Maharishi International University

Recommender System and Embedding Machine Learning, CS, MIU[1]

Mukhammadjon Yorkinov, Marjan Kamyab, Aisuluu Baktybekova, Bunyodjon Hoshimaliev, Ershad Islam

August 1, 2022

[1]Instructor: Prof. Anthony Sander

Abstract

Since the online business market includes streaming services such as Youtube and NetFlix, online shopping and other social media recommendation system is an essential part of it. In this project, we build a complete recommender system by utilizing techniques such as Demographic, content-based similarity, SVD, and neural network collaborative filtering to recommend relatively data for the user. Each technique was tested with different hyperparameters. We have used two popular movie datasets and applied the different preprocessing techniques to the accurate recommendations. Finally, We have used flask API to deploy this recommendation system project with docker and Heroku server as live system.

Contents

1 Introduction **1**

2 Exploratory Data Analysis **2**

2.1 Important Attributes 2

2.2 Graphs and Evaluation 3

3 Recommender System **8**

3.1 Demographic 8

3.2 Content Based 8

3.3 Collaborative filtering 8

3.3.1 Singular Value Decomposition (SVD) 8

3.3.2 Neural Collaborative Filtering 9

4 Deployment **12**

4.1 Docker 12

4.2 Boosting 12

5 Conclusion **13**

Chapter 1

Introduction

With the rise of platforms such as Youtube, Amazon, Netflix and many other such web services, recommender systems have taken more and more place in our lives. 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). The reason is to generate a huge amount of income and customer satisfaction when they are efficient. Also, it is to be a way to stand out significantly from competitors. In this project, our aim was to build a complete movie recommendation system by utilizing demographics, content based and collaborative filtering techniques.

As a data we used 2 different data sets obtained from Kaggle Community.[1] [2] Movies in the first data set had movies same as MovieLens Latest Full Dataset and contains nearly 45,000 movies with 28 features of different data types. This data contained sub data such as movies_metadata, keywords and credits. The second data is " MovieLens" dataset with two file ratings with 100004 records and four columns(UserID, movieID, rating, and timestamps) and 62423 movie records which has three columns including (movieId, title, genres). All the data sets were combined by id of the movie.

[1]https://www.kaggle.com/code/rounakbanik/the-story-of-film/data

[2]https://www.kaggle.com/datasets/garymk/movielens-25m-dataset

Chapter 2

Exploratory Data Analysis

The data had few Not Applicable values except homepage and tagline columns. Homepage doesn't generate information as it is unique for each entry but it was used in recommendation part. Since UI is also available, it shows just 7782 movies, which is the total number of Homepage available. Words in the tagline might have an effect on people's decision to watch the movie. Therefore, tagline was used as a variable in whole process even if it had less than 50% available data. Moreover, original titles were dropped because there was already a feature 'title' that have all in English.

Nearly 38,000 out of 45,000 of data didn't give information about revenue and budget; however, it carries an valuable information. As a solution firstly, 0 valued budget entries were replaced with NA value because it is nearly impossible to produce movie for 0$. Secondly, to scale revenue and budget ratio of them was taken as 'return' and later both was removed to not have high correlation with return. A return value ¿ 1 would indicate profit whereas a return value ¡ 1 would indicate a loss. Just 20% of the return is not Null but it may carry huge amount of information.

Release date was split to day, month, year since each of them might carry information about movie.[1] These attributes didn't have high correlations with other continuous attributes and showed variation in terms of popularity. So, each of them are effective attributes.

There are close to 0% adult movies in this data; thus, it was extremely imbalanced and it was dropped.

Some other features were also removed after obtaining Correlation Matrix (discussed below).

As a list of cast members, at most 10 important actors/actresses were chosen. In addition, as crew member only directors were chosen for simplification.

