CS 577 Final Project Report

Toluwani Oyewusi, Vimarsh Patel
Department of Computer Science
Illinois Institute of Technology
November 15, 2022

Movie Recommendation System

Abstract

This report contains details of the implementation of a movie recommender system using the deep learning approach. We also explore collaborative filtering which is used by many subscription based websites like Netflix, Youtube etc. We are able to predict a certain list of movies which the user has not seen using our model.The system is trained using the subset of MovieLens 25M Dataset which has 25 million movie ratings and one million tags applied to 62,000 movies by 162,000 users. Since this dataset is very large, we are implementing our model on MovieLens 100K Dataset. In this paper, we go through various methods of approach in relation to recommendation systems. Recommendation systems play a major part in advertising, streaming services and social media. It helps to give the user information or content that would be most beneficial to that person.

A. Problem Statement

A large number of movies are made every day and released in theaters for the public to watch. Everyone does not like the same kind of movies to watch and have different

tastes. Our aim with this project is to give movie recommendations to the user so they have a better knowledge of the movies they want to see. Recommendation systems make people's life easier by finding new movies the user likes to watch. These systems make the problem of choice between various media manageable.[2] The main inspirations for this project are streaming services like Netflix, Amazon Prime and Hulu, where majority of their streaming hours can be attributed to the recommendation system.

These systems are very important parts of these streaming systems because they control what the users are shown and choose from. An essential part of these recommender systems is the use of collaborative filtering that controls how video content is consumed. Collaborative filtering refers to the use of video ratings from specific users or similar users to predict and recommend content that those users are most likely to enjoy. This approach separates and groups users based on similarities and differences in video preferences. It is interesting to dig into because content watched by a specific user can be recommended to a user with similar preferences who has not watched that content yet. [1]

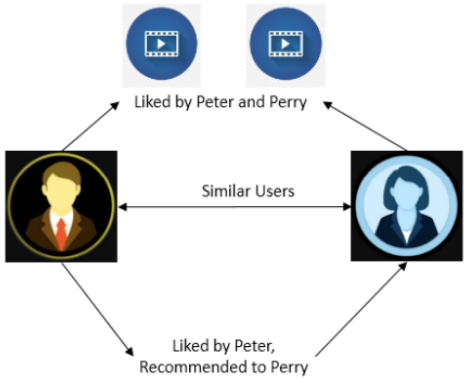


Figure 1: Collaborative Filtering .[4]

Recommendation tasks are particularly well-suited for deep learning, and it has proven itself effective in this area. Deep learning approaches have been gaining traction in the recommender system arena as a result of their cutting-edge performances and superior recommendation quality.[1] Compared to their conventional counterparts, recommender systems powered by deep learning perform better because of their capacity to handle non-linear data. The most notable advantages of using DL for suggestions are its non-linear transformation, representation learning, sequence modeling, and adaptability.

B Proposed Solution

We have given a way to build a movie recommender system using the Deep Learning model. There are various ways to achieve a recommender system using Deep Learning. They are Wide and Deep Learning , Deep Learning with Autoencoders, Deep Learning with Autoencoders with Softmax. There is not just one possibility but rather a number of them. Denoising, Stacked Denoising (SDAE), and MDAE are only a few of the many distinct types of autoencoders that are available.

There is another option of Neural Collaborative filtering(NCF). Here we have tried to implement With Deep Learning in a MultiLayer Perceptron. This system implementation is where deep learning shines because it is optimized for predictive modeling and statistics. Deep learning provides a better understanding of users' preferences, demands and relationships (similarities and differences) with each other.[1]

