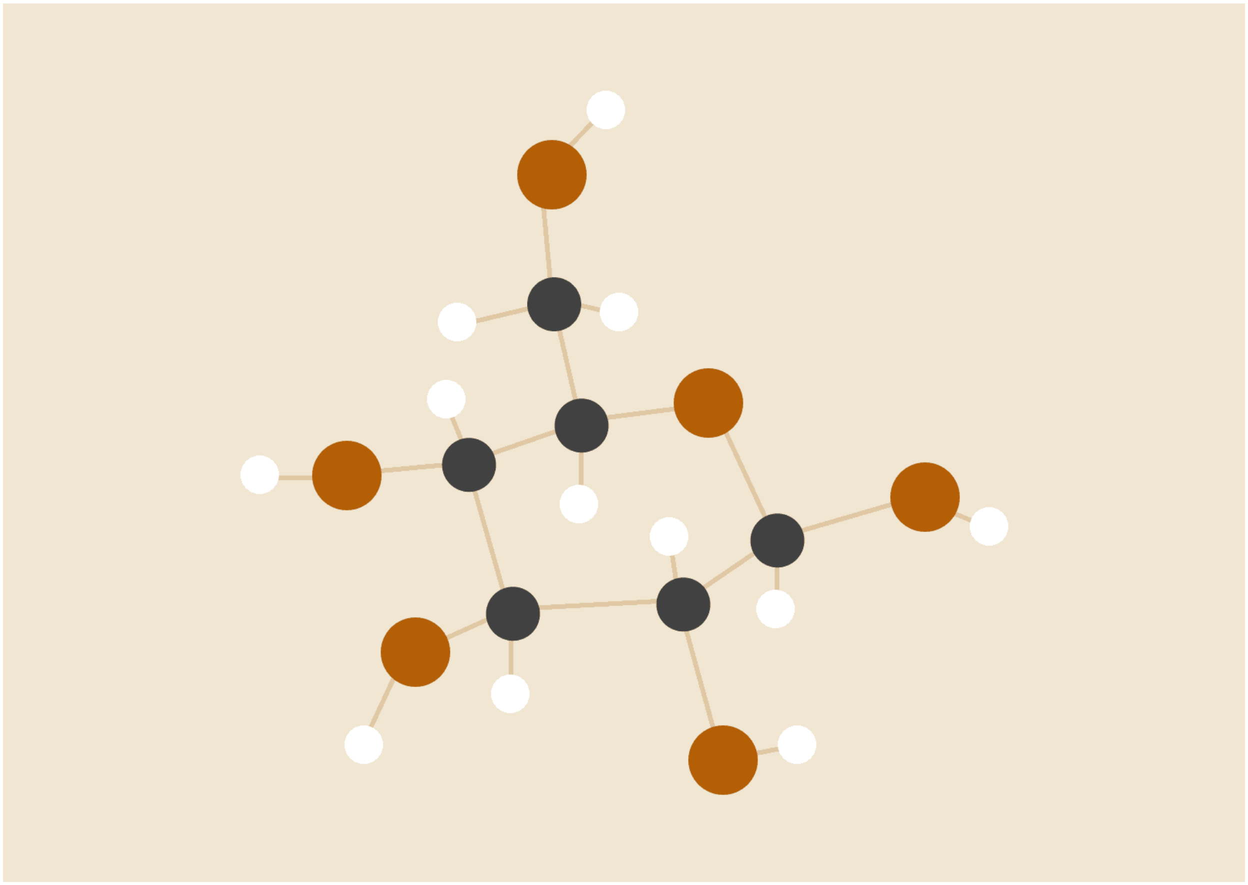
DS3 PROJECT REPORT

Movie Recommendation System



Aahan Rupal

B22183

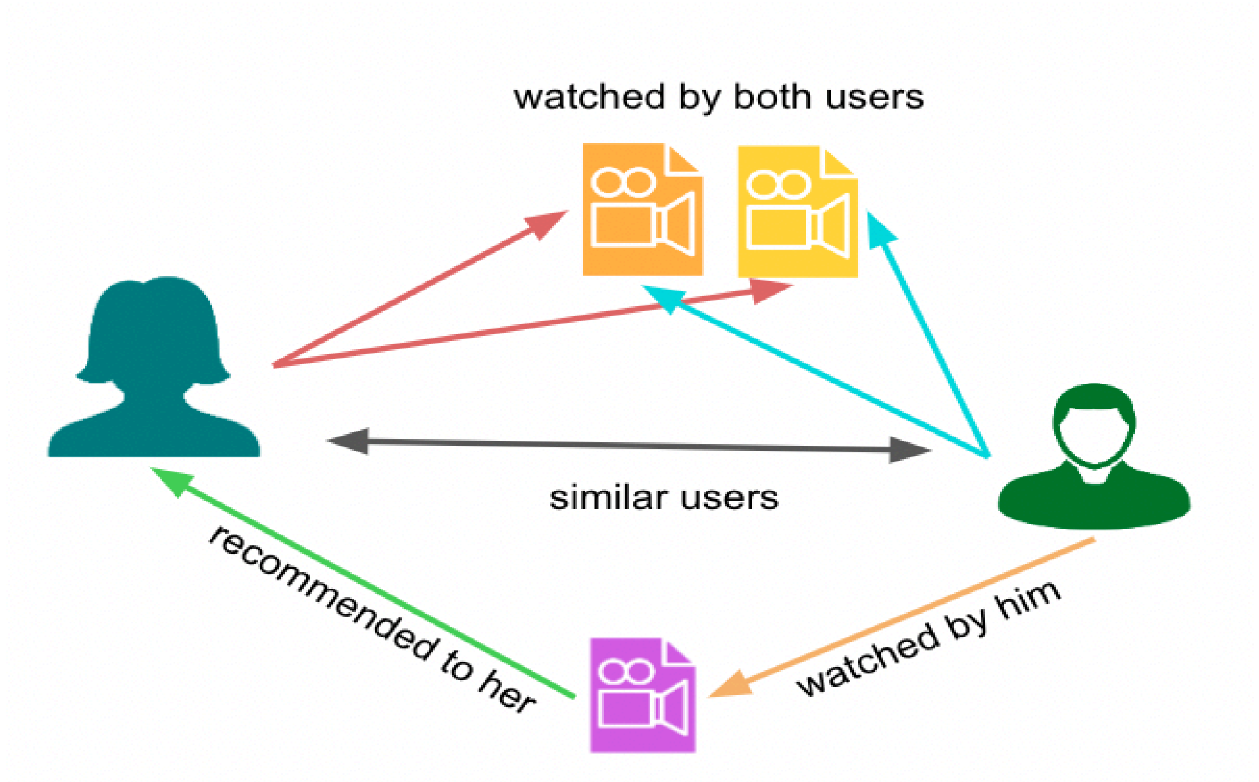
05.02.2024
2nd year, Electrical

INTRODUCTION

In the era of information overload and an abundance of choices, recommender systems play a pivotal role in aiding users in discovering content tailored to their preferences. Movie recommender systems, in particular, have become integral tools for platforms like streaming services, helping users navigate vast libraries of films. One approach that has gained prominence in the development of these systems is collaborative filtering using Matrix Factorisation.

APPROACH TO THE PROBLEM

The foundation of our recommender system lies in matrix factorization. We introduce a basic model, 'RecSysModel', and then an 'ImprovedRecSysModel' featuring user and movie embedding layers, concatenated



to capture latent features. The model is trained using Mean Squared Error (MSE) loss and optimized through the Adam optimizer. The training process is visualized through loss plots, demonstrating the model's learning progress.

