Introduction

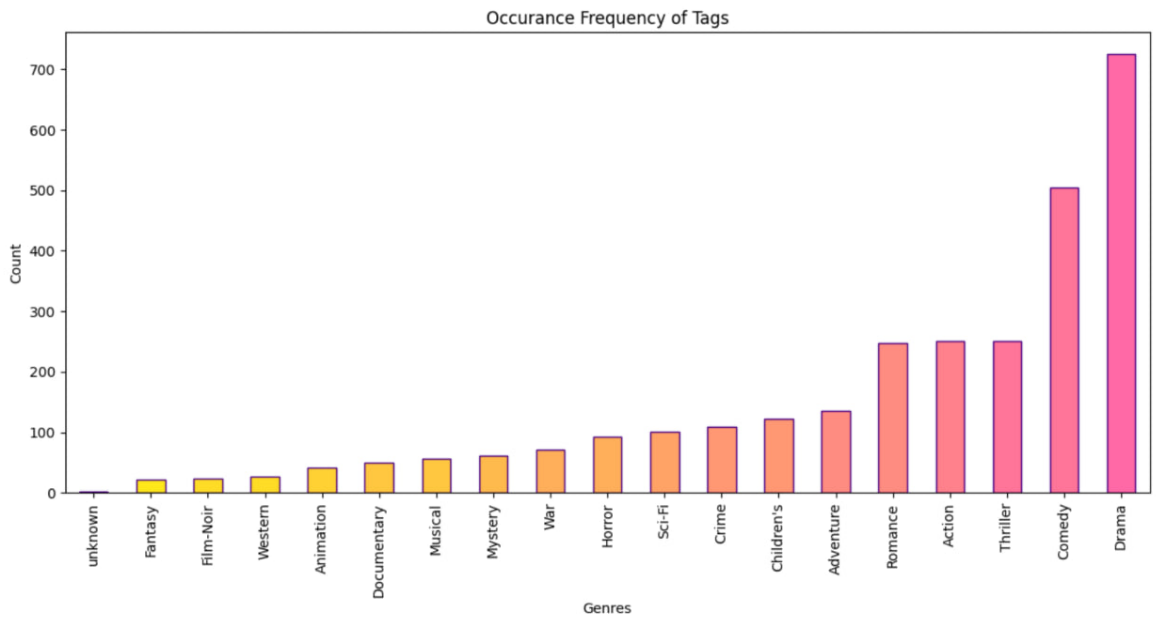
The DNN-MF recommender system is a hybrid model that combines the strengths of both the Deep Neural Network (DNN) and Matrix Factorization (MF) models. This system is designed to capture the complex non-linear relationships between the user and item features, as well as the latent factors in the user-item interactions. The system is capable of handling large datasets and complex data structures effectively, and it learns from the data to improve its predictions over time.

