**“TO DEVELOP A DEEP NEURAL NETWORK FOR SOCIAL RECOMMENDATION SYSTEM”**

SUBMITTED in partial fulfillment of the

requirementfor the award of the degree

of

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted By**

**ADITYA ANAND (18365)**

**BHUSWARNA KASHYAP (18370)**

**NIDHI BHARTI (18372)**

**RAHUL NATH (18373)**

**BINONG KIRI BEY (18379)**



**2018-2022**

**ASSAM SCIENCE AND TECHNOLOGY UNIVERSITY**

**GUWAHATI**

**ASSAM ENGINEERING COLLEGE, JALUKBARI**

**GUWAHATI-781013**

**2020**

# 

# 

Assam Engineering College

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**ASSAM SCIENCE AND TECHNOLOGY UNIVERSITY**

**Certificate**



# 

Assam Engineering College

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**ASSAM SCIENCE AND TECHNOLOGY UNIVERSITY**

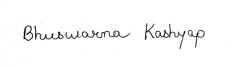
**Declaration by the Candidate**

We Aditya Anand, Bhuswarna Kashyap**,** Rahul Nath, Nidhi Bharati, Binong Kiri Bey, **Roll No** 18/365, 18/370, 18/372, 18/373, 18/379, B. Tech. students of the Department of Computer Science & Engineering, Assam Engineering College hereby declare that we have compiled this thesis reflecting all our works during the summer/ industrial training at PSG College of Technology as part of my B. Tech. curriculum.

We declare that we have included the descriptions etc of our work, and nothing has been copied/replicated from other’s work. The facts, figures, analysis, results, claims etc depicted in our report are all related to our training work.

We also declare that the same report or any substantial portion of this report has not been submitted anywhere else as part of any requirements for any degree/diploma etc.

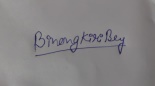


Date:18/10/2020RollNo - 18/365 Name - Aditya Anand Signature

C:\Users\ACER\Downloads\WhatsApp Image 2020-10-18 at 8.45.46 AM.jpeg RollNo - 18/365 Name - Bhuswarna Kashyap Signature

C:\Users\ACER\Downloads\Nidhi_Signature_Scan_(1).jpgRollNo - 18/365 Name - Rahul Nath Signature

RollNo - 18/365 Name - Nidhi Bharti Signature

RollNo - 18/365 Name - Binong Kiri Bey Signature



Assam Engineering College

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**ASSAM SCIENCE AND TECHNOLOGY UNIVERSITY**

**Acknowledgement**

We would like to convey our heartfelt gratitude to all the faculty members, friends and staff of Computer Science and Engineering Department of Assam Engineering College and also to Dr. Sudha Sadasivam G. and Prof. Vani Kandhasamy of PSG College of Technology for their tremendous help and support for the completion of this project.

We would like to convey our sincerest gratitude to our principal, Dr. Atul Bora, Assam Engineering College and Prof. Dinesh Shankar Pegu, Head of the Department of Computer Science and Engineering, Assam Engineering College for giving us the opportunity to work under them and lending immense support for the completion of this report.

We would like to thank our family members for their constant support, encouragement and good wishes, without which working on this project would have been difficult.

Lastly, we would like to thank all the people who have been related directly or indirectly in the process of preparation of this project.

**ABSTRACT**

During the last few decades the recommender system has gained more and more popularity with the advent of various web services such as Amazon, Netflix etc. In this era of personalization, every user wants personalized content to be displayed on their phones/laptops. We come in contact with the likes of recommendation systems in our day to day without even realising it. In Layman’s term Recommendation system is an algorithm which recommends relevant products to users. In our report we have built a recommender system using three different ways to check which is the most accurate. The three models that we have used are GMF, MLP, NeuMF.

Matrix Factorisation is one of the most popular models for building a recommender system. It was mainly popularised during the Netflix prize challenge due to its effectiveness. It represents the user/item as a vector of latent features which are projected into a shared feature space. Although we only used the general/vanilla matrix factorisation in our model, there are many other ways of calculating the matrix factorisation like SVD, ALS, WALS. We also explored the MLP to build a recommender system on the basis of a DNN Model. A multi-layer perceptron is a feed-forward neural network with multiple hidden layers between the input layer and the output layer. MLPs can be interpreted as stacked layers of nonlinear transformations, learning hierarchical feature representations. MLP is capable of estimating the intricate interactions between users and items. And the third model that we used is termed as NeuMF. It is nothing but a model created by the amalgamation of both GMF and MLP. This model replaces the simple dot product of the GMF with the flexibility and non-linearity of the neural networks that is MLP. Thus , giving us a model which has a very little loss value and is efficient and accurate enough.

