Social and Information Network Analysis 42913

Assignment 3 - Report

Movie Recommendation System

Team Members:

Student Name	Student Id
Vraj Mehta	13488642
Nivetha Anand	13663024
Stefania Kikuchi	99186506
Yogesh Babu Krishnakumar	13505538

Table of Contents

Abstract

Information technology and internet of things have been rapidly increasing since it was first developed in the 1950s. Since then, information technology has been making a crucial impact on our daily lives, such as recommendation systems in the digital world. However, with the exponential development of information and internet users, excess of data has led to distressed issues affecting users across the digital world. Recommender systems have become the best solution for our digital users. This refers to software tools, techniques, and algorithms providing impactful suggestions for services and products to be implemented to a digital user. Digital platforms such as Spotify, Netflix, YouTube, Amazon Prime, Stan, and others, implement such recommendation systems using ratings, similarity, and user interests to generate faultless recommendation outputs as well as generating high income for these industries. This paper discusses and analyses a range of different techniques and algorithms used in ***movie recommendation systems*** and ***implements*** a ***customized model*** based on ***Matrix-factorization algorithm*** with ***training validation loss*** to be **1.8237**, ***Root Mean Square Deviation (RMSD)*** to be **1.35,** along with the evaluation metrics to be ***recall value*** to be **0.736.**

Key Words: Movie Recommendation System, Collaborative filtering, Content-Based filtering, Matrix factorization.

1. Introduction

Web technology has been gaining ground rapidly since the internet of things first came out. It is well acknowledged that the current generation has welcomed, embraced, and become dependable on these technologies such as Smartwatch, Smartphones, TV, web-based applications including Netflix, Amazon, Spotify, to name a few. As a result, this generation is changing the current lifestyle as we are constantly surrounded by screens, and everything has become almost digital (Yiu-Kai 2017).

However, too much information has become a headache for consumers. What is more, since COVID-19 hit the world in 2020, making countries go lockdown, the usage of web-based applications increased tremendously. For instance, Netflix experienced a great 16 million new subscribers in the first three months of 2020 (BBC News 2020). As a result, to help internet users cope with this tremendous amount of information available on the web, Netflix and other web-based companies deployed recommendation systems that can accurately predict movie suggestions based on users' interests. Movie recommendation systems use users' reviews and interests expressed on the internet to predict similar movies users would enjoy watching. Moreover, such recommendation systems have become a crucial source of income for big industries.

The engine behind the movie recommendation system is powered by three types of filtering, which can be implemented to all kinds of products in the digital world the user enjoys watching:

- Collaborative based filter
- Content-based filter
- Hybrid based filter

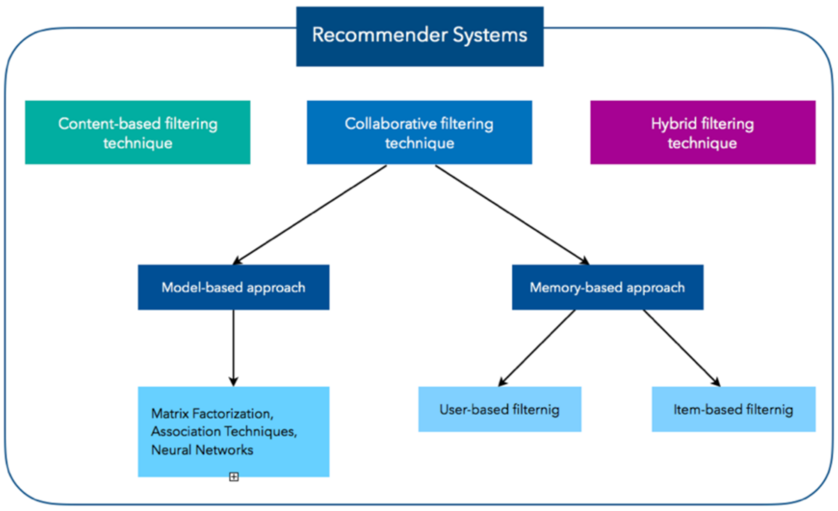

Recommender Systems
Content-based filtering technique
Collaborative filtering technique
Hybrid filtering technique
Model-based approach
Memory-based approach
Matrix Factorization, Association Techniques, Neural Networks
User-based filternig
Item-based filternig

Fig 1: Overview of the filtering Techniques

Content-Based Filter:

In this technique, recommendations are provided to the user's based on a single user's data and behaviour. ***"User's profile" and "Item description"*** are the two main terminologies that play an essential role in the content-based technique (Patel, 2017). The content-based filter **analyses the user's description of the preferred item**. Based on the likelihood of the description, it provides recommendation on the related items that have high similarity.

Collaborative-Based Filter:

Collaborative Filter basically ***builds datasets based on user preferences***. Datasets are created based on the categorical set of user's and other users who has similar interests to the categorical users. Recommendation to the effective user is provided by comparing other similar users interests and preferences who have rated the similar items same as that of the effective user. Most importantly, this system ***categorizes a set of audience*** based on ***analysing more than one common item*** (Patel, 2017). Further, this filtering technique is classified into two types such as memory-based and model-based (Patel, 2017).

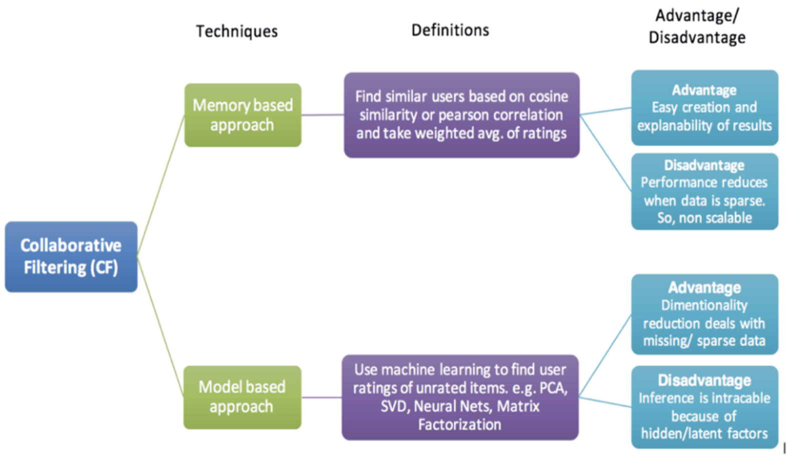

Techniques
Definitions
Advantage/ Disadvantage
Collaborative Filtering (CF)
Memory based approach
Find similar users based on cosine similarity or pearson correlation and take weighted avg. of ratings
Advantage Easy creation and explanability of results
Disadvantage Performance reduces when data is sparse. So, non scalable
Model based approach
Use machine learning to find user ratings of unrated items. e.g. PCA, SVD, Neural Nets, Matrix Factorization
Advantage Dimentionality reduction deals with missing/ sparse data
Disadvantage Inference is intracable because of hidden/latent factors

Fig 2: Broad Overview on the types of Collaborative filtering

Hybrid-Based filter:

The main idea of a hybrid-based filter is to ***combine both content-based filter and collaborative based filter*** to give more effective and better recommendations. Because on implementing both the algorithms, if there is any drawback of one algorithm, the other algorithm would overcome the drawbacks (Patel, 2017).