2.1 Important Attributes

- Popularity Score - assigned by TMDB and calculated depending on following categories:

  Number of votes for the day
  Number of views for the day
  Number of users who marked it as a "favourite" for the day
  Number of users who added it to their "watchlist" for the day

[1] https://towardsdatascience.com/machine-learning-with-datetime-feature-engineering-predicting-healthcare-appointment-no-shows-5e4ca3a85f96

Release date
Number of total votes
Previous days score

- Release Date - The date a new movie is entered into the theater distribution system for public viewing.
- Status - The status of the movie (Released, To Be Released, Announced, etc.)
- Video - Indicates if there is a video present of the movie with TMDB

2.2 Graphs and Evaluation

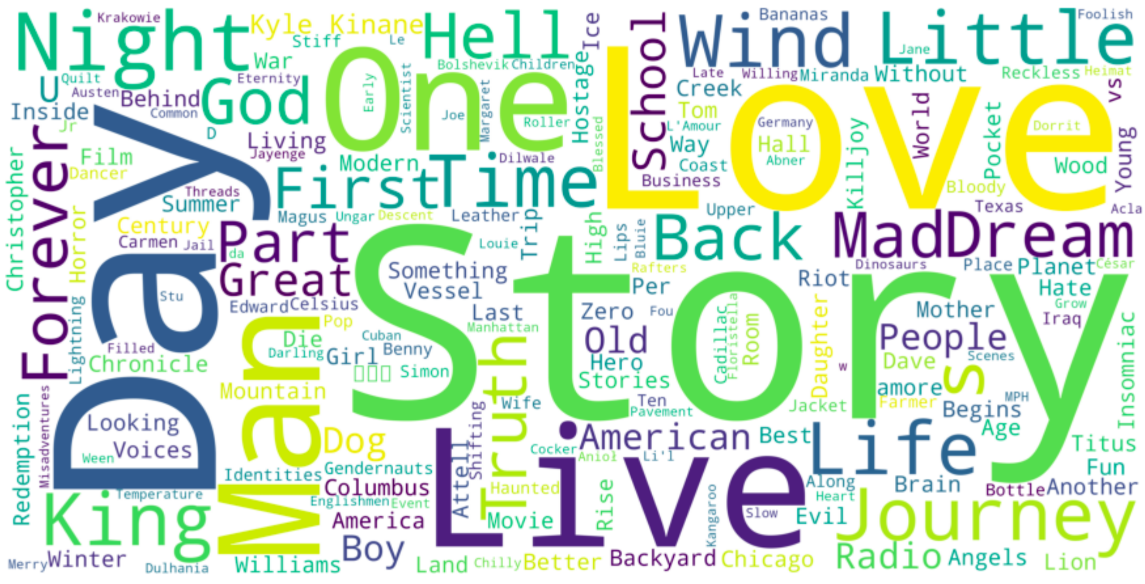


Figure 2.1: Word cloud of movie titles rated on average more than 9/10

This is the word cloud of movie titles which was rated on average more than 9/10. The word that appear most frequently is Story followed by Day, Love, Live, One and Man. Therefore, if the movie title contains word Story (which was not appeared for other lower ranges), most probable that it would be highly rated.