**CONTENTS**

**Page No.**

**Certificate**  i

**Candidates’ Declaration**  ii

**Acknowledgement** iii iii

**Abstract**  iv

**Contents**  v-vi

**List of Figures**  vii

**List of Tables**  viii

**List of Abbreviations and Symbols Used** ix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Chapter 1:**  **INTRODUCTION** | |  | 1-6 |  |  |
| 1.1: | Recommender System   * + 1. About     2. Types |  | 1-2 |  |  |
| 1.2: | Methods  1.2.1 Matrix Factorisation  1.2.2 Multi-Layer Perceptron  1.2.2.1 MLP  1.2.2.2 ANN  1.2.2.3 DNN |  | 2-6 |  |  |
| 1.3: | 1.3 NDCG |  | 6 |  |  |
| **Chapter 2 :**  **LITERATURE REVIEW** | |  | 7-8 |  |  |
| **Chapter 3:**  **IMPLEMENTATION AND RESULTS** | |  | 9-18 |  |  |
| 3.1 Dataset | | 9 |
| 3.2 Packages and Libraries | | 9-11 |
| 3.3 Implementation  3.3.1 NDCG  3.3.2 MF  3.3.3 MLP  3.3.4 NeuMF | | 11-13 |
| 3.4 Results  3.4.1 MF Model  3.4.2 MLP Model  3.4.3 NeuMF Model | | 13-19 |
| **Chapter 4:**  **CONCLUSION AND FUTURE SCOPE** | | 19 |
| 4.1: | Conclusion |  | 19 |  |  |
| 4.2: | Future Scope |  | 19 |  |  |
|  | **ANNEXURE A** |  | 20 |  |  |
|  | **ANNEXURE B** |  | 21-22 |  |  |
|  | **ANNEXURE C** |  | 23-25 |  |  |
|  | **ANNEXURE D** |  | 26-28 |  |  |
|  | **REFERENCES** |  | 29 |  |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No:** | **Name of Figure** | **Page No.** |
| 3.1: | Training Error vs Epoch for ‘Adam’ optimiser in the GMF model | 13 |
| 3.2: | GMF model | 14 |
| 3.3: | MLP model | 16 |
| 3.4: | Training Error vs Epoch for ‘Nadam’ optimiser in the NeuMF model | 17 |
| 3.5: | NeuMF model | 18 |
|  |  |  |
|  |  |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No:** | **Name of Table** | **Page No.** |
| 3.1: | To select the best optimiser for the GMF model | 13 |
| 3.2: | To select the best optimiser for the MLP model | 14 |
| 3.3: | To select the best optimiser for the NeuMF model | 17 |

**LIST OF ABBREVIATIONS AND SYMBOLS USED**

|  |  |
| --- | --- |
| ALS | Alternating Least Square |
| ANN | Artificial Neural network |
| CNN | Convolutional Neural Network |
| DNN | Deep Neural Network |
| GMF | General Matrix Factorization |
| ML | Machine Learning |
| MLP | Multi-layer Perceptron |
| MSE | Mean Squared error |
| NDCG | Normalized Discounted Cumulative Gain |
| NeuMF | Neural Matrix Factorisation |
| ReLU | Rectified Linear Unit |
| RNN | Recurrent Neural Network |
| SVD | Single Value Decomposition |
| WALS  ∑ | Weighted Alternating Least Square  Summation |

**CHAPTER 1 – INTRODUCTION**

**SECTION 1.1 – RECOMMENDER SYSTEM**

**1.1.1 ABOUT**

In this era of modernization and especially amidst this pandemic, every other person is dependent on the digital world. From their basic requirements to their desired needs, everyone is using the digital platform for their satisfactory living. But have you ever wondered how these online platforms are so impressive in the sense that they suggest us with items and products akin to our preference and choice? How it makes easier to prospect many more different but similar kind of items for us? It is because; these platforms use the recommendation system. With the rise in online services, there has been a motivation to create systems that can predict user’s preference of items leading to a better user experience.

But what is a recommendation system? Primarily used in commercial applications, a recommendation system is a system that recommends the user certain item based on users’ taste, which rely on learning an appropriated embedding representation of the queries and items. It is a section that filters the information and seeks to predict the "rating" or "preference" a user would give to an item, and helps users to find homogeneous and compelling content in a large catalog of items. These recommendations have not only made it simpler for us to search for similar kind of products but have also popularized various unexplored bunch of products in the internet.