Data Analysis

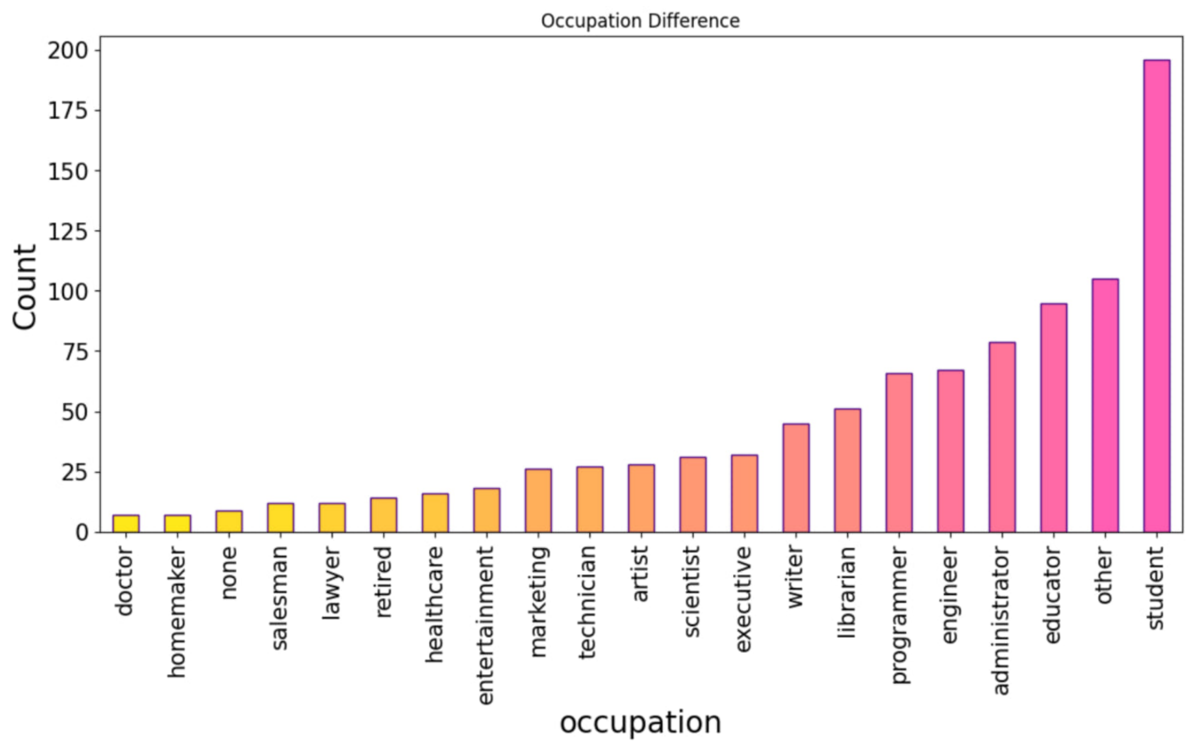
The initial data exploration phase involved loading and visualizing the raw data. The data was divided into several categories: User Data, Item Data, and Rating Data. The code reads the data from various CSV files and assigns appropriate column names. The data is then displayed using the `head()` method. The code also prints out some information about the data, such as the number of users, items, and ratings.

The code also generates plots to visualize the distribution of genres among movies and the distribution of occupations among users. These plots help in understanding the data distribution and can be used to make informed decisions during the model building phase.

Occurrence Frequency of Tags:



Occupation Difference:



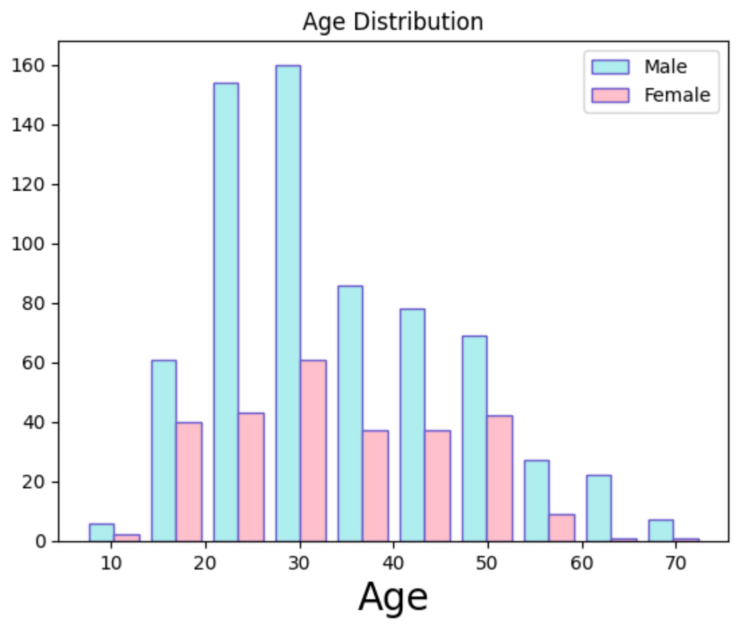
Top 10:

```
movie title
Star Wars (1977)                        583
Contact (1997)                          509
Fargo (1996)                            508
Return of the Jedi (1983)               507
Liar Liar (1997)                        485
English Patient, The (1996)             481
Scream (1996)                           478
Toy Story (1995)                        452
Air Force One (1997)                    431
Independence Day (ID4) (1996)           429
dtype: int64
```