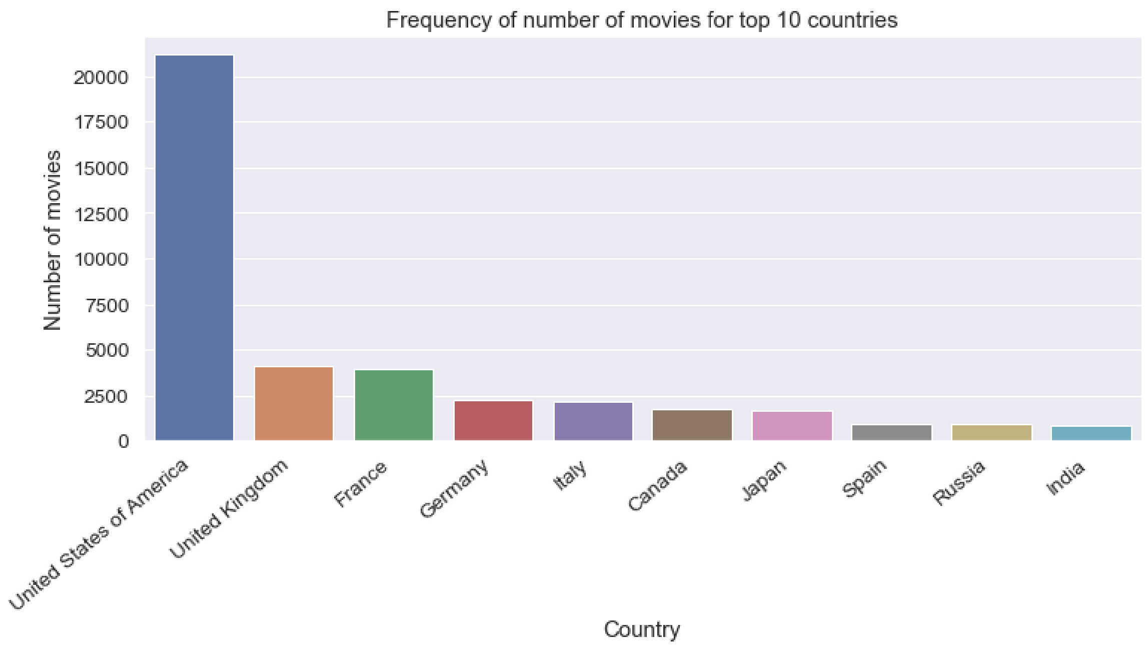


Figure 2.2: Top 10 countries in film production

USA produced the highest amount of movies (more than 20k) followed by UK and France (more than 4k) according to TMDB. Therefore, the recomender system will inclined towards USA produced movies.

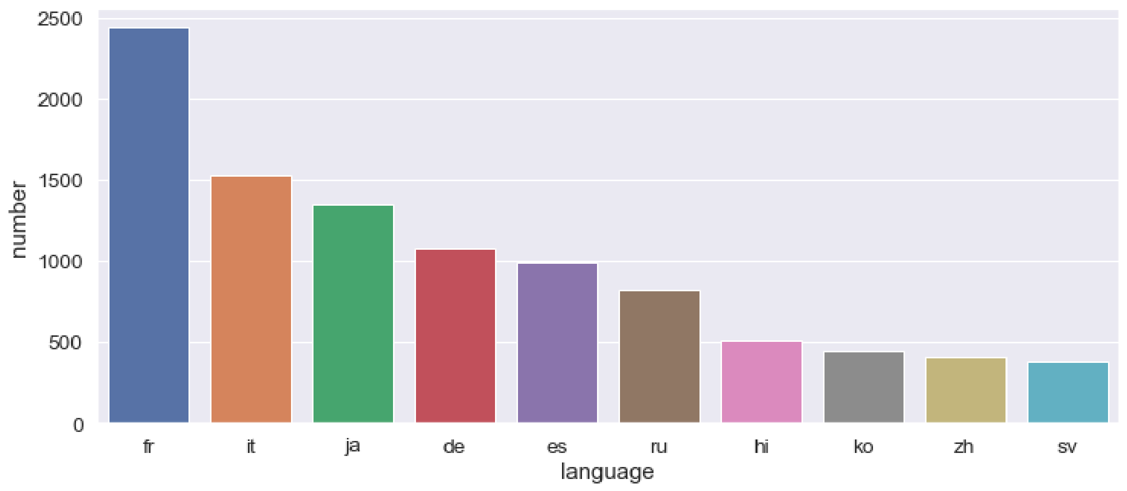


Figure 2.3: Most frequent 10 movie languages

Number of movies in English language is the highest, nearly 32,000 movies. French and Italian are the most commonly occurring languages after English. Japanese and Hindi form the majority of Asian Languages according to TMDB data.