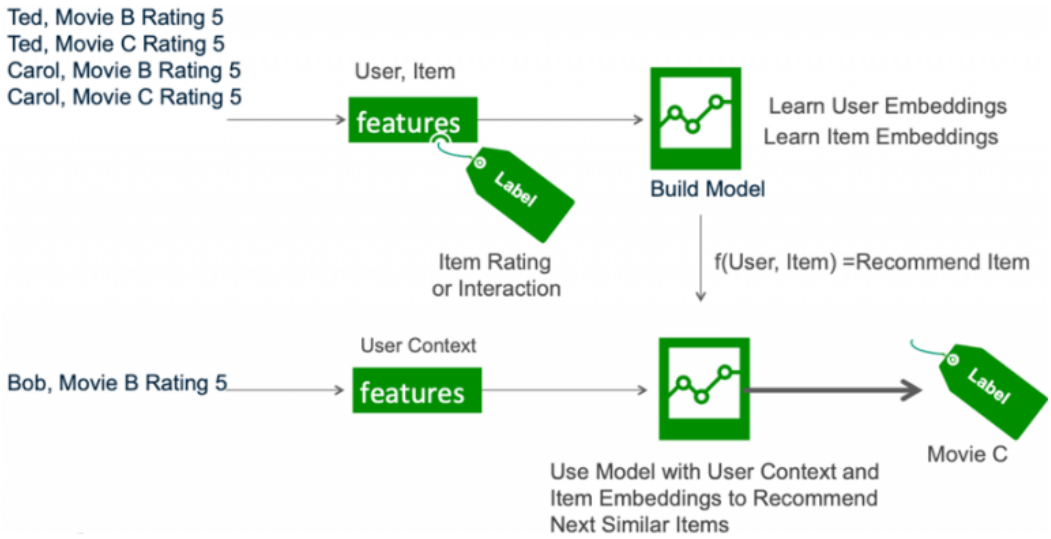


Figure 2: NCF [6]

Another suggested method of performing collaborative clustering is to use k-nearest neighbor clustering to measure similarities between user preferences. The user's profile can be represented as a vector and then cosine similarity is used to measure the similarity between other user vectors. [1]

C. Theory of Recommender Systems

Let us first take the Netflix recommendation system as a reference. Netflix uses user data such as interactions, similar users' tastes, and information about the titles of the show. In addition to that, personalization is done using other data such as time of day frequently watched, what device is used to watch and how long that user watches. It is clear that there is a significant number of factors that the system takes into account to recommend different shows to different people. [3]

The model-based method uses the ratings from each user to create a user model to predict the expected of unrated video content. This method uses a matrix that uses a

rating from a specific user for a given item. An example of a model-based method is the matrix factorization. It is represented by the formula below:

$$r_{ui} = q_i^T \times p_u \quad (1)$$

p_u provides information about the user interested in a specific item and r_{ui} describes the ratings that the user u gives to a video i . The value q_i tells us about the i-th item and returns a positive or negative figure depending on whether the user likes it or not. To find the values $q_i$and p_u, we use the formula below to factorize them to create a user to item matrix.

$$min \sum_{(u,i \in k)} (r_{ui} - q_i^T \times p_u)^2 + \delta(\left|\left|q_i\right|\right|^2 + \left|\left|p_u\right|\right|^2) \quad (2)$$

This formula allows the user to item matrix to be found and minimizes the error by reducing δ. [4]

Content based filtering is also a popular method implemented in recommendation systems. This does not use similar users but instead relies on the user's past behaviors to display what the user may prefer. This approach recommends movies to users based on similarity of genres. Movies with similar genres to the movie highly rated by the user, would be recommended to that specific user. Two matrices would be created, a genre matrix and a ratings matrix. The values in the matrix are dependent on presence and if a given genre is not present in a movie, the value would be assigned to 0. The dot product of both matrices produces the user profile matrix and after the

computation, the least distance between user and other users is calculated to measure similarity. The values with the least deviation are recommended to the user. [5]

There are several approaches to collaborative filtering and content-based filtering. and this is true even when using MLP to accomplish Deep Learning.
Training and inference are the two stages that make up the deep learning life cycle for recommendation systems. This cycle can be broken down further into sub-stages. During the training phase, the model is taught to predict user-item interaction probabilities (compute a preference score) by providing it with examples of interactions (or non-interactions) that have occurred in the past between users and items. This is done by exposing the model to data from the past. After the model has been trained to produce predictions with an adequate degree of precision, it is implemented as a service to infer the likelihood of future interactions.[6]