Recommender System Methods	**Content-based filtering**	**Collaborative filtering techniques**	**Hybrid filtering**
Number of users	Based on a single user	Based on many users having similar interests	Combination of content-based and collaborative filtering
Advantages	User independence, Transparency	It improved recommendation and performance.	Overcome cold start and sparsity problem.
Disadvantages	Limited content analysis, new user	Data sparsity, Scalability, Synonymy	Increased complexity, expensive in implementation.

Fig 3: Comparisons among different recommender techniques

2. Literature Review:

As we have chosen a model-based collaborative filtering technique for designing our movie recommendation system. This section will have an in-depth knowledge of different model-based algorithms based on which the recommendation systems are built.

K-Nearest Neighbour Algorithm (KNN)

The KNN algorithm is most popularly used in classification algorithms. The ***concept of KNN*** is to obtain from the ***principle of nearest neighbour classification***. The principle of nearest neighbour classification is to determine the objects based on their category. KNN determines the category of the object based on categorical attributes of more than one object (Li, 2020). A hybrid based collaborative filter method was developed by combining K-nearest neighbour, which calculates the correlation between users and items and gradient boosting with ensemble learning was used to predict the ratings of the items by users (Lu, 2018).

Matrix-factorisation

Matrix-factorisation is an extension of a collaborative filtering engine that consists of breaking down layers with the purpose of extracting features from both users and items matrix, where rows represent a user and columns represent an item, alleviating data sparsity issue (Barathy R. & Chitra P. 2020, pp.636). User and item matrix interaction can be acquired by implementing the dot product of their respective dense vectors in that same space. In addition, matrix factorisation is based on user-item scores. CF (Collaborative Filtering) methods have improved accuracy in RS (Recommender Systems); however, the downfall is data sparsity problems.

Singular Value Decomposition (SVD)

SVD is also an extension of the CF (Collaborative Filtering) algorithm. It is a better and improved method algorithm of matrix factorization which is utilised to decrease key features in a dataset by "decreasing space dimensions from N to K, where K<N is used" (Barathy R. & Chitra P. 2020, pp.637). The key difference between matrix factorisation and SVD is that matrix factorisation produces two lower dimensionality matrices from the original dot product matrix, whereas SVD produces another three other matrices.

Neural Networks (NN):

Neural Networks is another algorithm used by deep learning, which uses a collaborative filtering method to provide recommendations. Several neural network approaches were used to implement recommender system by reducing general problems such as ***Sparsity, Cold start, Scalability, Privacy and Predictability.*** The main aim of using a multi-layered neural network is to reduce the "***Sparsity"*** problem (Chakrabarti, 2019). ***Probabilistic Neural Networks (PNN)*** was used to calculate trust based on users' ratings and alleviate cold start and sparsity problems (Devi, 2010). ***Collaborative Neural Network (CoNN)*** was implemented for improving personalized recommender system which uses ***Personalised Feature Extractor (PFE)*** to solve this problem (Devi, 2010).

3. Recommender System

Matrix Factorisation

As mentioned earlier, matrix factorisation is one of the algorithms applied in the collaborative filter engine to generate hidden features and layers by multiplying two different types of factors, users and items, and identify their relationship. Matrix factorisation decomposes the user-item matrix interaction into the product of two lower matrices.

The dot product of these two factors can produce a rating matrix, where the user matrix contains users and features, and the item matrix contains items and features. In addition, this algorithm implements a function that reduces dimensions by removing random and empty variables leading to obtaining better prediction and improved outputs.

Moreover, this technique of breaking down dimension from raw data can be visualised for improved and better predictions. However, a downfall in this algorithm would be the collection of data which might contain multiple boisterous factors that could affect the accuracy prediction and outputs. Thus, this algorithm's function can quickly solve this issue, aiming to adapt its dimensions for a clearer and better understanding of the collected data. The following figure 4 formula represents the decomposition equation:

$$\hat{Y} = UV^T$$

Fig 4. Matrix. Decomposition formula

Where U represents m*k, and V represents n*k. Let m = users, n= items, and k=features.

Figure 5 below portrays the rating prediction formula for matrix factorisation user-item. Ui represents a user, and Vj represents an item. Figure 6 illustrates the concept of matrix factorisation in a collaborative filtering engine.

$$\hat{y}_{ij} = u_i \cdot v_j$$

Fig 5. dot product rating prediction for user-item pair.

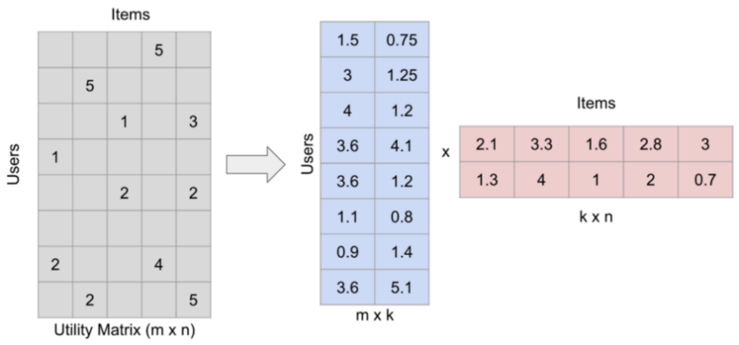


Fig 6. Matrix factorisation representation

4. Network Analysis

The selected dataset consists of 100k records, which have the user's movie rating and its details. Based on this dataset, the movie recommendation system was implemented. The dataset consists of details like user id, movie id, and movie ratings by the user. These three columns are used to generate the graph to analyse the movie ratings and understand the likeliness of the movie ratings between the users.

Since the 100k records were difficult to process to implement the graph, the number of records is reduced to 2000 records with seven unique user IDs and 999 unique movie IDs. This 2000 dataset records were enough to populate the general trend of the users via social network analysis of users' movie likeliness.

The network graph in figure 7 shows that the modularity of the network graph is calculated with all seven users. In figure 8, their network graph shows the likeliness filtered to only one of the users' movie ratings.