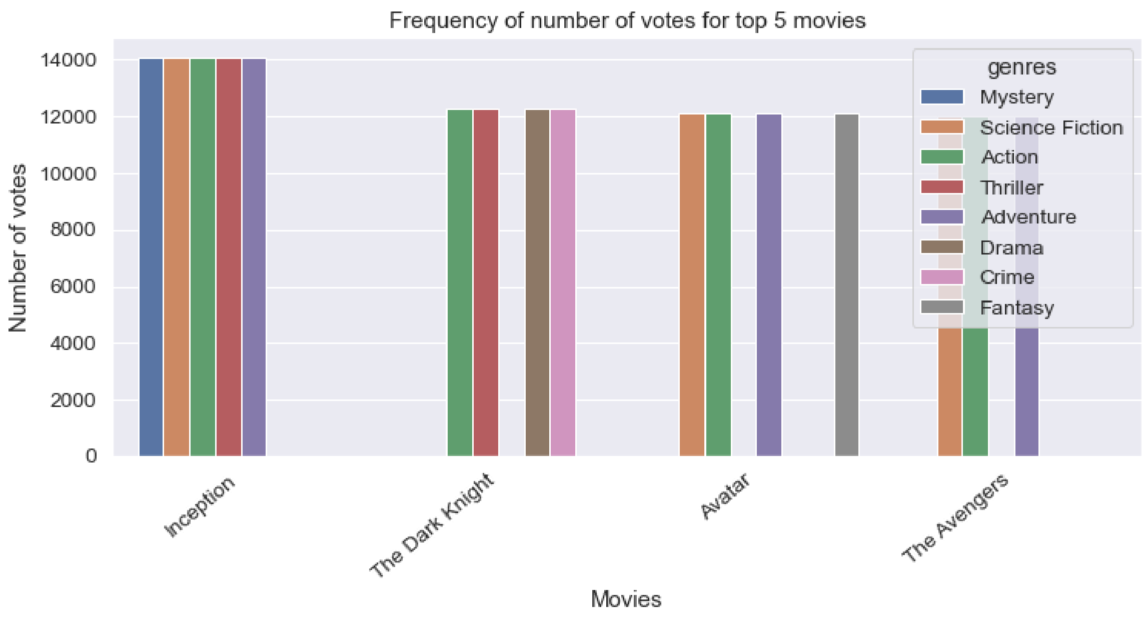


Figure 2.4: Top 5 highest voted movies with genres

The Figure 2.4 shows top5 highest voted movies. What made them special? All top voted movies have Action genre followed by Science Fiction and Adventure. The least preferred genres to vote are Drama, Crime, Mystery, Fantasy and other unseen genres. Also, all top 5 voted movies have mixture of at least 3 genres. The top, "Inception", is a mixture of 5 different genres. Therefore, if movie making company produce mixture of at least 3 genres such as Action, Science Fiction, Adventure, Thriller and some minor genres, the probability of movie to be voted will be high. In addition, with the help of correlation matrix it can be said that vote count and popularity has positive correlation. Hence, higher the number of votes of a movie, higher its popularity score.