D. Implementation details

- **Dataset:** We have used the MovieLens 100k Dataset which contains 100,000k ratings from 1000 users on1700 movies. It is a subset of the 25M Dataset. We used this dataset since it was time saving and power efficient. Since the GPU's in our laptops are not that fast and do not have large memories, 100k Dataset was preferred.
- After we have downloaded the data, we merge the data of user, user id, movie name, movie id and ratings. This merged data will be used as the dataset for training the model. We do not require other details while training so we do not consider them.

```
ratings_dataset = pd.merge(user_ratings_data, movie_details_data,
how='inner', on='movie id')
final_dataset = ratings_dataset.groupby(by=['user id','movie title'],
as_index=False).agg({"rating":"mean"})
```

- We use the sklearn LabelEncoder() to transform the values to numerical of our dataset so it is easy for the model to understand and compute the missing values.

```
user_enc = LabelEncoder()
final_dataset['user'] = user_enc.fit_transform(final_dataset['user
id'].values)
n_users = final_dataset['user'].nunique()
```

- Split our dataset into training and test sets
- We normalize our ratings

```
rating_train = (rating_train - min_rating)/(max_rating - min_rating)
rating_test = (rating_test - min_rating)/(max_rating - min_rating)
```

- Our aim with the model is that we will input the user and the ratings of the movies he has seen. Our model will calculate the ratings of the movie the user has not seen. It will generate a preference score depending on the ratings of the similar movies given by the user and the ratings given to the movie by other users. Our model makes a matrix R and gives the ratings to all the movies with missing values for that particular user.
- **Model Architecture:** We are following the MultiLayer Perceptron model stated in our reference paper. The regularization and loss functions have been taken accordingly. Though we tried using different functions. But after some research and implementation RMSE as loss, L2 as regularization, SGD as optimizer and SoftMax activation in the output layer has been chosen in the end. Additionally we have added embedded layers for input and 3 dense layers with ReLu activation.

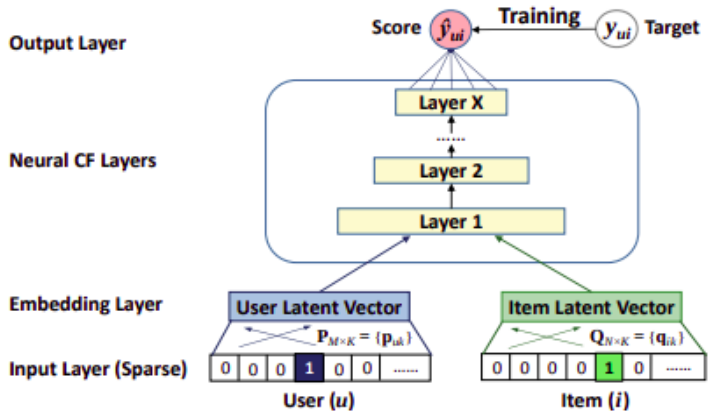


Figure 3. [6]

- **Layers :** Our model uses a combination of embedded layers, dense layers, dropout layers and the final output layer.
  - ❖ First there are 2 input layers for the user and movie.
  - ❖ We then pass our inputs to their respective Embedded layers.
  - ❖ Embedding layers are used to speed up our job. Embeddings are a method of representing items using learnt vectors. Simply described, it is a mapping of discrete objects to a sequence of continuous numbers. [7]
  - ❖ With enough data, we can train an algorithm to comprehend relationships between items and automatically learn features to describe them. They convert vectors with high dimensions to vectors with low dimensions.
  - ❖ The embedding layer allows us to turn each word into a fixed length vector of defined size. The resulting vector is dense, with real values rather than just 0s and 1s.[7]
  - ❖ We then concatenate our embedded layers.
  - ❖ After this it is passed to 3 dense layers one after another with dimensions 256,512,1024. Dense layers are used to change the dimension of the vector. It uses linear operation to activate the input.

    output = activation(dot(input, kernel) + bias)

- ❖ Dropout has been added to the layers to prevent overfitting. It randomly assigns inputs to 0. Dropout modifies the network itself to reduce overfitting. It ensures the model is careful in learning the features and makes it more generalizable.