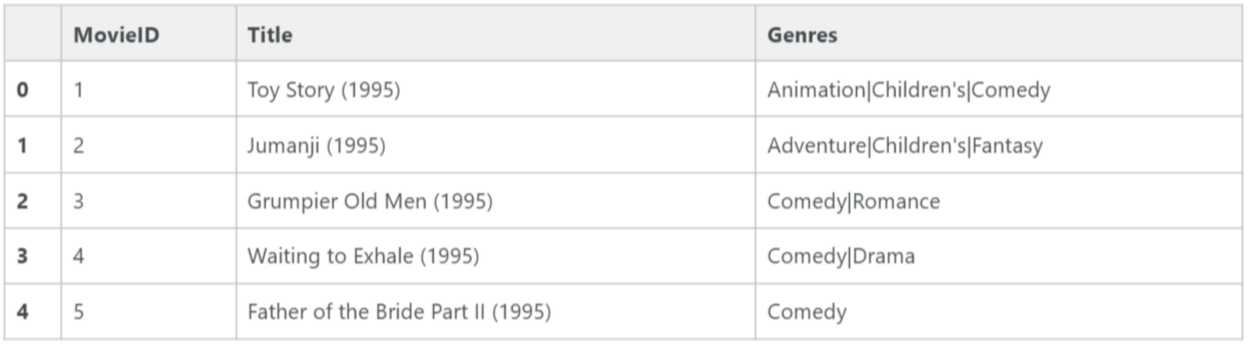
DATASETS USED

The datasets used describe 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. It contains 100836 ratings and 3683 tag applications across 9742 movies.

The data are contained in the following files-

1. **Movies.csv** :-

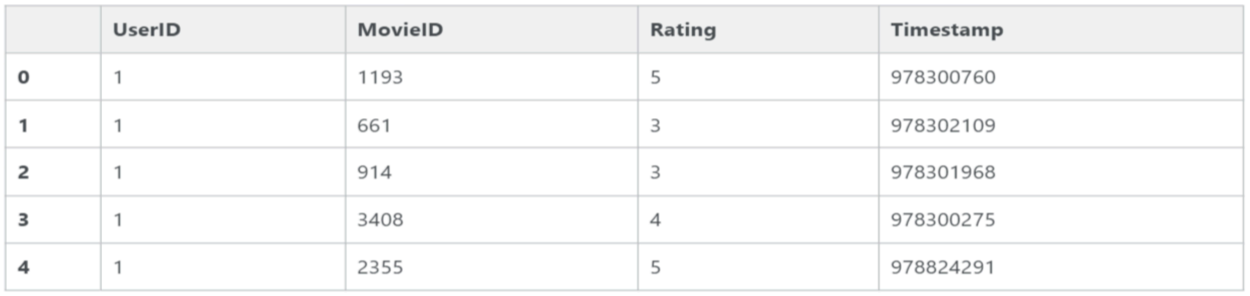
   Movie information is contained in the file `movies.csv`. Each line of this file after the header row represents one movie, and has MovieId, Title & Genres for the attributes.



	MovieID	Title	Genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy

2. **Ratings.csv** :-

   All ratings are contained in the file `ratings.csv`. Each line of this file after the header row represents one rating of one movie by one user, and has UserId, MovieId, Rating & Timest

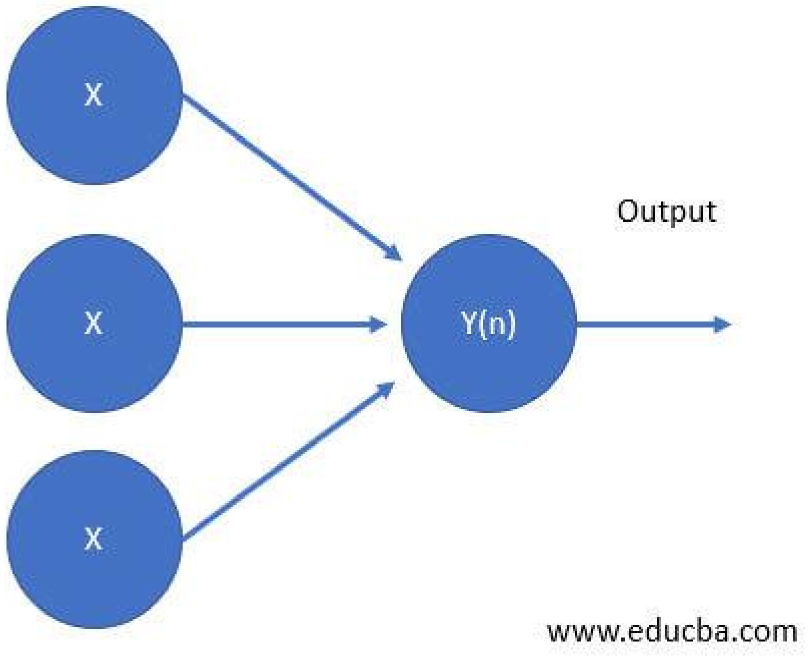


	UserID	MovieID	Rating	Timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291

MODELS

RecSysModel:-

This model is a basic collaborative filtering recommendation system. It uses embedding layers for both users and movies, concatenating them before passing through a linear layer to predict ratings. Mean Squared Error (MSE) loss is used for training.