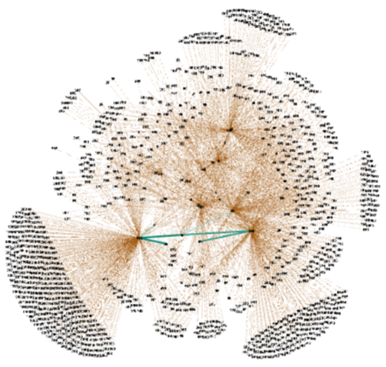


Fig 7. Network Graph visualisation with all 7-modularity communities

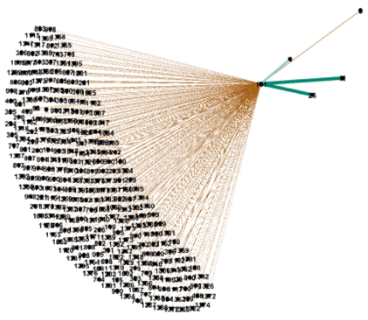


Fig 8. Network Graph visualisation with all 1-modularity community for user_id 108

The network graph in Figure 9 is produced by using the modified dataset. This network graph helps to deduce the patterns of user's likeness in watching the movie and their ratings. The network graph is distinguished by using the modularity values. According to modularity calculation, there are seven communities which describe the seven unique users' movies pattern.

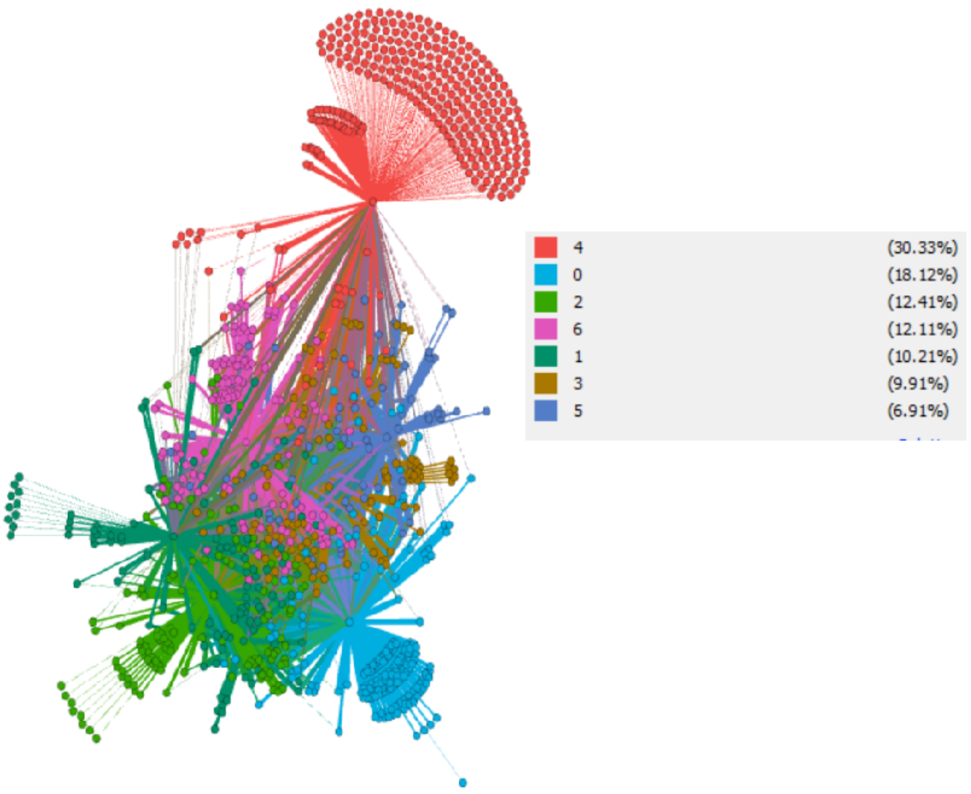


Fig 9. Network graph distinguished based on modularity

Network Statistics Summary:

- Number of Nodes: **2000**
- Number of Edges: **8004**
- Modularity: **0.376**
- Number of communities: **7**
- Average Degree: **3.974**
- Network Diameter: **4**
- Average Path Length: **2.699**

Analysis - Movie Enthusiasts and General movie reviewers:

The network in figure 10 illustrated that the graph consists of enthusiastic movie users and non-frequent movie-watching users. When the network is enlarged, it shows the common likeliness of the movie between two users (Figure 11). That cluster shows that both users like the same movies and may recommend the same movie due to the collaborative filtering algorithm. This shows that by using collaborative filtering, the algorithm movie recommends new users by matching the profiles of the previous users.