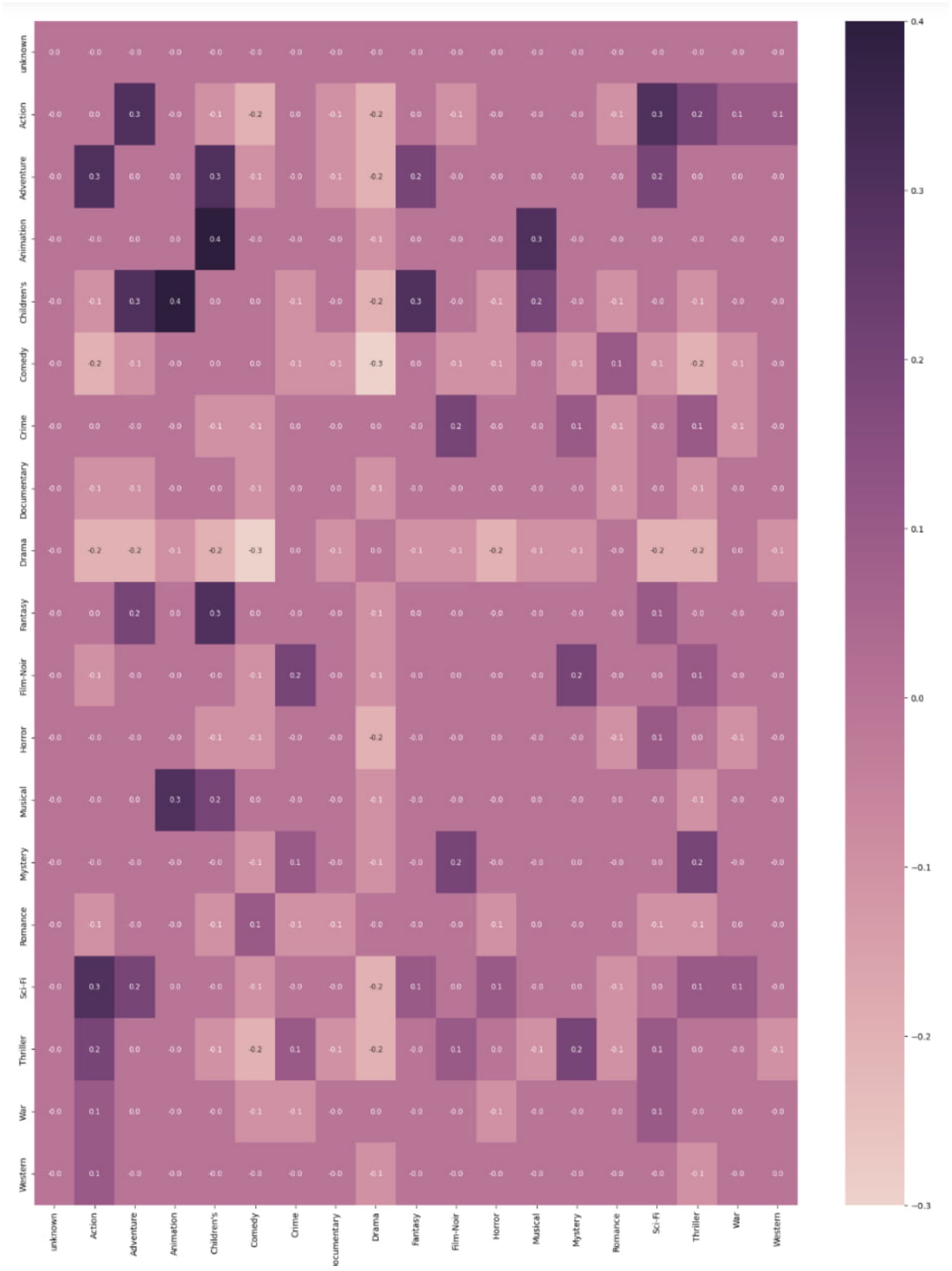
Top 20 with ratings:

	size	rating mean
movie title		
They Made Me a Criminal (1939)	1	5.000000
Marlene Dietrich: Shadow and Light (1996)	1	5.000000
Saint of Fort Washington, The (1993)	2	5.000000
Someone Else's America (1995)	1	5.000000
Star Kid (1997)	3	5.000000
Great Day in Harlem, A (1994)	1	5.000000
Aiqing wansui (1994)	1	5.000000
Santa with Muscles (1996)	2	5.000000
Prefontaine (1997)	3	5.000000
Entertaining Angels: The Dorothy Day Story (1996)	1	5.000000
Pather Panchali (1955)	8	4.625000
Some Mother's Son (1996)	2	4.500000
Maya Lin: A Strong Clear Vision (1994)	4	4.500000
Anna (1996)	2	4.500000
Everest (1998)	2	4.500000
Close Shave, A (1995)	112	4.491071
Schindler's List (1993)	298	4.466443
Wrong Trousers, The (1993)	118	4.466102
Casablanca (1942)	243	4.456790
Wallace & Gromit: The Best of Aardman Animation (1996)	67	4.447761