- **Regularization:** We have added one last regularization layer before the output to further reduce overfitting. It lowers the complexity of the model. L2 regularization is also called Wide Decay or Ride Regression.

  L2 Formula: $\Omega(W) = ||W||_2^2 = \sum_i \sum_j w_{ij}^2$

  L2 Loss :

$$\hat{\mathcal{L}}(W) = \frac{\alpha}{2}||W||_2^2 + \mathcal{L}(W) = \frac{\alpha}{2}\sum_i \sum_j w_{ij}^2 + \mathcal{L}(W)$$

  Alpha is the regularization rate.

- **Loss :** It refers to the deviations that occur between the values that are predicted by a model and the actual values that are wanted. To determine the Root Mean Squared Error, simply take the square root of the Mean Squared Error (MSE)

$$\mathbf{L} = \sqrt{\frac{1}{N}[\,\Sigma(\hat{Y} - Y)^2]}$$

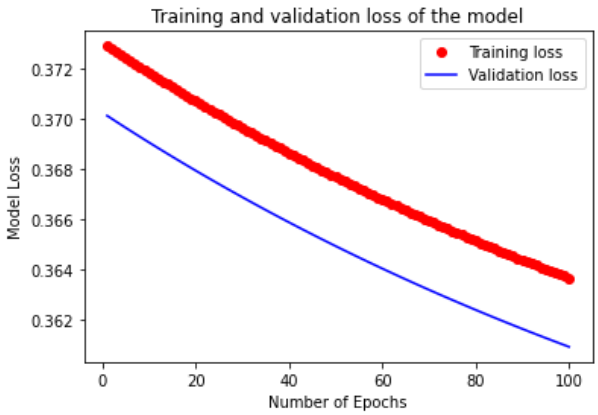
  We choose this loss function because we are doing regression analysis.
  This is advantageous in the context of a recommender system since forecasting a high rating for an item that the user did not enjoy has a significant influence on the quality of the suggestions. Smaller prediction errors, on the other hand, are likely to result in recommendations that are still useful—the regression may not be

precisely accurate, but the highest anticipated rating is likely to be relevant to the user.[1]

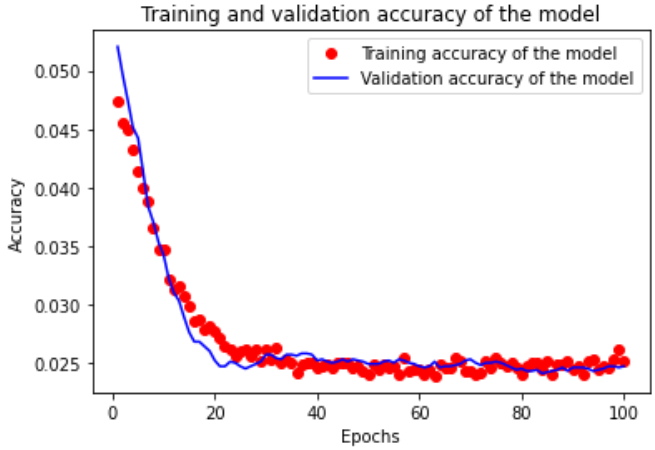
- **Softmax :** In neural network models that predict a multinomial probability distribution, the softmax function is utilized as the activation function in the output layer. In other words, softmax is employed as the activation function in multi-class classification issues when class membership on more than two class labels is needed. It makes a probability distribution out of the model's predictions for all the things. Then, it finds the maximum probability of the things that have been seen compared to the things that have not been seen.[8]
- **SGD :** SGD is more likely to converge to the minimums at flat or unsymmetrical reservoirs or pits, which are often better at generalization than other types of minimums. Most of the time, SGD enhances performance of the system gradually, but it could get better test results.
- After our model is trained, we find the list of unseen movies of the user and test our model and predict the ratings of those movies and display the list of the movies
- Finally, we build the recommendation system where we ask for the input of the user id and the number of movies to be recommended and then call our model and pass both the values to predict the movies