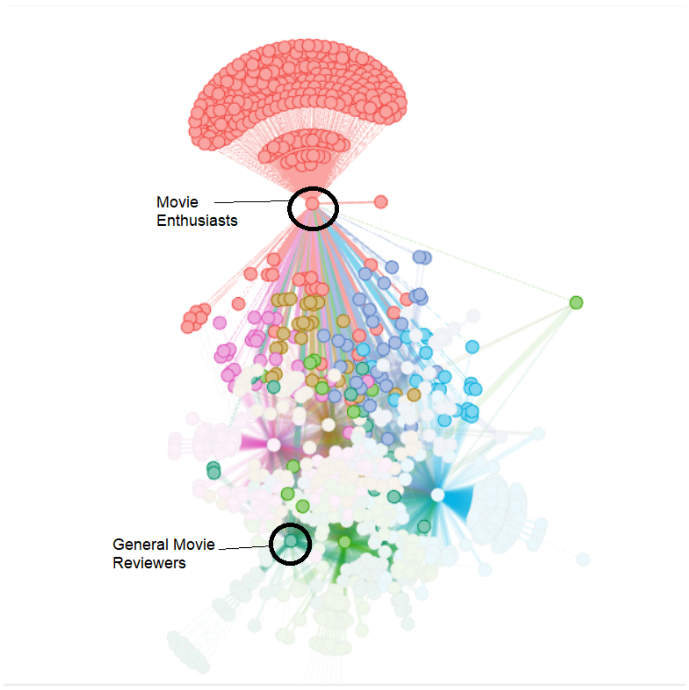


Fig 10. Network graph with Movie Enthusiasts and General movie reviewers

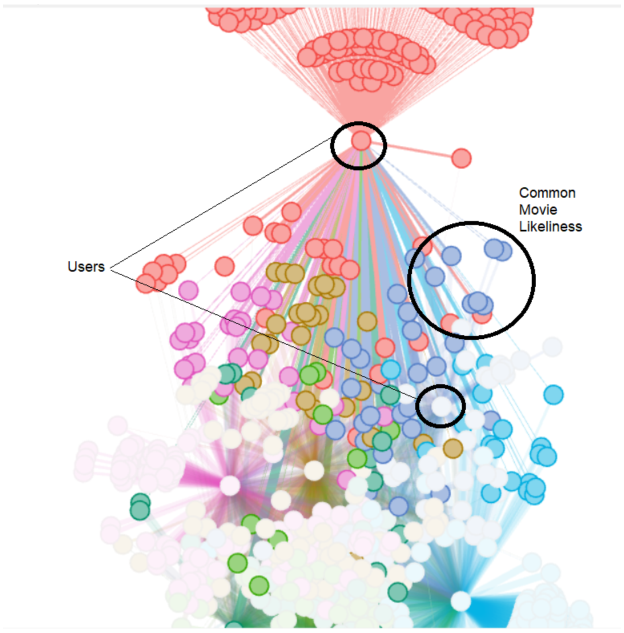


Fig 11. Network graph with user's likeliness

The network also illustrated the difference between hit movies and typical movies (Figure 12). Most of the hit movies were watched by multiple users, and the system will consider these movies as high recommend movies. So, these big hit movies will be recommended to the users even whose profile has not matched that movie type. According to the system, that recommended movie has more likeliness to the new user since most users have recommended that movie.

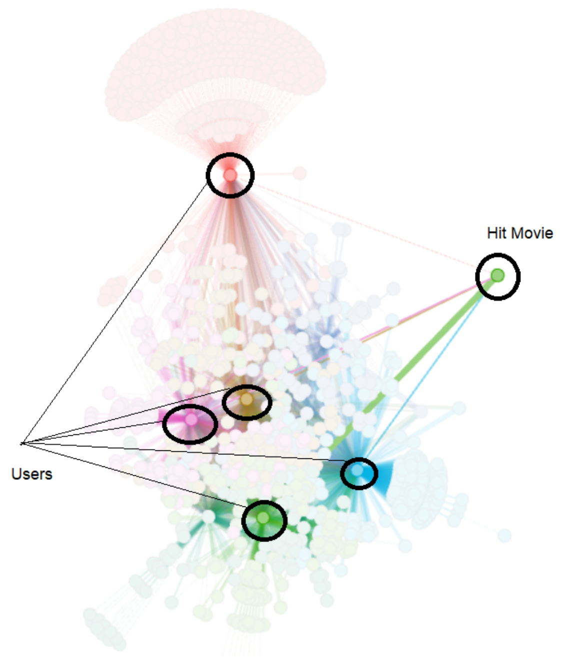


Fig 12. Network graph with a hit movie and general movies

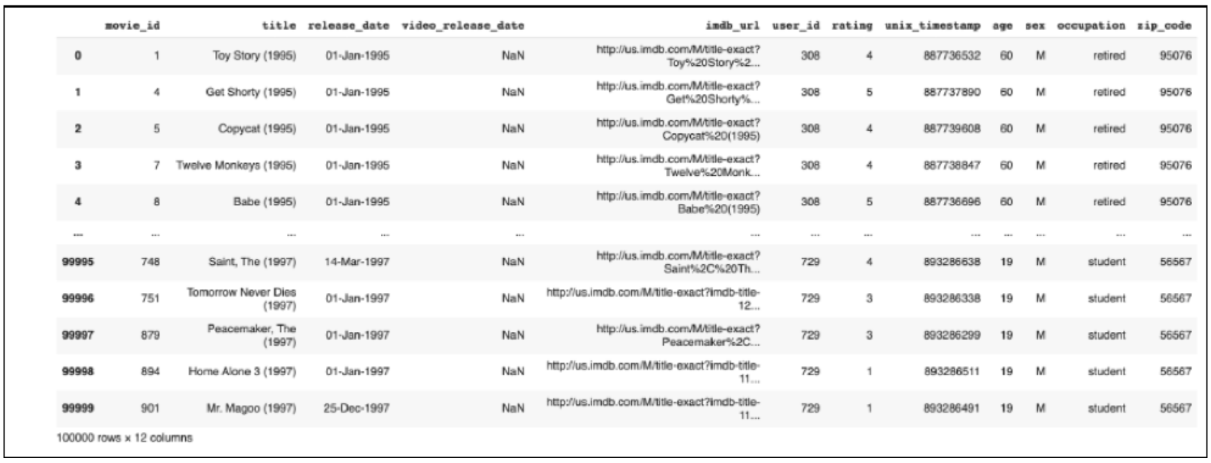
5. Python Implementation

Custom Matrix Factorisation algorithm is being used to predict the rating and provide recommendations.

Dataset Exploration

Reading and combining three given files, namely: u_user, u_data, u_item, followed by combining and converting it into a data frame, will result.

There is a total of **100k** rows with **12** distinct columns.



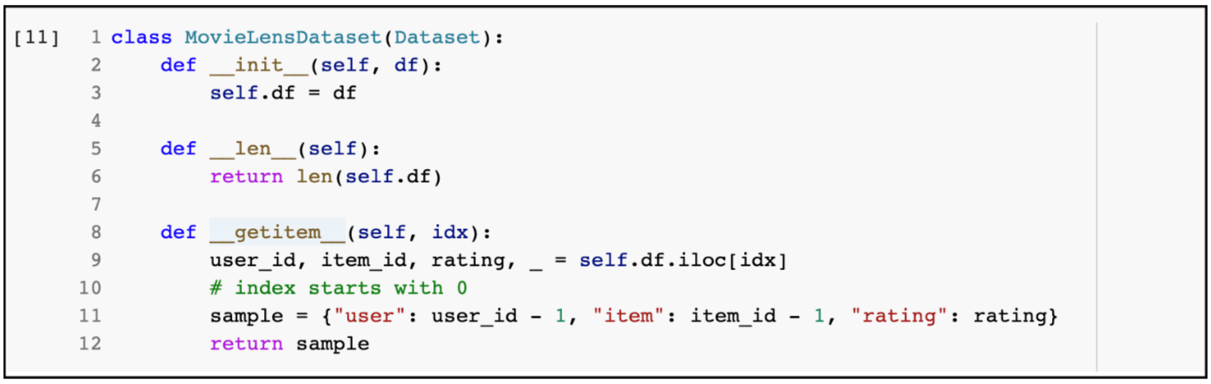
	movie_id	title	release_date	video_release_date	imdb_url	user_id	rating	unix_timestamp	age	sex	occupation	zip_code
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	308	4	887736532	60	M	retired	95076
1	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	308	5	887737890	60	M	retired	95076
2	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	308	4	887739608	60	M	retired	95076
3	7	Twelve Monkeys (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Twelve%20Monk...	308	4	887738847	60	M	retired	95076
4	8	Babe (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Babe%20(1995)	308	5	887736696	60	M	retired	95076
...	...	...	...	...	...	...	...	...	...	...	...	...
99995	748	Saint, The (1997)	14-Mar-1997	NaN	http://us.imdb.com/M/title-exact?Saint%2C%20Th...	729	4	893286638	19	M	student	56567
99996	751	Tomorrow Never Dies (1997)	01-Jan-1997	NaN	http://us.imdb.com/M/title-exact?imdb-title-12...	729	3	893286338	19	M	student	56567
99997	879	Peacemaker, The (1997)	01-Jan-1997	NaN	http://us.imdb.com/M/title-exact?Peacemaker%2C...	729	3	893286299	19	M	student	56567
99998	894	Home Alone 3 (1997)	01-Jan-1997	NaN	http://us.imdb.com/M/title-exact?imdb-title-11...	729	1	893286511	19	M	student	56567
99999	901	Mr. Magoo (1997)	25-Dec-1997	NaN	http://us.imdb.com/M/title-exact?imdb-title-11...	729	1	893286491	19	M	student	56567