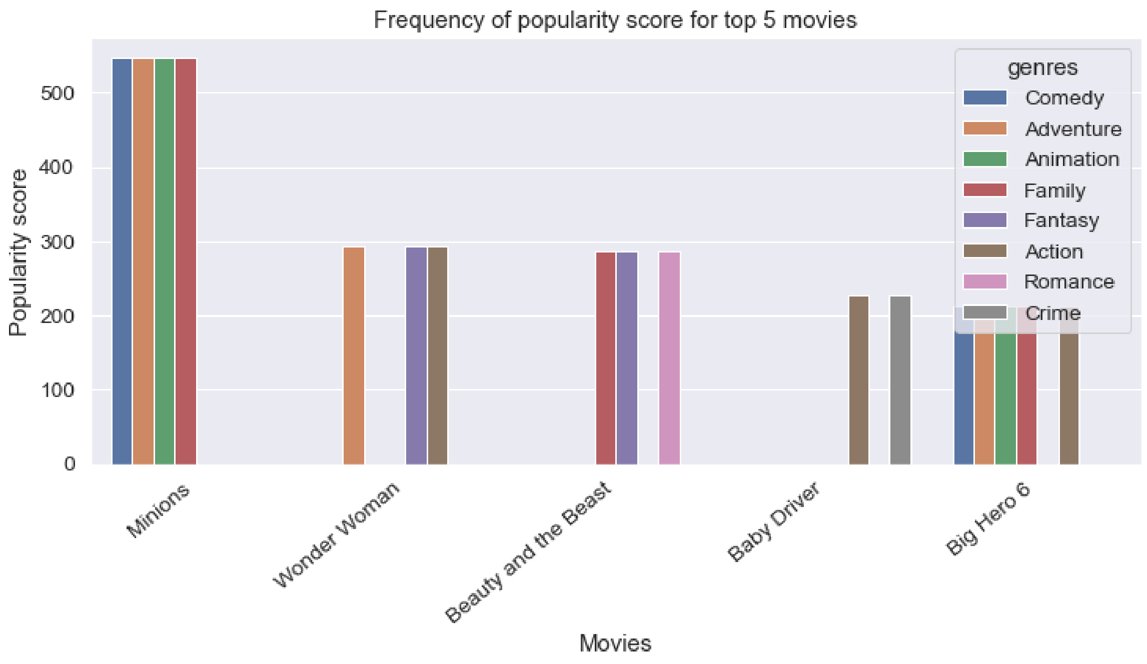


Figure 2.5: Top 5 popular movies with genres

According to Figure 2.5, the most popular movies are type of Adventure, Family and Action. The least popular ones are type of Romance, Crime and other unseen genres according to top 5 movies popularity score. Nonetheless, frequencies are not high, it is well distributed among top 5. Therefore, it is hard to tell about preferred genres. The reason might be that these 5 different movies are targeted to different range of ages. In addition, it can be observable that Animations such as Minions and Big Hero 6 contains high mixture of genres than others. Minions are nearly twice as popular as Wonder Woman or Beauty and the Beast with approximately 550 popularity score.

Also, according to Figure 2.4 and Figure 2.5, the least favorable types of genres are Romance and Crime.

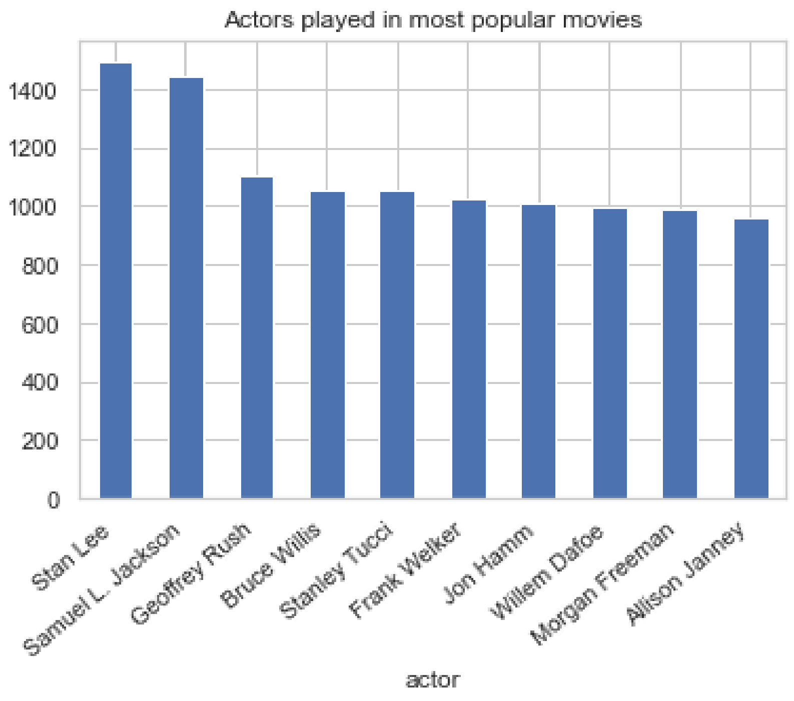


Figure 2.6: Top 10 most popular actors

Interestingly, Stan Lee and Samuel L. Jackson are the actors who appeared in most popular movies according to TMDB.