E. Results and Discussion

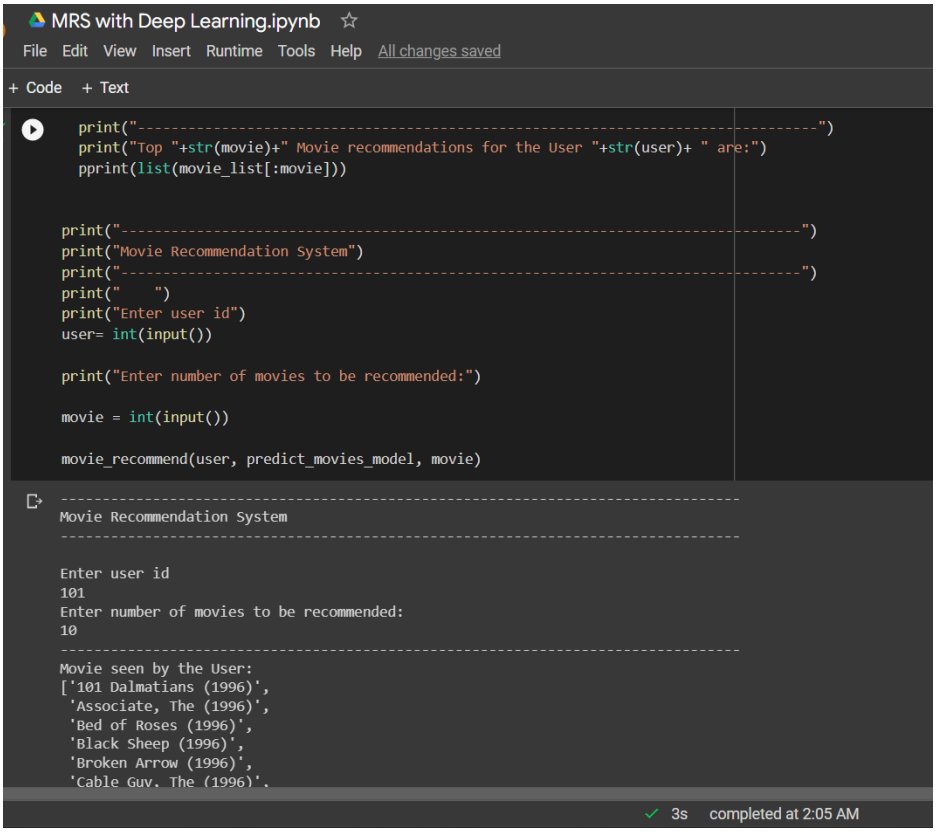
- The model's training and validation loss