100000 rows × 12 columns

Total number of distinct users: **943**
Total number of distinct movies: **1682**

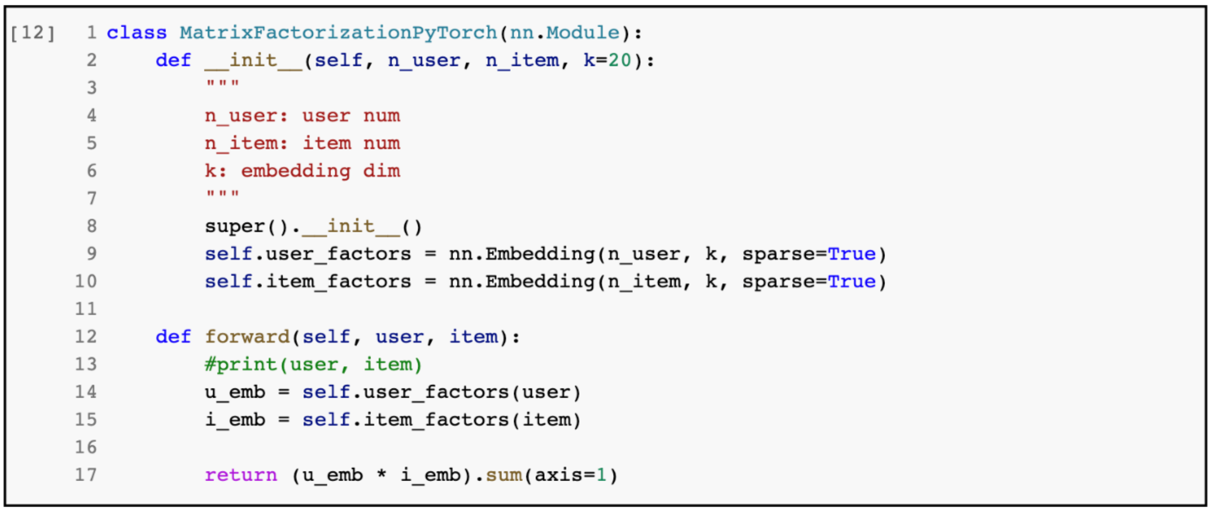
Defining Custom Dataset and Model

We defined a custom class to load the dataset. It consists of three functions, including the constructor. Function '__len__' will return the total number of rows in the dataset (data frame). Function '__getitem__' will return a dictionary of user_id, item_id and rating. Here item_id refers to movie_id.



```
[11]
class MovieLensDataset(Dataset):
    def __init__(self, df):
        self.df = df

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        user_id, item_id, rating, _ = self.df.iloc[idx]
        # index starts with 0
        sample = {"user": user_id - 1, "item": item_id - 1, "rating": rating}
        return sample
```

Given a user number and item number, Matrix Factorization PyT orch will return a 2-D Tensor. It has embedding layers that will calculate individual factors, then multiplied to result in a final tensor.

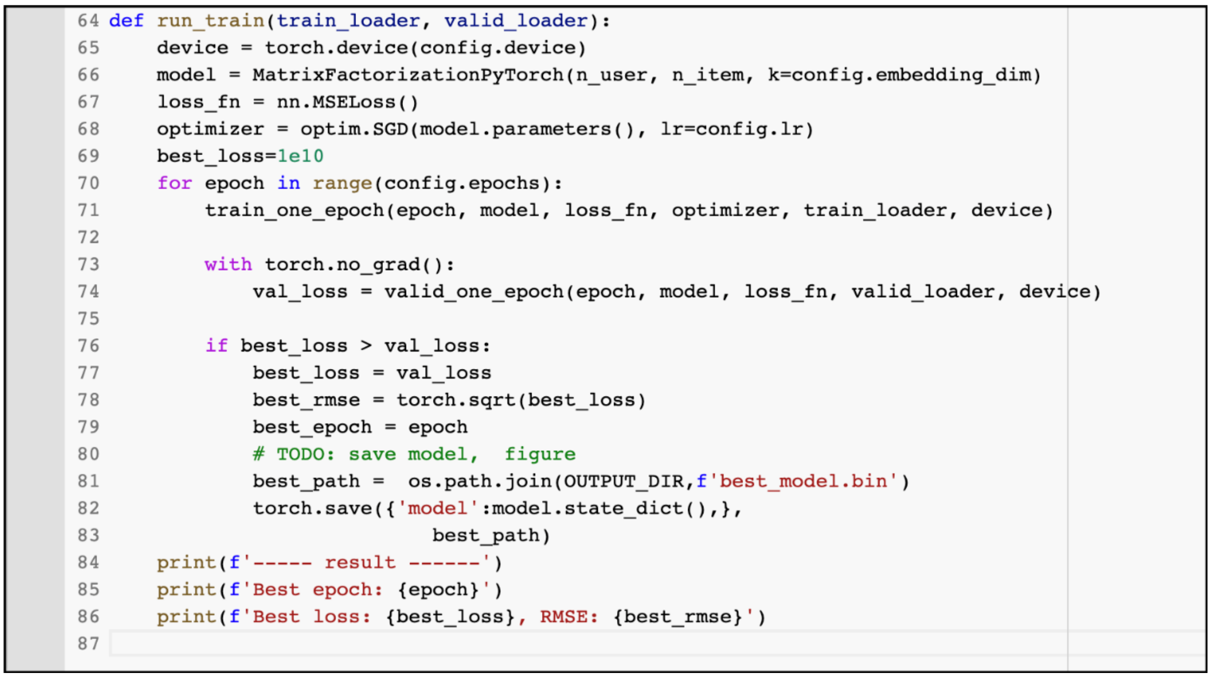


```python
class MatrixFactorizationPyTorch(nn.Module):
    def __init__(self, n_user, n_item, k=20):
        """
        n_user: user num
        n_item: item num
        k: embedding dim
        """
        super().__init__()
        self.user_factors = nn.Embedding(n_user, k, sparse=True)
        self.item_factors = nn.Embedding(n_item, k, sparse=True)

    def forward(self, user, item):
        #print(user, item)
        u_emb = self.user_factors(user)
        i_emb = self.item_factors(item)

        return (u_emb * i_emb).sum(axis=1)
```

