the items precisely, the recommendation will be poor and thus hand-engineered features are required, or tags need to be assigned.

5.3 Collaborative Filtering

Collaborative Filtering, does not require any information about the items or the users themselves. It recommends items based on users' past behavior. I will elaborate more on Collaborative Filtering in the following paragraphs.

Collaborative filtering (CF) systems work by collecting user feedback in the form of ratings for items in a given domain and exploiting similarities in rating behavior among several users in determining how to recommend an item.

CF accumulates customer product ratings, identifies customers with common ratings, and offers recommendations based on inter-customer comparisons. It's based on the idea that people who agree in their evaluations of certain items in the past are likely to agree again in the future. For example, most people ask their trusted friends for restaurant or movie suggestions.



User based filtering- These systems recommend products to a user that similar users have liked. For measuring the similarity between two users we can either use Pearson correlation or cosine similarity. This filtering technique can be illustrated with an example. In the following matrixes, each row represents a user, while the columns correspond to different movies except the last one which

records the similarity between that user and the target user. Each cell represents the rating that the user gives to that movie.

Item Based Collaborative Filtering - Instead of measuring the similarity between users, the item-based CF recommends items based on their similarity with the items that the target user rated. Likewise, the similarity can be computed with Pearson Correlation or Cosine Similarity. The major difference is that, with item-based collaborative filtering, we fill in the blank vertically, as oppose to the horizontal manner that user-based CF does

In this part we use the **Surprise** library that used extremely powerful algorithms like **Singular Value Decomposition (SVD)** to minimize RMSE (Root Mean Square Error) and give great recommendations.

Single Value Decomposition

One way to handle the scalability and sparsity issue created by CF is to leverage a **latent factor model** to capture the similarity between users and items. Essentially, we want to turn 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**.

Latent it is a broad idea which describes a property or concept that a user or an item have. For instance, for music, latent factor can refer to the genre that the music belongs to. SVD decreases the dimension of the utility matrix by extracting its latent factors. Essentially, we map each user and each item into a latent space with dimension r. Therefore, it helps us better understand the relationship between users and items as they become directly comparable

Get a mean **Root Mean Square Error** of 0.8930 which is more than good enough for our case.
For movie with ID 6268, we get an estimated prediction of **2.636**.

One good thing about this recommender system is that it doesn't care what the m ovie.
It works purely on the basis of an assigned movie ID and tries to predict ratings based on how the other users have predicted the movie.

6. Conclusion

We create recommenders using demographic, content- based and collaborative filtering. The IMDB Weighted Rating System was used to calculate ratings on which the sorting was finally performed.
We did some content processing for the second type of filtering. We will be using the cosine similarity to calculate a numeric quantity that denotes the similarity between two movies.
We used the powerful Surprise Library to build a collaborative filter based on single value decomposition. The RMSE obtained was less than 1 and the engine gave estimated ratings for a given user and movie and we can predict the rating that a user will give to a never watched movie and base on that recommendations can be made.

This model was my first experience making recommendation systems and so I get help from the previous works that has been done in this topic.