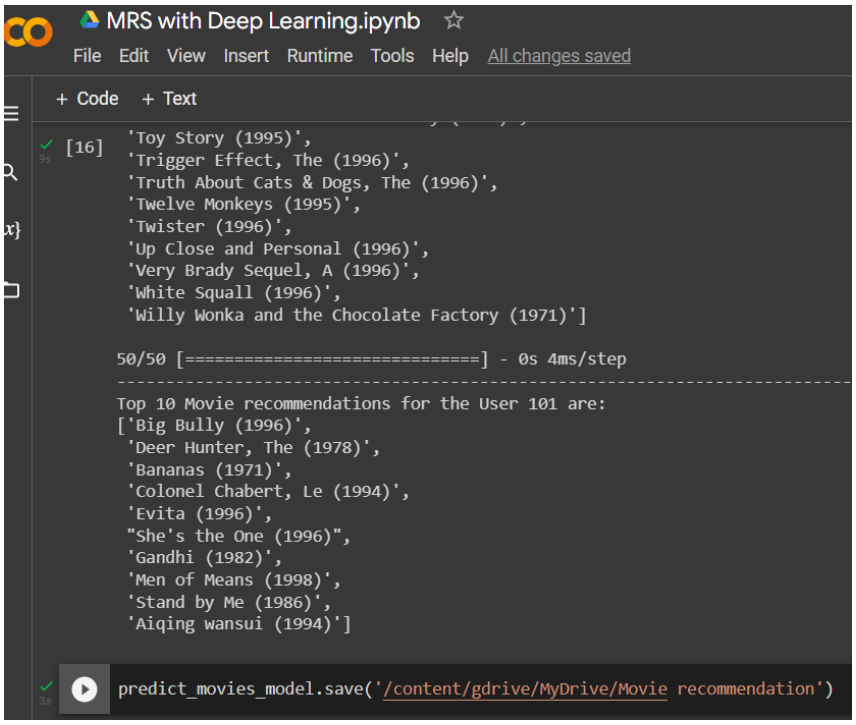


- The model's training and validation accuracy

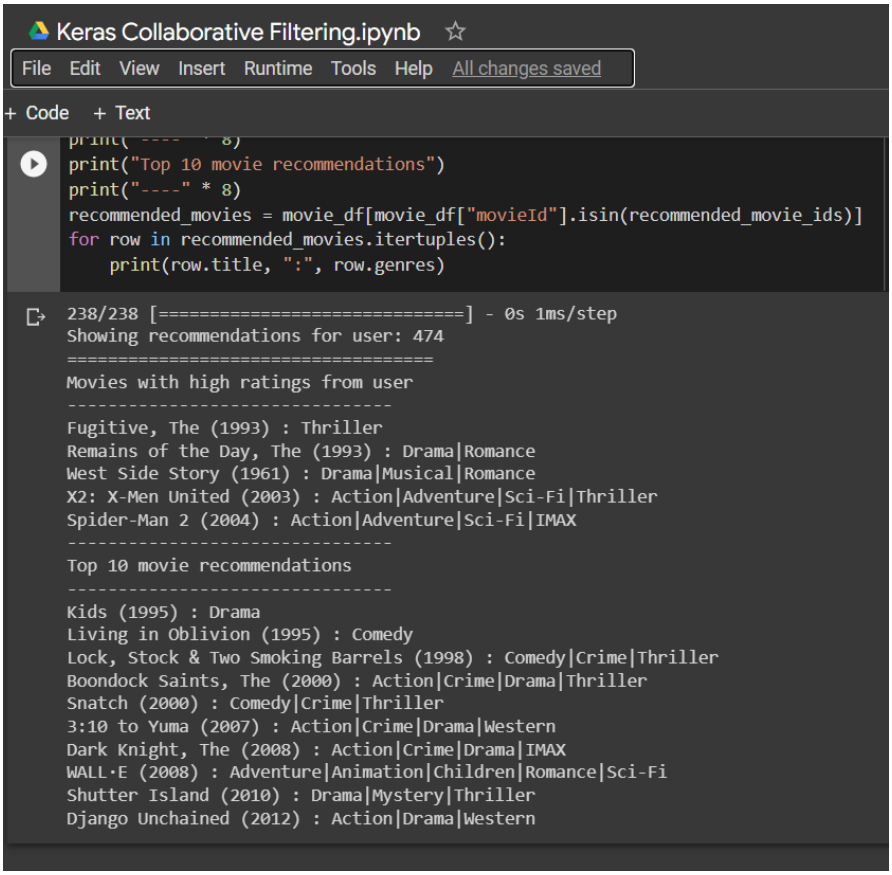


- With our model we are successfully able to predict the movies for a specific user. We do achieve exceptional results since our computers are not that efficient. But we are able to achieve our purpose and build a recommendation system.