Implementing Matrix Factorisation

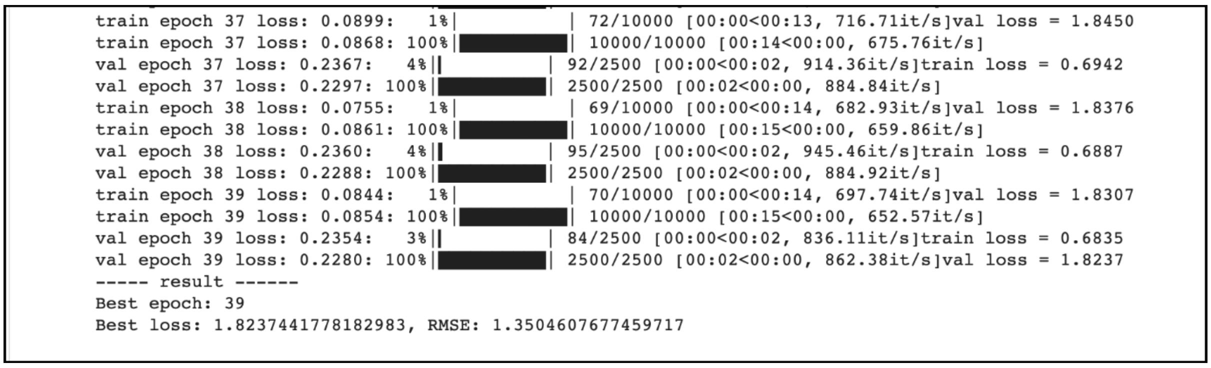
$$\tilde{r}_{ui} = \sum_{f=0}^{nfactors} H_{u,f} W_{f,i}$$

The python implementation is as shown below. We are using PyTorch libraries to develop the model. It consists of the customs class defined along with the loss function. The optimizer we chose is SGD. The model is trained for 40 epochs. The least loss model will be saved separately.



```python
def run_train(train_loader, valid_loader):
    device = torch.device(config.device)
    model = MatrixFactorizationPyTorch(n_user, n_item, k=config.embedding_dim)
    loss_fn = nn.MSELoss()
    optimizer = optim.SGD(model.parameters(), lr=config.lr)
    best_loss=1e10
    for epoch in range(config.epochs):
        train_one_epoch(epoch, model, loss_fn, optimizer, train_loader, device)

        with torch.no_grad():
            val_loss = valid_one_epoch(epoch, model, loss_fn, valid_loader, device)

        if best_loss > val_loss:
            best_loss = val_loss
            best_rmse = torch.sqrt(best_loss)
            best_epoch = epoch
            # TODO: save model,  figure
            best_path =  os.path.join(OUTPUT_DIR,f'best_model.bin')
            torch.save({'model':model.state_dict(),},
                               best_path)
    print(f'----- result ------')
    print(f'Best epoch: {epoch}')
    print(f'Best loss: {best_loss}, RMSE: {best_rmse}')
```

Training Results

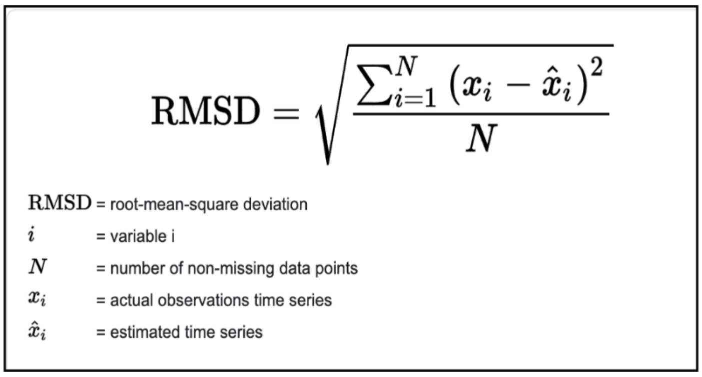


```
train epoch 37 loss: 0.0899:   1%|          | 72/10000 [00:00<00:13, 716.71it/s]val loss = 1.8450
train epoch 37 loss: 0.0868: 100%|██████████| 10000/10000 [00:14<00:00, 675.76it/s]
val epoch 37 loss: 0.2367:   4%|▍         | 92/2500 [00:00<00:02, 914.36it/s]train loss = 0.6942
val epoch 37 loss: 0.2297: 100%|██████████| 2500/2500 [00:02<00:00, 884.84it/s]
train epoch 38 loss: 0.0755:   1%|          | 69/10000 [00:00<00:14, 682.93it/s]val loss = 1.8376
train epoch 38 loss: 0.0861: 100%|██████████| 10000/10000 [00:15<00:00, 659.86it/s]
val epoch 38 loss: 0.2360:   4%|▍         | 95/2500 [00:00<00:02, 945.46it/s]train loss = 0.6887
val epoch 38 loss: 0.2288: 100%|██████████| 2500/2500 [00:02<00:00, 884.92it/s]
train epoch 39 loss: 0.0844:   1%|          | 70/10000 [00:00<00:14, 697.74it/s]val loss = 1.8307
train epoch 39 loss: 0.0854: 100%|██████████| 10000/10000 [00:15<00:00, 652.57it/s]
val epoch 39 loss: 0.2354:   3%|▎         | 84/2500 [00:00<00:02, 836.11it/s]train loss = 0.6835
val epoch 39 loss: 0.2280: 100%|██████████| 2500/2500 [00:02<00:00, 862.38it/s]val loss = 1.8237
----- result ------
Best epoch: 39
Best loss: 1.8237441778182983, RMSE: 1.3504607677459717
```

The best epoch was No.39, which resulted in a validation loss of **1.8237**.

