**UNIVERSITY OF MACAU**

**FACULTY OF SCIENCE AND TECHNOLOGY**



**CISC7201**

**Introduction to Data Science Programming**

**Group Project Report**

GROUP V

CHAN UN IN MB955057

SHU ZHENG MB955180

ZHOU MENGYUAN MB955265

1. **Introduction**

A recommender system refers to a system that is capable of predicting the future preference of a set of items for a user, and recommends the top items. The Simple Recommender offers generalized recommendations to every user based on movie popularity and (sometimes) genre. The basic idea behind this recommender is that movies which are more popular and more critically acclaimed will have a higher probability of being liked by the average audience.

In this project, we depicted a story of movie voting by two datasets called “movies\_metadata.csv” and “credits.csv”. By collecting, cleaning, exploring, processing data and making visualizations, we gave a rank for thousands of movies according to “Weighted Rating (WR)” score and made visualization of some variables. This project has a little similarity to simple recommendation system.

We used some libraries to build our project, including *NumPy*, *pandas*, *scipy.stats*, *matplotlib.pyplot*, *seaborn* and *ast.literal\_eval*. Among them, *scipy.stats*, *seaborn* and *ast.literal\_eval* were not mentioned in class.

1. **Data Description**

The datasets were download from Kaggle and here is the link: <https://www.kaggle.com/rounakbanik/the-movies-dataset>

We used two datasets called *“movies\_metadata”*(32.8MB) and *“credits”*(181MB). The first dataset *“movies\_metadata”* consists of 24 columns and 45455 rows. The column names are *“adult”*, *“belongs\_to\_collection”*, *“budget”*, *“genres”*, *“homepage”*, *“id”*, *“imdb\_id”*, *“original\_language”*, *“original\_title”*, *“overview”*, *“popularity”*, *“poster\_path”*, *“production\_companies”*, *“production\_countries”*, *“release\_date”*, *“revenue”*, *“runtime”*, *“spoken\_languages”*, *“status”*, *“tagline”*, *“title”*, *“video”*, *“vote\_average”* and *“vote\_count”*. The second dataset *“credits”* consists of 3 columns and 45504 rows. The column names are *“cast”*, *“crew”* and *“id”*.

Since we didn’t need so much items in this project, we should transform raw data to suitable data by data cleaning and processing. Some items format is a dictionary which is not easy to process, so we converted it into more suitable items by programming.

1. **Data Cleaning and Processing**

We extracted *md[‘year’]* from *md[‘release\_date’]* by using *apply()* method and transformed *md[‘popularity’]*, *md[‘vote\_count’]* and *md[‘vote\_average’]* into float format. Then we made a simple comparison and found out that only 2 of the movies are both popular and most voted ones. Then we used the code *“genres = (md['genres'].fillna(‘[]’).apply(literal\_eval))”* to fill the blank data and turned rows from string to list. As genres item from raw data is dictionary which is difficult to utilize, we used the code *“genres = (genres.apply(lambda x: [i['name'] for i in x] if isinstance(x, list) else []))”* to get the item of movie genres.

We could use the average ratings of the movie as the score, nevertheless it would not be fair enough. As a movie with 8.9 average rating and only 3 votes cannot be considered to be better than the movie with 7.8 as average rating but 40 votes. Therefore, we would adapt IMDB's weighted rating (wr) which is given as:



where,

* v is the number of votes for the movie
* m is the minimum votes required to be listed in the chart
* R is the average rating of the movie
* C is the mean vote across the whole report

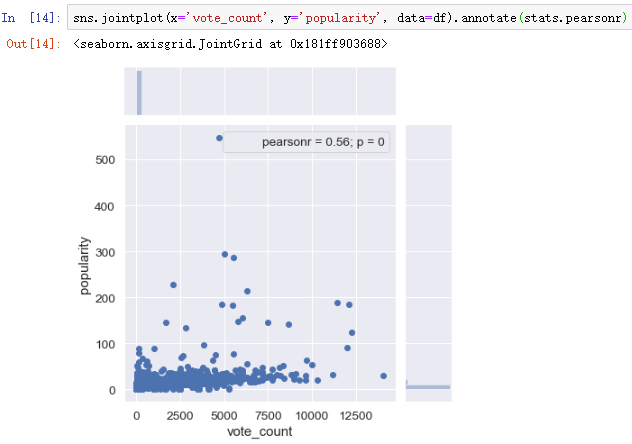
To calculate WR, we defined a function *weighted\_rating(x, m, C)*. By this function, we could easily calculate the WR value.

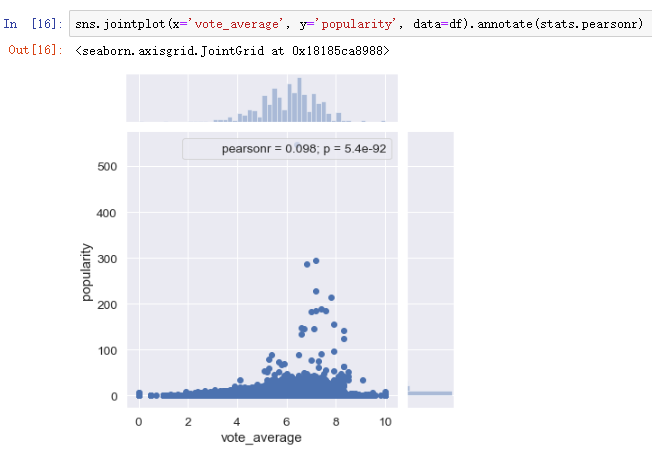
Furthermore, we defined a function named *build\_chart(genre, percentile)*. By this function, we can input two parameters: genre and percentile. As this step, we can pass in a genre argument to get the top movies of a particular genre. The meaning of percentile is that it must have more votes than at least percentile of the movies in the list for a movie to feature in the charts. For example, if we input *build\_chart('Crime', 0.90)*, we will attain a list of movies which genre is “Crime” ranked by “WR” score, and these movies must have more votes than at least 90% of this genre movies.

Finally, we combined the two tables *“movies\_metadata”* and *“credits”* into a more intuitive table. We can get the director ranked by “WR” score from a new created table.

1. **Data Visualization**

We used the *seaborn* library to make visualizations. The two graphs below demonstrate the relationship between *“vote\_count”*, *“vote\_average”* and *“popularity”*. Moreover, we used the *scipy.stats.pearsonr* to calculate Pearson Correlation Coefficient of two variables. From the graphs we can easily discover relevance of two variables. The *“vote\_count”* is not so correlated with *“popularity”* (r=0.56) and the average vote value (0-10) is not correlated with popularityat all. As a result, the more popular movie does not necessarily mean higher rating.





1. **Analysis and Conclusion**

In this project, we used our datasets to do a analysis about movie voting and made visualization to explore the correlation between *“vote\_count”*, *“vote\_average”* and *“popularity”*. We have found out that *“vote\_count”* is not so correlated with *“popularity”* (r=0.56) and the average vote value (0-10) is not correlated with popularity at all. The more popular movie does not necessarily mean higher rating.

By a series of operation such as cleaning, process, integrate, we can select our interested movies from datasets ranked by “WR” score. After combining the *“movies\_metadata”* with *“credits”* datasets, we generated a new DataFrame including *“director”* item, and then we can select the highest rated directors by “WR” score.

**References**

There are some excellent references that we studied and would like to share.

1. <https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75>

2. <https://www.kaggle.com/ibtesama/getting-started-with-a-movie-recommendation-system>

3. <https://www.kaggle.com/rounakbanik/movie-recommender-systems>