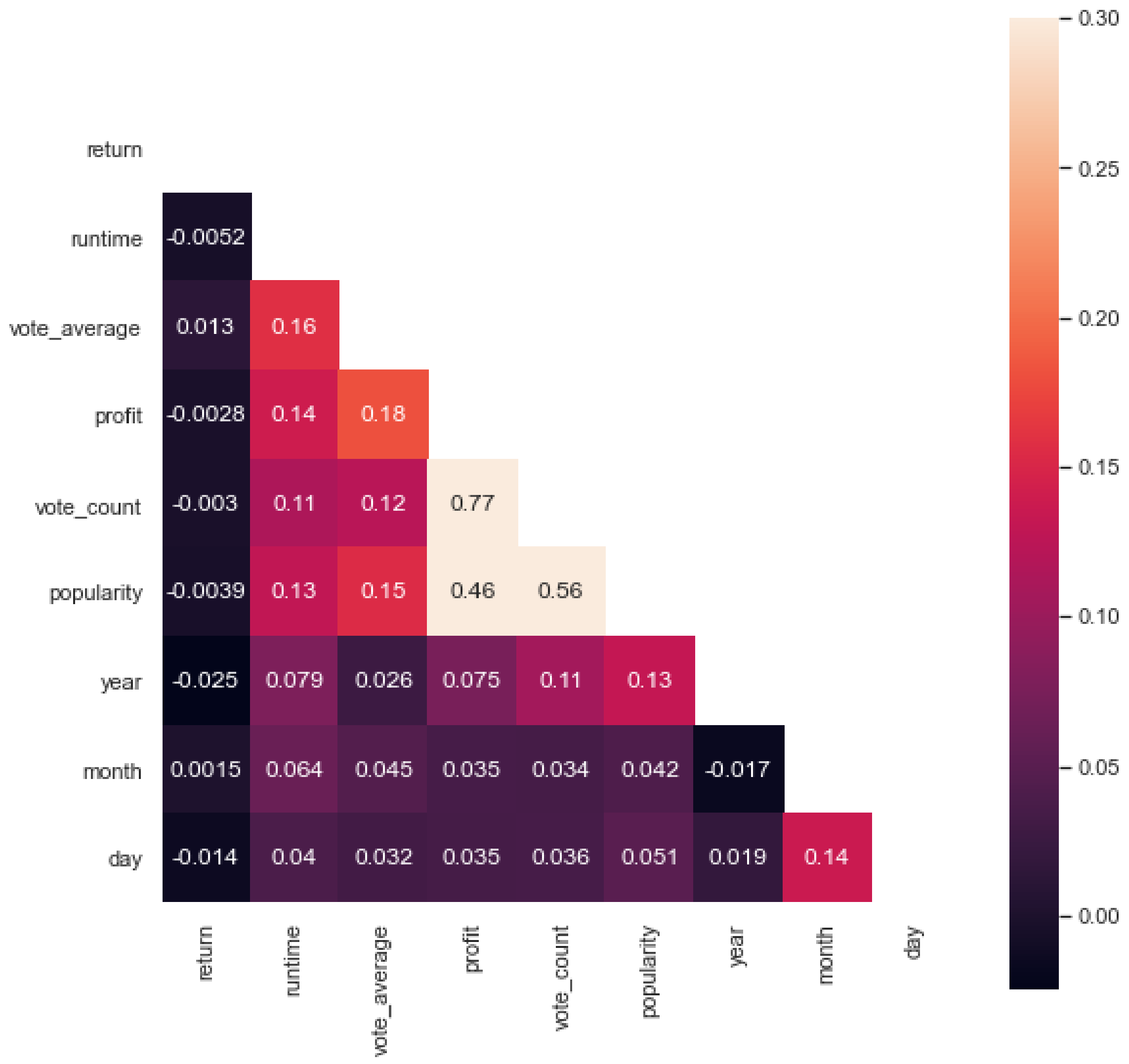


Figure 2.7: Correlation Matrix

From correlation matrix it can be seen that Profit has high linear correlation with 'vote count' and popularity whereas Return has very low correlation with all the features. Therefore, to eliminate multicollinearity Return should be chosen as derivative of revenue and budget and Profit will be removed. Also, 'vote count' has moderate correlation with popularity which will increase variance of the models. As a solution, 'vote count' will be removed because popularity carries a lot more information about the movie as it stated in the attribute description part.

Chapter 3

Recommender System

3.1 Demographic

For this part, 3 different recommendation criteria was used. The first, which is trending movies, was ordered by using special IMDB score that was calculated in preprocessing part. The formula to calculate this weighted rating is "(v / (v + M) * R) + (M / (M + v) * c)" where v = number of votes, R = vote average, M = required minimum vote to be listed. The second is top popular movies which is lineup depending on popularity, the special score that was already been calculated by TMDB. The third is top language based movies which used generic(raw) data.

3.2 Content Based

The movie preference can vary depending on many features, not only genres and titles. Hence, the whole clean and more reliable data that was obtained after feature engineering was processed in this part. As features [keywords, cast, director, genres, overview] were used. Firstly, by using CountVectorizer from Scikit-learn the text features were transformed into vector of token counts. This process also provided the capability to preprocess text data prior to generating the vector representation making it a highly flexible feature representation module for text. Then by using Cosine Similarity from Sklearn, we applied cosine similarity metrics to compute similarity between movies. After processing data it is serialized to another file with pickle.

3.3 Collaborative filtering

3.3.1 Singular Value Decomposition (SVD)

In this parts we briefly discuss process of building and testing an SVD for our movies recommender system on the MovieLens dataset using the Surprise library in Python with a focus on hyperparameter tuning.