Root Mean Square Deviation (RMSD)

RMSD will help to check that the predicted values against the actual ones. It roughly gives an idea of what rating the user might provide compared to what is predicted by the model.



$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N}\left(x_i - \hat{x}_i\right)^2}{N}}$$

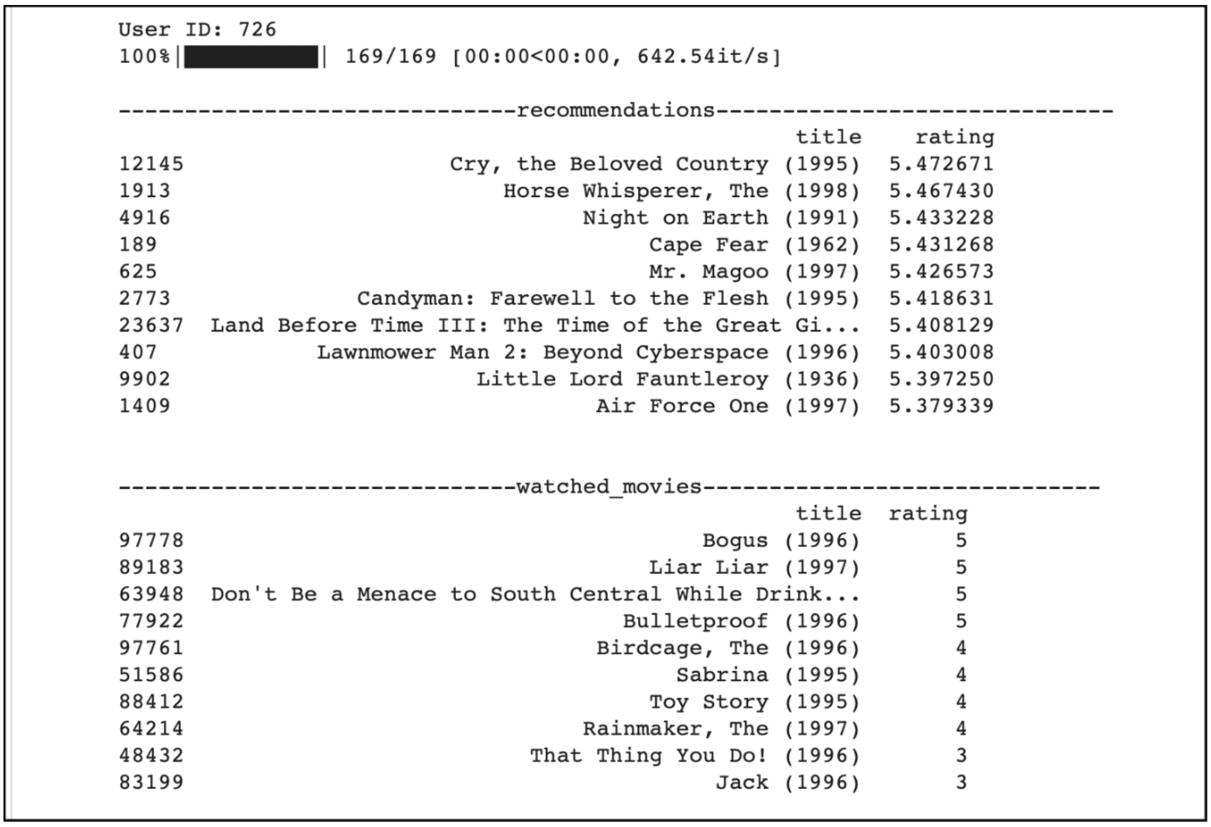
RMSD	= root-mean-square deviation
i	= variable i
N	= number of non-missing data points
x_i	= actual observations time series
$\hat{x}_i$	= estimated time series

The **RMSD** achieved by our model is **1.35**.

Predicting Movies

This will be carried out in three steps. Step 1 will be loading the best model from the training pipeline. Step 2 is storing the predicted ratings for a given user to a particular movie. Step 3 is to use those results and find the top five recommendations of any given user.

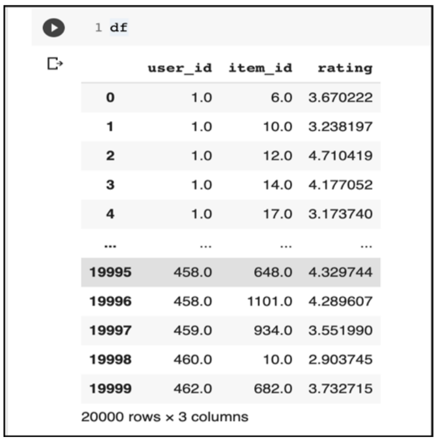
Below are the predicted movie recommendations for a random user.
Here the random user is User ID: 726



```
User ID: 726
100%|██████████| 169/169 [00:00<00:00, 642.54it/s]

------------------------------recommendations------------------------------
                                                 title    rating
12145                   Cry, the Beloved Country (1995)  5.472671
1913                        Horse Whisperer, The (1998)  5.467430
4916                              Night on Earth (1991)  5.433228
189                                    Cape Fear (1962)  5.431268
625                                    Mr. Magoo (1997)  5.426573
2773               Candyman: Farewell to the Flesh (1995)  5.418631
23637  Land Before Time III: The Time of the Great Gi...  5.408129
407                Lawnmower Man 2: Beyond Cyberspace (1996)  5.403008
9902                       Little Lord Fauntleroy (1936)  5.397250
1409                                 Air Force One (1997)  5.379339


------------------------------watched_movies------------------------------
                                                 title  rating
97778                                     Bogus (1996)       5
89183                                 Liar Liar (1997)       5
63948  Don't Be a Menace to South Central While Drink...       5
77922                               Bulletproof (1996)       5
97761                             Birdcage, The (1996)       4
51586                                   Sabrina (1995)       4
88412                                 Toy Story (1995)       4
64214                            Rainmaker, The (1997)       4
48432                        That Thing You Do! (1996)       3
83199                                      Jack (1996)       3
```

Recall

The recall is defined by True Positives and False Positives. It is used to explain the behaviour of the model. The higher the value of recall results in better predictions. A true positive will be defined when the actual rating by a user is greater than their mean rating over all the movies, and the model prediction result is also great than their mean. A false positive will be defined when the actual rating by a user is greater than their mean rating over all the movies but the



```
1 df
```

	user_id	item_id	rating
0	1.0	6.0	3.670222
1	1.0	10.0	3.238197
2	1.0	12.0	4.710419
3	1.0	14.0	4.177052
4	1.0	17.0	3.173740
...	...	...	...
19995	458.0	648.0	4.329744
19996	458.0	1101.0	4.289607
19997	459.0	934.0	3.551990
19998	460.0	10.0	2.903745
19999	462.0	682.0	3.732715

20000 rows × 3 columns

The figure shows the mean rating for user x over item id.