Do you know how the recommendation system works? Every social platform contains text files with small piece of data files called cookies which cookies contain information about the user’s activities on the internet and based upon these data the system recommends different items to the user on the online platforms. This is the basic working structure of a recommendation system.

However, a recommendation system also display items that users might not have thought to search for their own. This is because there are different types of recommendation systems which consider various parameters for their evaluation.

**1.1.2 TYPES**

As stated above, recommendation system is classified into different types depending on the user’s personal preference and the interests of similar users. Candidate generation being the first stage of recommendation uses one of the two approaches to recommend different items to the user.

* Content – Based Filtering
* Collaborative Filtering

1. **Content Based Filtering –** Content-based filtering is the system that recommends items similar to what the user likes, based on their previous actions and explicit feedback, i.e. content based filtering uses the item feature to suggest items to the user which. The main idea behind this is to create “Item” and “User” profiles which collectively matches with the catalog and accordingly recommends items to the users.

Content based filtering is advantageous as the model captures the specific interest of a user for recommending items, and scales to a large number of users as it does not need any data about other users. But in case of new users, the model fails to suggest items to the user due to absence of any previously existing data, leading to limited ability of the method to expand the users’ existing interest.

1. **Collaborative Filtering –** Collaborative filtering is a powerful method that overcomes the constraints of Content-based filtering and is more accurate than content based filtering approach. Collaborative Filtering takes into consideration the likes and dislikes of similar users along with the approaches of content based filtering method. Collaborative filtering allows for serendipitous recommendations as it uses similarities between users and items simultaneously to provide recommendations. Collaborative filtering is advantageous as it automatically learns the embedding and doesn’t require any contextual feature. Collaborative-filtering systems focus on the relationship between users and items. The similarity of items is determined by the similarity of the ratings of those items by the users who have rated both items.

There are two parts of collaborative filtering –

1. User based : It measures the similarity of interests between different users
2. Item based : It measures the similarity between the items that the target users rate or interact and other items.

**SECTION 1.2 – METHODS**

Following are the different ways to implement a recommender system with collaborative filtering as the basis of recommendation –

1. Matrix Factorisation
2. Multi-Layer Perceptron
3. NeuMF

**1.2.1 GENERAL MATRIX FACTORIZATION**

One of the widely used methods to implement a recommendation system is General Matrix Factorization which is a simple embedding model. It is a class of collaborative filtering algorithm which is very effective and efficient. The basic idea behind general matrix factorization model is to predict the output from the utility matrix and recommend items to user accordingly.

Initially, a rating matrix in the matrix factorization approach contains both implicit and explicit feedback (rating) record of different users for different products, each row representing the unique users and each column corresponding to different items that are being rated. The rating matrix is usually a sparse matrix because of the absence of every user rating all the items. Matrix Factorization decomposes the matrix into its constituent elements which makes complex computational matrix problems easier.

In this situation, a matrix R (u × i), which is a user is divided into two matrix Q (u × k) and P (k × i). The matrix Q is a user-concept matrix and P is an item-concept matrix. To compute the rating of a user for an item we take the dot product of two matrices-

**rui = quT . pi**

The loss function regarding the accuracy –

**L =**

where,

rui : rating associated with user u, item i

qu : user vector

pi : item vector

: regularization parameter

The term on the right is the regularization term; added to forbid the decomposed matrix x and y to over-fit to the original matrix.

In order to go through with matrix factorization, we have studied the working and basics of three well-known algorithms. Those are –

1. Single Value Decomposition
2. Alternating Least Square
3. Weighted Alternating Least Square

**SVD –** In SVD, a complex matrix M( m × n) is decomposed into three smaller matrix U (m × r),∑( r × r),V\* (r × n)where U and V are orthonormal matrices, ∑ is a diagonal decreasing matrix and r is the rank of the matrix M. It is used to decrease the dimensionality of the matrix and also to deal with the sparsity of the user-item matrix M.

**ALS method** – ALS method or Alternating Least Square method of MF is one of the iterative methods in order to find two matrices X and Y that best approximates the rating matrix R. In this method, the loss function is minimized with respect to either the row or column factor while keeping the other factor as a constant.

**WALS method** – In this method, weights are introduced in the loss function. These weights vary for the zero and the non-zero entries of the matrix. The weights are generally calculated by the sum of the non-zero entries of a row in order to normalize the entries of users who rated a different number of items. This method is usually more efficient than ALS method.