MRS with Deep Learning.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
print("----------")
print("Top "+str(movie)+" Movie recommendations for the User "+str(user)+ " are:")
pprint(list(movie_list[:movie]))
print("----------")
print("Movie Recommendation System")
print("----------")
print("   ")
print("Enter user id")
user= int(input())
print("Enter number of movies to be recommended:")
movie = int(input())
movie_recommend(user, predict_movies_model, movie)
Movie Recommendation System
Enter user id
101
Enter number of movies to be recommended:
10
Movie seen by the User:
['101 Dalmatians (1996)',
'Associate, The (1996)',
'Bed of Roses (1996)',
'Black Sheep (1996)',
'Broken Arrow (1996)',
'Cable Guy, The (1996)',
3s completed at 2:05 AM


MRS with Deep Learning.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[16]
'Toy Story (1995)',
'Trigger Effect, The (1996)',
'Truth About Cats & Dogs, The (1996)',
'Twelve Monkeys (1995)',
'Twister (1996)',
'Up Close and Personal (1996)',
'Very Brady Sequel, A (1996)',
'White Squall (1996)',
'Willy Wonka and the Chocolate Factory (1971)']
50/50 [==============================] - 0s 4ms/step
Top 10 Movie recommendations for the User 101 are:
['Big Bully (1996)',
'Deer Hunter, The (1978)',
'Bananas (1971)',
'Colonel Chabert, Le (1994)',
'Evita (1996)',
"She's the One (1996)",
'Gandhi (1982)',
'Men of Means (1998)',
'Stand by Me (1986)',
'Aiqing wansui (1994)']
predict_movies_model.save('/content/gdrive/MyDrive/Movie recommendation')

- We have also implemented the keras collaborative filtering approach simultaneously and predict movies through that as well.



F. Conclusion

We draw the conclusion that each given recommendation system may be constructed in a number of different ways. A movie-recommendation Deep Learning Model was developed using this method. Today's world is constantly evolving, and with that comes increasing expectations. Using deep learning to do this and enhance the suggestions in our lives may go a long way, and not just in forecasting movies. It can also be used to predict the kind of occupations we should seek, the best stream of career, and other similar examples.

G. Team Member Contributions:

- Vimarsh Patel: Performed code implementation, contributed to the report, and the retrieval of other relevant research papers.
- Toluwani Oyewusi: Assisted with code implementation and designed the report, retrieved source data, and made the class presentation.

H. References & Bibliography:

Web Sources and Paper References:

[1]. J. Lund and Y. -K. Ng, "Movie Recommendations Using the Deep Learning Approach," 2018 IEEE International Conference on Information Reuse and Integration (IRI), 2018, pp. 47-54, doi: 10.1109/IRI.2018.00015.

[2]. Gomez-Uribe, Carlos A., and Neil Hunt. "The netflix recommender system: Algorithms, business value, and innovation." *ACM Transactions on Management Information Systems (TMIS)* 6.4 (2015): 1-19.

[3]. “How Netflix's Recommendations System Works.” *Help Center*, help.netflix.com/en/node/100639.

[4]. Goyani, Mahesh, and Neha Chaurasiya. "A review of the movie recommendation system: Limitations, Survey and Challenges." *ELCVIA: electronic letters on computer vision and image analysis* 19.3 (2020): 0018-37.

[5]. Reddy, S. R. S., et al. "Content-based movie recommendation system using genre correlation." *Smart Intelligent Computing and Applications*. Springer, Singapore, 2019. 391-397

[6]. McDonald, Carol. “How to Build a Deep Learning Powered Recommender System, Part 2.” *How to Build a Deep Learning Powered Recommender System, Part 2*, 2 May 2021, https://developer.nvidia.com/blog/how-to-build-a-winning-recommendation-system-part-2-deep-learning-for-recommender-systems/#:~:text=Deep%20Learning%20for%20Recommendation&text=In%20the%20training%20phase%2C%20the,and%20items%20from%20the%20past.

[7]. Tenorio, Grace. “Building a Recommender System Using Embeddings.” *Building a Recommender System Using Embeddings*, 31 July 2019, https://drop.engineering/building-a-recommender-system-using-embeddings-de5a30e655aa.

[8]. Wu, Jiancan, Xiang Wang, Xingyu Gao, Jiawei Chen, Hongcheng Fu, Tianyu Qiu, and Xiangnan He. "On the Effectiveness of Sampled Softmax Loss for Item Recommendation." *arXiv preprint arXiv:2201.02327* (2022).

Software Libraries:

- Numpy
- Pandas
- Tensorflow
- Keras
- Sklearn
- Matplotlib
- Google Colab
- Github