1. Model Architecture:

The RecSysModel is a collaborative filtering model designed for movie recommendation. It utilizes matrix factorization, a common technique in recommendation systems, to capture latent features of users and movies. The model architecture consists of embedding layers for users and movies, followed by a linear layer to predict movie ratings.

2. Embedding Layers:

Users Embedding: The model incorporates an embedding layer for users, self.users_embed, with a specified embedding size of 32. Each user is represented as a dense vector of 32 dimensions, capturing latent characteristics that influence their movie preferences.

Movies Embedding: Similarly, the model includes an embedding layer for movies, self.movie_embed, with the same embedding size of 32. Each movie is represented as a 32-dimensional vector capturing its latent features.

3. Concatenation:

The user and movie embeddings are concatenated along the second dimension (dim=1) using torch.cat. This concatenation creates a combined vector representation of both the user and the movie, resulting in a vector of size 64 (32 dimensions for users + 32 dimensions for movies).

4. Linear Layer:

The concatenated vector is then passed through a linear layer, self.out, with one output unit. This linear layer computes the final predicted movie rating. The use of a linear layer allows the model to learn complex relationships and interactions between user and movie features.

5. Forward Method:

The forward method of the model, defined by the forward function, takes user and movie indices as input and computes the predicted ratings. The forward pass involves obtaining the user and movie embeddings, concatenating them, and passing through the linear layer to get the final rating prediction.

6. Training Objective:

The model is trained to minimize the Mean Squared Error (MSE) loss between predicted ratings and actual ratings. During training, the model adjusts its parameters to improve the accuracy of predicting user ratings for movies.

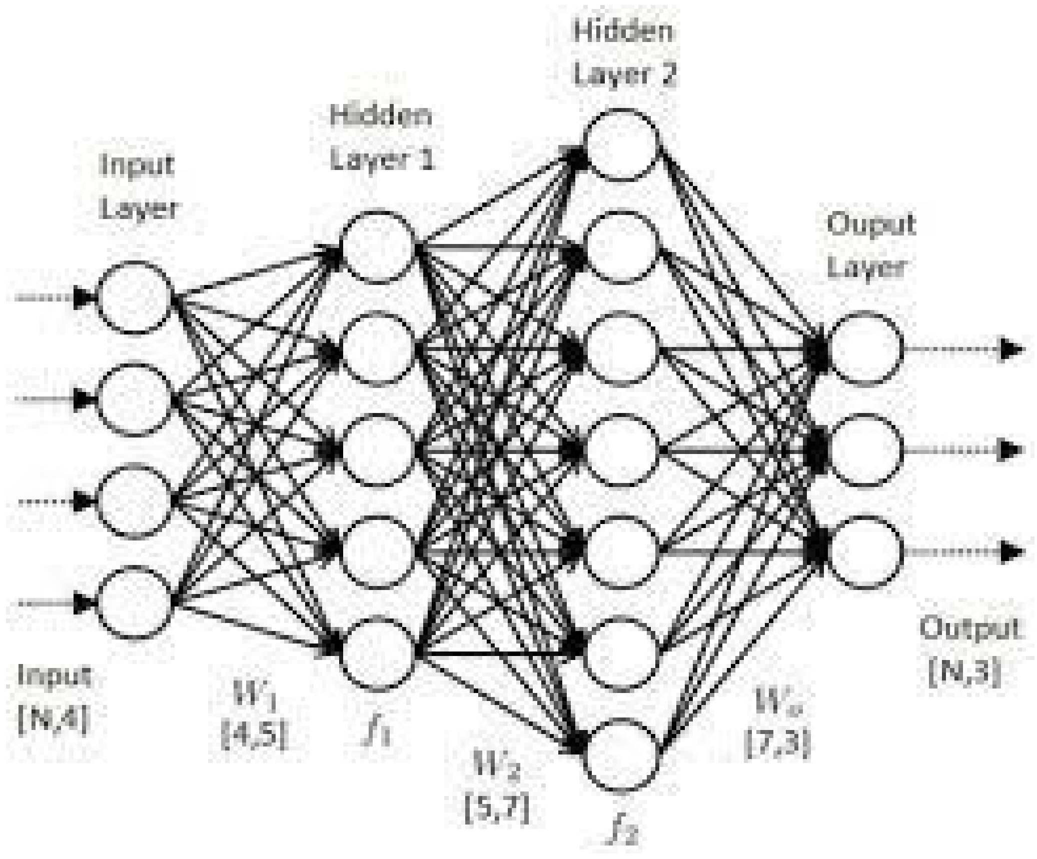
Predictions:

For instance, we predict the top 7 preferences predicted by the user number 22 using Simple Neural network.

```
Top recommendations for user 22:

{'MovieID': 1488, 'Title': "Devil's Own, The (1997)", 'Genres': 'Action|Drama|Thriller'}

{'MovieID': 8126, 'Title': 'Shock Corridor (1963)', 'Genres': 'Drama'}
```

```
{'MovieID': 2289, 'Title': 'Player, The (1992)', 'Genres': 'Comedy|Crime|Drama'}

{'MovieID': 5299, 'Title': 'My Big Fat Greek Wedding (2002)', 'Genres': 'Comedy|Romance'}

{'MovieID': 8718, 'Title': 'Snake Pit, The (1948)', 'Genres': 'Drama'}

{'MovieID': 5649, 'Title': 'Horror of Dracula (Dracula) (1958)', 'Genres': 'Horror'}

{'MovieID': 7084, 'Title': 'Play It Again, Sam (1972)', 'Genres': 'Comedy|Romance'}
```

ImprovedRecSys:



1. Model Architecture:

The ImprovedRecSysModel is an enhanced version of the collaborative filtering model designed for movie recommendation. It builds upon the RecSysModel by introducing additional layers, such as batch normalization, dropout, and a hidden linear layer, to improve the model's predictive capabilities and robustness.

2. Embedding Layers:

Users Embedding: Similar to the base model, the improved model incorporates an embedding layer for users (self.users_embed).

Movies Embedding: It also includes an embedding layer for movies (self.movie_embed). Both embedding layers have a specified embedding size, which is a hyperparameter that determines the dimensionality of the dense vectors representing users and movies.

3. Batch Normalization:

A batch normalization layer (self.batch_norm) is introduced to normalize the concatenated user and movie embeddings. This helps in stabilizing and accelerating the training process by reducing internal covariate shift.

4. Linear Layers and Dropout:

First Linear Layer (self.linear1): The concatenated and normalized embeddings are passed through a linear layer with hidden size (hidden_size). This introduces a non-linearity to the model, enabling it to learn complex relationships between user and movie features.

Dropout Layer (self.dropout): To prevent overfitting, a dropout layer is added after the first linear layer. Dropout randomly deactivates a proportion of neurons during training, helping the model generalize better to unseen data.

Second Linear Layer (self.linear2): The output of the dropout layer is then passed through another linear layer with one output unit, responsible for predicting the final movie rating.

5. Forward Method:

The forward method of the model, defined by the forward function, takes user and movie indices as input and computes the predicted ratings. The forward pass involves obtaining the user and movie embeddings, concatenating them, normalizing the concatenated embeddings, passing through linear layers with non-linear activations, applying dropout, and obtaining the final rating prediction.

6. Training Objective:

Similar to the base model, the improved model is trained to minimize the Mean Squared Error (MSE) loss between predicted ratings and actual ratings. The optimization process involves adjusting model parameters to improve the accuracy of predicting user ratings for movies.

7. Significance and Applicability:

The ImprovedRecSysModel is designed to address potential issues of overfitting and training instability present in simpler models. By incorporating batch normalization, dropout, and additional hidden layers, the model aims to provide more robust and accurate movie recommendations.