**1.2.2 MULTI-LAYER PERCEPTRON**

**1.2.2.1 MLP**

A perceptron is the basic unit powering deep learning. A perceptron can also be realized as an artificial neuron, which has the ability to learn and solve complex problems. It is basically a binary classification algorithm, that takes inputs and produces either a 1(yes) or 0(no) as their output.

The combination of such several perceptron stacked in different layers is known as Multi-Layer Perceptron or MLP. The union of these artificial neurons forms the Artificial Neural Network, a complex system which is mastered by training to solve tough problems. MLP takes real inputs which are then multiplied by weights and are fed into the activation function after adding the bias factor, ultimately producing a classification decision as output. Neural network algorithms learn by discovering better and better weights that result in a more accurate prediction, as weights allow the perceptron to evaluate the relative importance of each of the outputs. Back propagation is one such supervised learning algorithm used to select the correct weight, which performs iterative back passes in attempt to minimize the loss between the correct and actual model prediction. The perceptrons uses different weights for each signal going from one perceptron of the input to the other of the hidden layer, and further to the output layer.

**1.2.2.2 ANN**

A neuron of the human brain has the capability of taking messages as input through dendrites, doing some computations and producing output through axons. There are millions of neurons in the human brain which together constitute the neural network.

Similarly, Artificial Neural Network was developed as simulating neurons or networks of neurons. In the computer world, we use a simple model of human brain neurons to implement a neuron as a logistic unit for an ANN. It was developed for computers to mimic the human brain. Just like the human brain does most of the work unconsciously, a computer is trained to function like a human brain using ANN, where the neuron takes input in the input layer, does some computations (in the hidden layer) and accordingly produces output through the output layer. Neural Networks are an effective study-art technique for modern day ML applications.

ANN is a broad topic that covers any form of neural network learning model, such as CNN, RNN, DNN etc. ANN can be classified as Shallow or Deep depending on the number of layers that are present in between the input and output layers. A three layer MLP is called a Non-deep or Shallow neural Network whiles a four or more layered MLP is called a Deep Neural Network.

**1.2.2.3 DNN**

A DNN is an ANN which is capable of doing complicated calculations due to the presence of multiple hidden layers. The DNN finds the correct mathematical manipulation to turn the input into the output, be it either a linear or a non-linear relationship. The network moves through the layers calculating the probability of each output. DNN is more efficient than Shallow Neural Network because the decision functions uses activation function other than step functions, resulting in output of real values (usually between 0 and 1, or between -1 and 1).

**1.2.3 NEURAL MATRIX FACTORISATION**

DNN is an effective method for preparing recommending system model. However, the exploration of deep neural networks on recommender systems has received relatively less scrutiny. NeuMF is a more preferred approach for implementing recommendation systems as it supports flexibility and non-linearity of neural networks, and aims at enhancing the model expressiveness.

NeuMF concatenates two sub-networks including GMF and MLP. The GMF is a generic neural network version of matrix factorization which takes the product of user vector and item vector as inputs. MLP being the other component of NeuMF, uses the user and item embeddings as input. With the complicated connections and nonlinear transformations, it is capable of estimating the intricate relations between users and items. NeuMF concatenates the outputs of the two components, GMF and MLP and projects it with an activation function.

**SECTION 1.3 – NDCG**

Normalized Discounted Cumulative Gain is a measure of ranking quality. It measures how the product is ranked on the basis of its relevance.

Calculating NDCG is a three step process –

1. First we need to calculate the cumulative gain (CG). Cumulative Gain is the sum of all the relevance scores in a recommendation set

CG = ∑ relevancei

1. Second we need to calculate DCG. DCG makes sure that the most relevant product is shown first.

DCG = ∑ (2relevancei-1) / log2 (i+1)

1. Lastly, we need to calculate the NDCG. Normalisation is required for DCG in order for it to have a proper upper and lower range so that it doesn't vary for every user.

In order to calculate the final NDCG we need to calculate –

1. DCG of the recommended order
2. DCG of the ideal order (iDCG).

NDCG= DCG/iDCG

**CHAPTER 2 – LITERATURE REVIEW**

Recommender system is an important and popular field in this era of information and personalization, where people want to get recommendations of items similar to their choice. Recommender systems have become an important research field since the emergence of the first paper on collaborative filtering in the mid-1990s. Although academic research on recommender systems has increased significantly over the past 10 years, there are deficiencies in the comprehensive literature review and classification of that research.

**The Netflix Prize**

The recommendation system became big mainly after the Netflix prize challenge.

The **Netflix Prize** was an open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films, i.e. without the users or the films being identified except by numbers assigned for the **contest**.