The test dataset is being used, which has 20k rows.

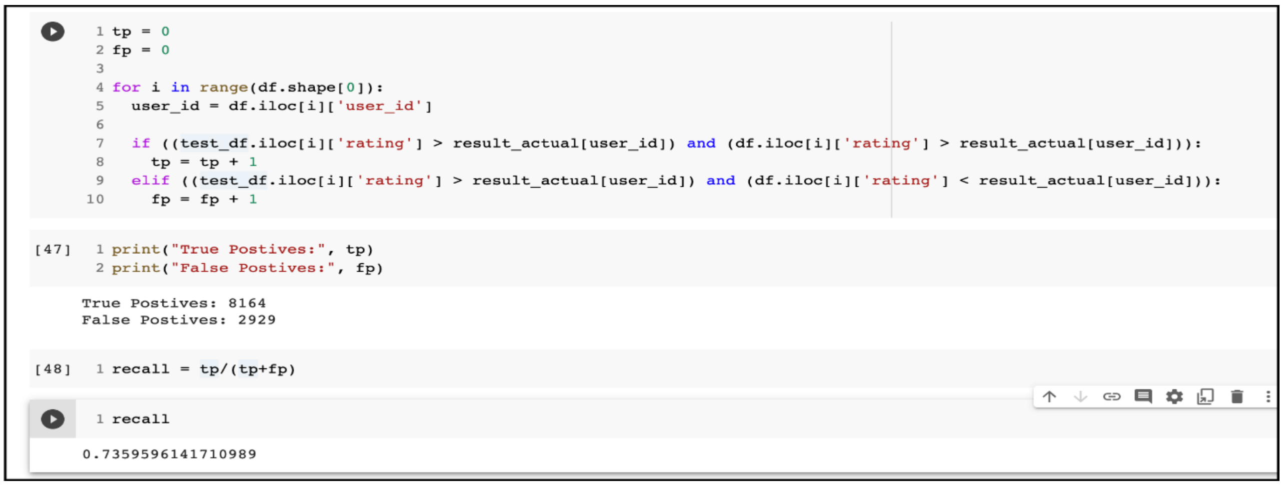
Below is the code to find the True Positives and False Positives from the values predicted by the model.

True Positives: **8164**

False Positives: **2929**

Recall = **0.736**

Thus, the moving a **73**% accuracy in predicting the correct rating for a movie.



```
tp = 0
fp = 0

for i in range(df.shape[0]):
  user_id = df.iloc[i]['user_id']

  if ((test_df.iloc[i]['rating'] > result_actual[user_id]) and (df.iloc[i]['rating'] > result_actual[user_id])):
    tp = tp + 1
  elif ((test_df.iloc[i]['rating'] > result_actual[user_id]) and (df.iloc[i]['rating'] < result_actual[user_id])):
    fp = fp + 1
```

```
[47] print("True Postives:", tp)
     print("False Postives:", fp)

True Postives: 8164
False Postives: 2929
```

```
[48] recall = tp/(tp+fp)
```

```
recall

0.7359596141710989
```

6. Conclusion

Over an era of time, the recommender system has been an efficient way of filtering significant information. Also, in this paper, we have presented a basic overview and literature review of the recommender system. From analysing different filtering technique that has been used for building recommender system on literature review section, we noticed that collaborative filtering had been the best technique to provide a personalized movie recommendation system. Using the collaborative filtering technique, we have also proposed a customized Matrix Factorization algorithm to predict the rating and provide the recommendations. The customized model is developed using PyTorch libraries; the model has trained 40 epochs where the best validation loss was found on the 39th epoch with **1.8237**. The recall was used as the evaluation metrics for the model with a value of **0.736**. Moreover, we have also visualised the dataset for a better understanding of its distances, neighbours, communities, and proximities and modularity by implementing the software Gephi. ***Network Graph visualization*** was implemented and distinguished based on modularity values, where the interesting fact was found on ***Network Summary Statistics.***

7. Future Work

Working on a project like this has been a fantastic experience because we were able to look at a dataset from various angles, including graphical visualisation (Gephi) and developing algorithm by making improvements in the currently existing technology. We aim to explore advanced approaches to accuracy. Some of these include algorithms such as Bayesian time SVD, Graphtec and GC-MC. This methodology could also help us better analyse datasets in situations where few people prefer to rate the movies they've seen, making it more difficult to forecast whether another user will enjoy the film or not. We might be able to better analyse these users' habits and use that information to improve the accuracy of our recommendation system if we go any further.

References

1. Patel, B., Desai, P. & Panchal, U. 2017, 'Methods of recommender system: A review', *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pp. 1-4.

2. Jain, S., Grover, A., Thakur, P.S. & Choudhary, S.K. 2015, 'Trends, problems and solutions of recommender system', *International Conference on Computing, Communication & Automation*, pp. 955-8.

3. Yiu-Kai, Ng., 2017, ‘MovRec: a personalised movie recommendation system for children based on online movie features’, *International Journal of Web Information Systems,* vol. 13, no. 4, pp 445-470.
4. BBC News 2020, *Netflix gets 16 million new sign-ups thanks to lockdown*, viewed 22 April 2020, <https://www.bbc.com/news/business-52376022>.
5. Li, N., Sheng, Y. & Ni, H. 2019, 'CoNN: Collaborative Neural Network for Personalized Representation Learning with Application to Scalable Task Classification', *2019 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pp. 1-5.

6. Lu, S., Wang, B., Wang, H. & Hong, Q. 2018, 'A Hybrid Collaborative Filtering Algorithm Based on KNN and Gradient Boosting', *2018 13th International Conference on Computer Science & Education (ICCSE)*, pp. 1-5.

7. Devi, M.K.K., Samy, R.T., Kumar, S.V. & Venkatesh, P. 2010, 'Probabilistic neural network approach to alleviate sparsity and cold start problems in collaborative recommender systems', *2010 IEEE International Conference on Computational Intelligence and Computing Research*, pp. 1-4.

8. Chakrabarti, N. & Das, S. 2019, 'Neural Networks and Collaborative Filtering', *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, pp. 1-4.

9. Li, B., Wan, S., Xia, H. & Qian, F. 2020, 'The Research for Recommendation System Based on Improved KNN Algorithm', *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, pp. 796-8.

10. Ahuja, R., Solanki, A. & Nayyar, A. 2019, 'Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbour', *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 263-8.