Age distribution:



Genres correlation:

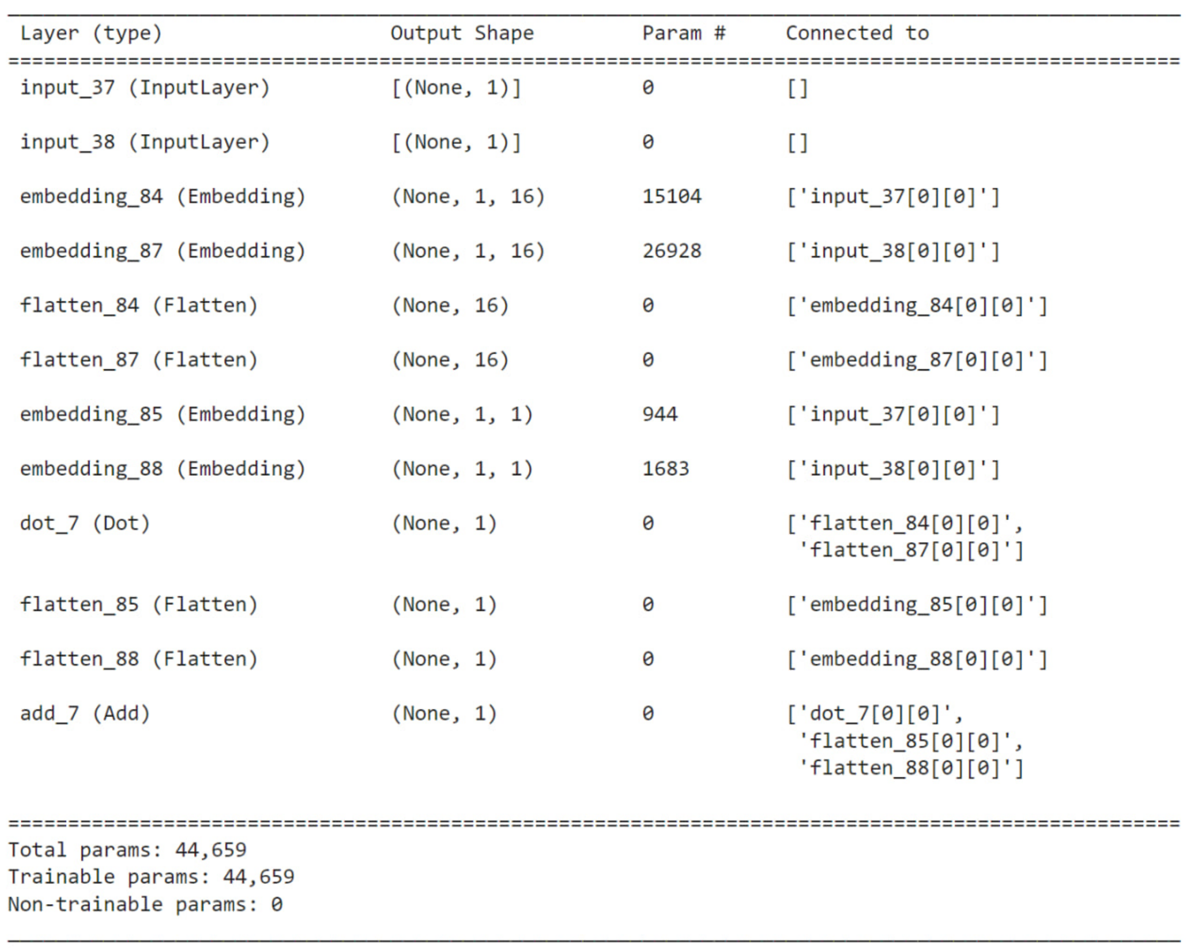


Model Implementation

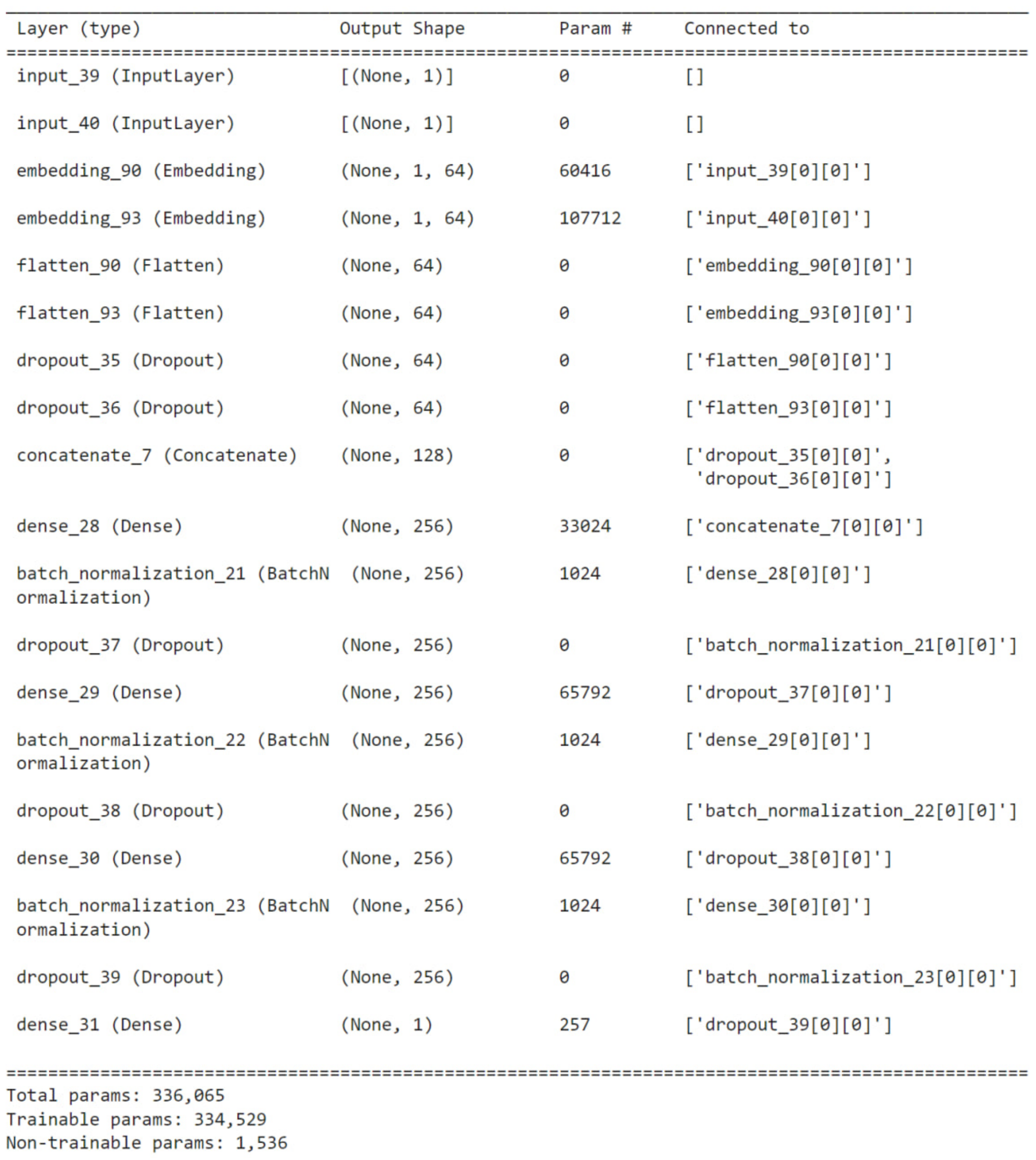
The data preprocessing phase involved several steps: Creating Pivot Tables, Scaling User Age, Splitting the Data, and Scaling Ratings. The preprocessed data was then saved to CSV files for later use.

The model building phase involved building a DNN-MF recommender system. The models were trained on each of the seven datasets separately and in turn (u1, u2, u3, u4, u5, ua, ub).