Predictions:

We predict the top 7 preferences predicted by the user number 22 using Simple Neural network.

```
Top recommendations for user 22:

{'MovieID': 4614, 'Title': 'Kickboxer (1989)', 'Genres': 'Action'}

{'MovieID': 2699, 'Title': 'Arachnophobia (1990)', 'Genres': 'Comedy|Horror'}

{'MovieID': 355, 'Title': 'Flintstones, The (1994)', 'Genres': 'Children|Comedy|Fantasy'}

{'MovieID': 2402, 'Title': 'Rambo: First Blood Part II (1985)', 'Genres': 'Action|Adventure|Thriller'}

{'MovieID': 1970, 'Title': 'Nightmare on Elm Street 3: Dream Warriors, A (1987)', 'Genres': 'Horror|Thriller'}

{'MovieID': 4947, 'Title': 'Gauntlet, The (1977)', 'Genres': 'Action'}
```

```
{'MovieID': 1477, 'Title': 'Love Jones (1997)', 'Genres': 'Romance'}
```

COMPARISON BETWEEN MODELS

1. Complexity:

RecSysModel is a simpler model with a single linear layer, while ImprovedRecSysModel introduces additional layers for increased complexity.

2. Non-linearity:

RecSysModel lacks explicit non-linear activation functions, while ImprovedRecSysModel incorporates ReLU activation, enhancing its ability to capture non-linear relationships.

3. Regularization:

ImprovedRecSysModel includes batch normalization and dropout, providing better regularization to mitigate overfitting.

4. Training Time:

RecSysModel may have a faster training time due to its simplicity, while ImprovedRecSysModel may take longer due to its increased complexity.

5. Performance:

The choice between the two models may depend on the dataset and the complexity of patterns in the user-movie interactions. A more complex dataset might benefit from the additional layers in ImprovedRecSysModel.

6. Recommendation Quality:

The quality of recommendations may be influenced by the ability of the model to capture nuanced user preferences, where ImprovedRecSysModel has the potential advantage.