Datasets

MovieLens provides available rating datasets from Kaggle, [1] it contains two files which are "user_ratings.csv" and "tmdb_5000_movies.csv" with 100004 ratings and 5,000 movies.

Library and Hyperparameters

we have utilized Surprise which is a scikit package for building and analysing recommender systems maintained by Nicolas Hug. For the prediction we have used cross validate with

[1]https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata.

the cv=5, 'RMSE' and 'MAE' as measurement. we explicitly divide the rating data into 75% for training and rest of 25%. We have so far found out with the RMSE values range from 0.8906 to 0.9028, while we get the range for MAE from 0.6862 to 0.6939 in the 5 splits of MAE. In the data preprocessing, we merged our movie dataset with the credits dataset by df = movies_df.merge(credits_df, on='id'), we get with the df for any movie with all the details for that movie. Finally, the model serialized as PKL file in order to use in the future and deployment.

Result analysis

We have checked ratings by ratings to denote all the movies a particular user might have rated for a specific movie when we use the svd. predict(1, 302, 3), it means that the user_id = 1 and movie_id = 302 and the rating of that user for that movie = 3. We found out that the predict function outputs = 2.672, which is closer to the real value of 3. We have so far found out that the RMSE values range from 0.8906 to 0.9028, while we get the range for MAE from 0.6862 to 0.6939 in the 5 splits of MAE.

We have found in the prediction for each user for whom 20 movies are being predicted. And we have predicted 100 users from the dataset. The movie ratings for users are being predicted here for which users have not yet rated those movies. Here we utilized 4803 movies to be predicted with users who have not rated these movies previously, or we may say these are the unwatched movies yet, in such of measurements, some movies may be rated already by some users but not rated by other users, so the movies coming into the space of predictable scenario redundantly.

3.3.2 Neural Collaborative Filtering

Neural network architecture

the Neural network comprises data preprocessor, embedding layer, Neural network Layer. We also applied dropout to reduce the dimensionality and overcome with the overfitting problems.

Datasets

In this part of the project, we have used the " MovieLens" dataset for training our model, which is a well-known dataset for recommendation systems. MovieLens dataset is available on the Kaggle [2] website with two file ratings with 100004 records and four columns(UserID, movieID, rating, and timestamps) and 62423 movie records which has three columns including (movieId, title, genres).

Training procedure and hyper-parameters setting

The model implemented based on the Keras deep learning API written in python language. The models are executed on the Windows operating system with 3.00 GHz processor and 8 GB RAM. We have two embedding layers, including "user embedding" and "movie embedding". Then the two embedding layers Concatenated to use as input for neural network layers. In training, 80% of datasets observations were used for training, 10% for validation, and 10% for testing. We have applied four Dense Layers with 64, 32, 16, 8, and 1 sizes, respectively. Dropout with the value of 0.5 was used at the output of the first and second dense layers to avoid overfitting. From 1-4, dense layers used ReLu as activation function and the L2 as regularization. The data fit with 256 batch sizes and 50. After the dense layer, a Sigmoid function is used for the binary classifier. Finally, binary cross-entropy is used for the training model, and Adam optimizes the model's learning rate by 0.0001.

[2] https://www.kaggle.com/datasets/garymk/movielens-25m-dataset

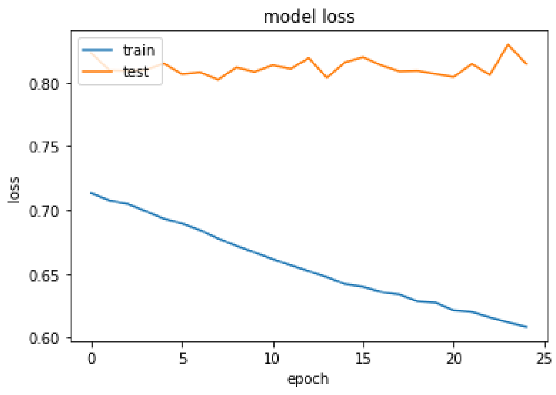
Table 3.1: Model hyper-parameters setting.