Some summaries of the model:



```
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_37 (InputLayer)          [(None, 1)]          0           []

 input_38 (InputLayer)          [(None, 1)]          0           []

 embedding_84 (Embedding)       (None, 1, 16)        15104       ['input_37[0][0]']

 embedding_87 (Embedding)       (None, 1, 16)        26928       ['input_38[0][0]']

 flatten_84 (Flatten)           (None, 16)           0           ['embedding_84[0][0]']

 flatten_87 (Flatten)           (None, 16)           0           ['embedding_87[0][0]']

 embedding_85 (Embedding)       (None, 1, 1)         944         ['input_37[0][0]']

 embedding_88 (Embedding)       (None, 1, 1)         1683        ['input_38[0][0]']

 dot_7 (Dot)                    (None, 1)            0           ['flatten_84[0][0]',
                                                                  'flatten_87[0][0]']

 flatten_85 (Flatten)           (None, 1)            0           ['embedding_85[0][0]']

 flatten_88 (Flatten)           (None, 1)            0           ['embedding_88[0][0]']

 add_7 (Add)                    (None, 1)            0           ['dot_7[0][0]',
                                                                  'flatten_85[0][0]',
                                                                  'flatten_88[0][0]']

==================================================================================================
Total params: 44,659
Trainable params: 44,659
Non-trainable params: 0
__________________________________________________________________________________________________
```



```
_____________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
=====================================================================================================
 input_39 (InputLayer)          [(None, 1)]          0           []

 input_40 (InputLayer)          [(None, 1)]          0           []

 embedding_90 (Embedding)       (None, 1, 64)        60416       ['input_39[0][0]']

 embedding_93 (Embedding)       (None, 1, 64)        107712      ['input_40[0][0]']

 flatten_90 (Flatten)           (None, 64)           0           ['embedding_90[0][0]']

 flatten_93 (Flatten)           (None, 64)           0           ['embedding_93[0][0]']

 dropout_35 (Dropout)           (None, 64)           0           ['flatten_90[0][0]']

 dropout_36 (Dropout)           (None, 64)           0           ['flatten_93[0][0]']

 concatenate_7 (Concatenate)    (None, 128)          0           ['dropout_35[0][0]',
                                                                  'dropout_36[0][0]']

 dense_28 (Dense)               (None, 256)          33024       ['concatenate_7[0][0]']

 batch_normalization_21 (BatchN  (None, 256)         1024        ['dense_28[0][0]']
 ormalization)

 dropout_37 (Dropout)           (None, 256)          0           ['batch_normalization_21[0][0]']

 dense_29 (Dense)               (None, 256)          65792       ['dropout_37[0][0]']

 batch_normalization_22 (BatchN  (None, 256)         1024        ['dense_29[0][0]']
 ormalization)

 dropout_38 (Dropout)           (None, 256)          0           ['batch_normalization_22[0][0]']

 dense_30 (Dense)               (None, 256)          65792       ['dropout_38[0][0]']

 batch_normalization_23 (BatchN  (None, 256)         1024        ['dense_30[0][0]']
 ormalization)

 dropout_39 (Dropout)           (None, 256)          0           ['batch_normalization_23[0][0]']

 dense_31 (Dense)               (None, 1)            257         ['dropout_39[0][0]']

=====================================================================================================
Total params: 336,065
Trainable params: 334,529
Non-trainable params: 1,536
```

Model Advantages and Disadvantages

The DNN-MF recommender system has several advantages. It can capture the complex relationships between users and items, predict the ratings of the items accurately, and provide meaningful recommendations to the users. However, it also has some disadvantages. The system might not be able to handle very large datasets due to memory limitations. Also, the training process might be time-consuming, especially for large datasets.

Training Process

The training process involved fitting the model to the training data and validating it on the validation data. The best models were saved using the `ModelCheckpoint()` callback from Keras.

Evaluation

The model evaluation phase involved evaluating the performance of the trained models. The performance was measured using several metrics, including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Precision@10. The models were evaluated on the testing data, and the results were printed out.

Results

The DNN-MF recommender system was successful in capturing the complex relationships between users and items. The system was able to predict the ratings of the items accurately, as indicated by the low RMSE and high precision values. The system also provided meaningful recommendations to the users, as indicated by the high precision@10 value. The system was able to handle large datasets and complex data structures effectively. The system was also able to learn from the data and improve its predictions over time.