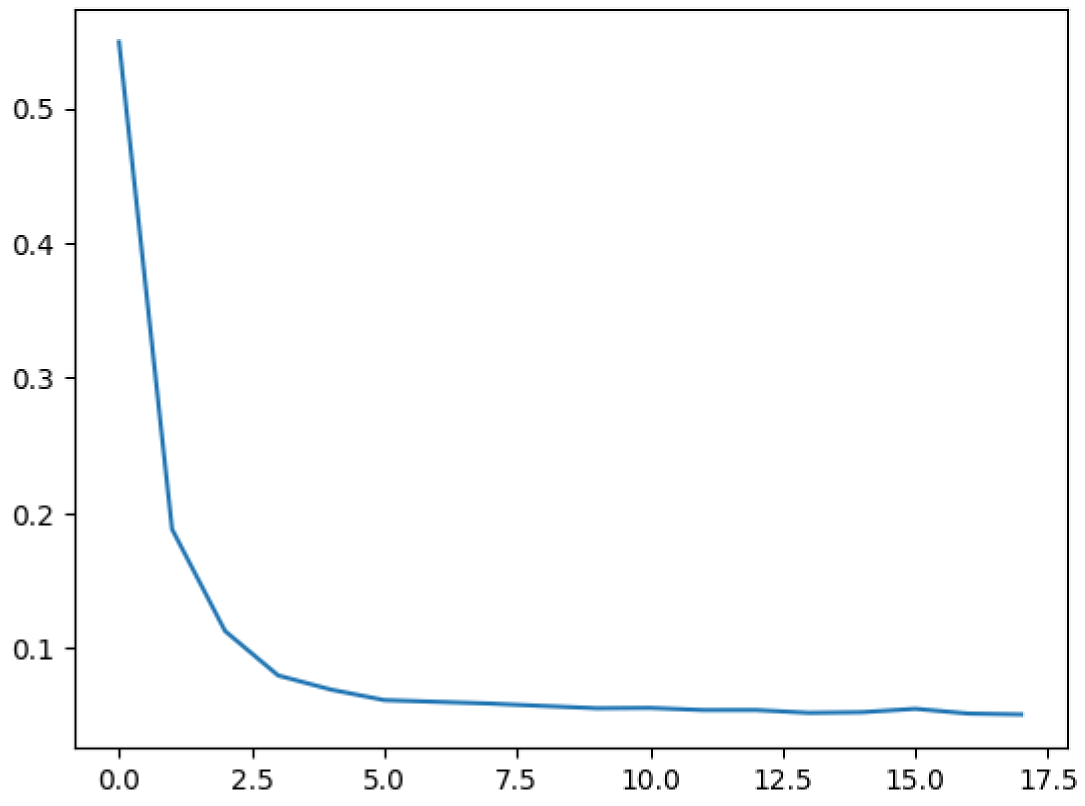


Fig-1: Cost Function for Simple NN

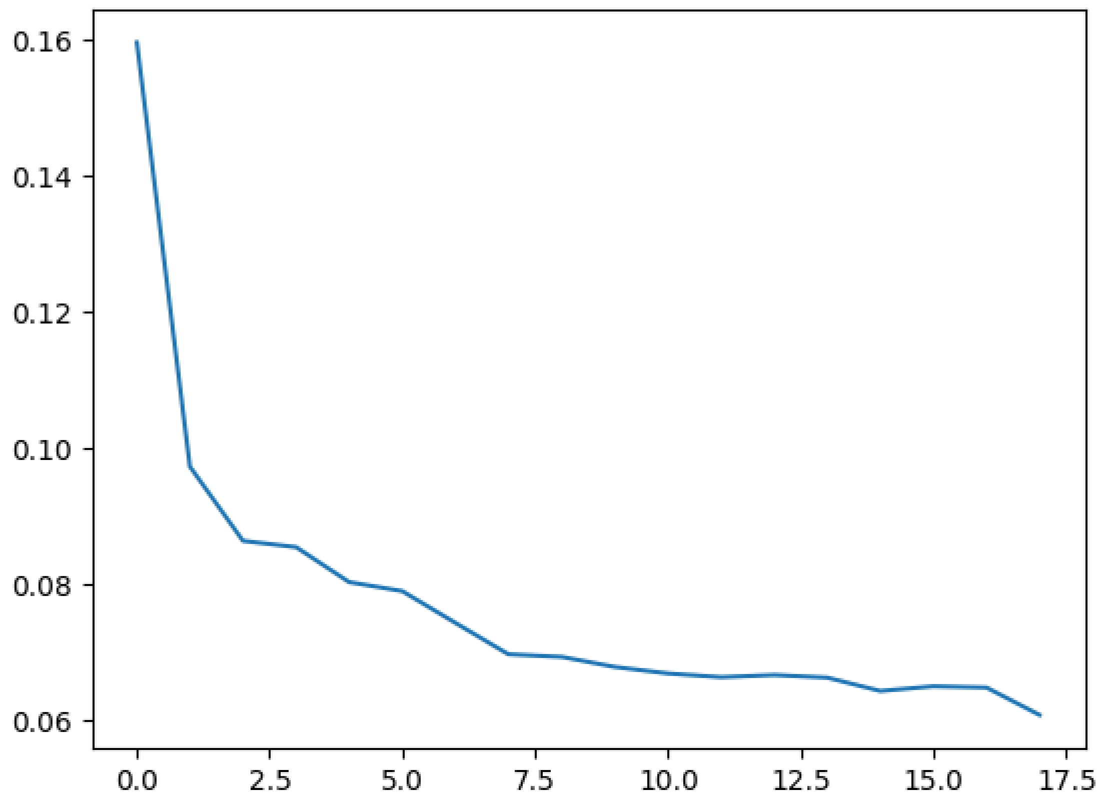


Fig-2: Cost Function for Deep NN

CONCLUSION

In summary, the choice between the two models depends on the trade-off between model complexity, training time, and the dataset's characteristics. If a simpler model is sufficient for the dataset, RecSysModel might be preferred for its simplicity and faster training. However, for more complex datasets, ImprovedRecSysModel offers the potential for better capturing intricate user-movie interactions, despite the trade-off in training time.