Hyperparameter	**Value**
Number of epochs	[1–50]
Batch size	256
Embedding dimension	50
Number of Embedding	2
Dropout rate	0.5
Number of NN layers	5
Regularization	L2
Learning rate	0.0001
Optimizer	Adam
Activation function type	ReLu, Sigmoid

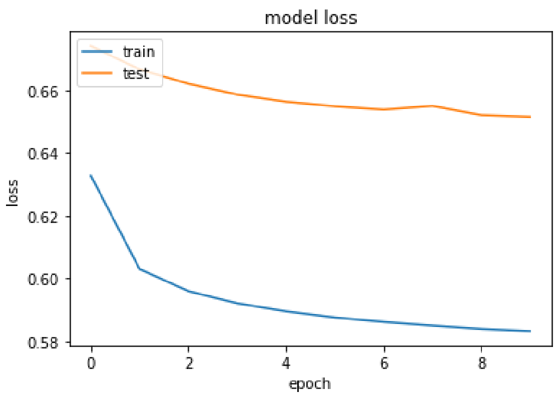
Table 3 describes the hyper-parameters value in our model. Finally, the model save as the HDF5 files as further use and have the ability to recommend the movies based on the rating and personality.

Result Analysis

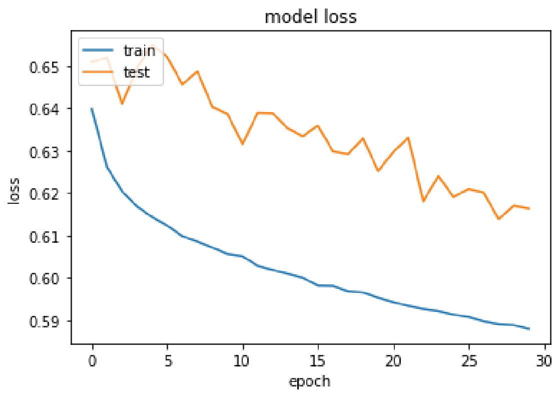
The experiment result validates that the model performed well and achieved significant performance. Figure 1(a-b) shows the improvement process of this model. The model validation loss values become stable after 8 epochs until the end of training and its the excellent achievement.



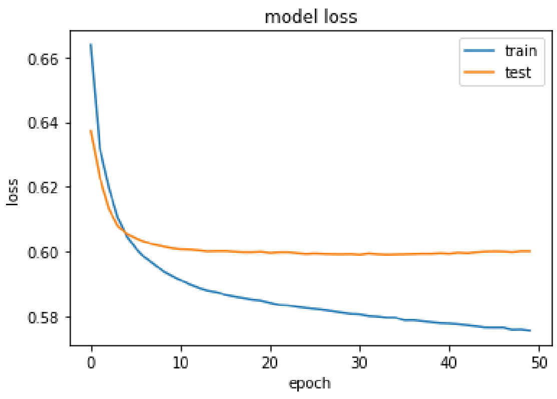
(a)



(b)



(c)



(d)

Figure 3.1: Neural Collaborative Filtering loss Curve based on the 80% train and 20% test dataset and the improvement process, the figure 3.1(d) is our best model

Chapter 4

Deployment

After implementing Demographic, Content-Based, SVD, and neural collaborative filtering and export, all models as serializable, Demographic, Content-Based, SVD saved as PKL file and neural collaborative filtering stored in HDF5 format, which is Organized, flexible, and interoperable. We have used flask API to deploy this recommendation system project with docker and Heroku server.

4.1 Docker

We have used docker containers for deployment of the project we have configured docker configurations.

4.2 Boosting

We have used serialization for speeding up and when we run our server we are processing data and according to our needs we are saving them into another files using pickle once we need them in app we are opening corresponding file and fetching data from them.

Chapter 5

Conclusion

By the end of this project, we understand those recommender techniques are widely used for online business and are critical parts of online business markets. In this project, we have successfully implemented the most used techniques such as Demographic, Content-Based, SVD, and neural collaborative filtering. pre-processing techniques were applied to the dataset to make the data clean for the data transformation. The Neural network comprises a data preprocessor, embedding layer, and Neural network Layer. We also applied dropout to reduce the dimensionality and overcome the overfitting problems. Experiments were conducted on two different versions of movie MovieLens datasets. Neural networks achieved a val loss of 0.5970 and loss of 0.5768, and other methods recommend accurately. After the Demographic, Content-Based, SVD, and neural collaborative filtering and export, all models as serializable, Demographic, Content-Based, SVD saved as PKL file and neural collaborative filtering stored in HDF5 format. finally, We have used flask API to deploy this recommendation system project with docker and Heroku server.