In October, 2006 Netflix organized an open challenge to propose an algorithm which was 10% better than their current algorithm, i.e. CineMatch, keeping prize money of total $1m.

Netflix published a dataset with approximately 17,000 moves, 500,000 users and 100 million ratings (training set of 99 million ratings). The Root-mean-square-error (RMSE) was used to measure the performance of algorithms. CineMatch had an RMSE of 0.9525.

The winner “Bellkor’s Pragmatic Chaos” completed the challenge almost after 3 years, having improved the algorithm by almost 10.6%. This is how the recommendation grew in popularity.

Tapestry was the first recommender system which was designed to recommend documents for newsgroups. The term “collaborative filtering” was also introduced by the authors as they used social collaboration to help users with large volume of documents.

# The lines below were taken from an excerpt from a research paper on ‘A Literature Review and Classification of Recommender Systems on Academic Journals’ by Deuk Hee Park, Hyea-Kyeong Kim, II-Young Choi and Jae Kyeong Kim.

In between 2001 to 2010, there were about 37 journals published on recommender system which were searched based on "Recommender system", "Recommendation system", "Personalization system", "Collaborative filtering" and "Contents filtering". These 37 journals were selected from top 125 journals of the MIS Journal Rankings.

187 articles were reviewed and classified into eight recommendation fields (book, document, image, movie, music, shopping, TV program, and others) and eight data mining techniques (association rule, clustering, decision tree, k-nearest neighbor, link analysis, neural network, regression, and other heuristic methods). The results found had quite a significant impact. First, based on previous publication, it could be inferred that the interest in the recommender system related research would develop significantly in the future. Second, 49 articles were related to movie recommendation whereas image and TV program recommendation were identified in only 6 articles. As these results meant that the MovieLens data set was in use mostly, so it became important to prepare data sets of other fields. Third, it has been observed that social network analysis has been implemented in various applications.

**CHAPTER 3 – IMPLEMENTATION AND RESULTS**

**SECTION 3.1 – DATASET**

The dataset rating.csv includes the rating information of movies. It includes five columns which are :

* userid
* productid
* categoryid
* rating
* helpfulness

The figures provide information about a user rating a specific product of a particular category and the helpfulness of the rating to recommend further items to the user. The model is trained with the given stats of userid, productid and rating so as to build a proper recommender system.

**SECTION 3.2 – PACKAGES AND LIBRARIES**

* “Surprise” is the main package which contains a set of built-in algorithms and datasets. It is a Python scikit for building and analyzing recommender system that deals with rating data explicitly.
* “Pandas” library is used for reading the dataset, manipulating and analyzing it.
* “SVD” package is used for Matrix Factorization.
* Functions mse() and mae() are used to calculate accuracy of the model.
* Different libraries used in the code are –

1. Pandas
2. Tensorflow
3. Numpy
4. Sklearn
5. Math
6. Keras

* keras.layers

Different layers of keras used –

Embedding – It is the first hidden layer of a network and is used for neural network on text data. It is **used when we want to create the embeddings to embed higher dimensional data into lower dimensional vector space.**

Flatten – It is used to flatten the input, without affecting the batch size. Flattening a tensor means to remove all of the dimensions except for one. The flatten operation reshapes the tensor to the shape which is equal to the number of elements contained in tensor.

Dot – The dot layer is the layer that computes a dot product between samples in two tensors.

Dense – It is one of the most commonly and frequently used layers of Keras. Dense layer is the regular deeply connected neural network layer.

* keras.optimisers
* keras.utils
* keras.regularizers
* keras.initializers
* keras.model
* Optimizers used in the code while trial and error –

1. SGD – SGD is an iterative method for optimizing an objective function with suitable smoothness properties. It picks single data from the dataset randomly per iteration so as to reduce the computations enormously.
2. RMSprop – Root Mean Square Propagation is a gradient based optimization technique designed for training artificial neural networks.
3. Adagrad – Adagrad is effectively SGD with a per-node learning rate scheduler built into the algorithm thus improving SGD by giving weights historically accurate leaning rates.
4. Adadelta – Adadelta optimization is a stochastic gradient descent method that is based on adaptive learning rate per dimension. It is basically a robust extension of Adagrad.
5. Adam – Adaptive Moment Estimation is a combination of Adagrad and RMSprop, which is used to calculate the individual adaptive learning rate for each parameter from estimates of first and second moments of the gradients.
6. AdaMax – It a variant of Adam based on the infinity room. AdaMax provides an important advantage of being much less sensitive to the choice of the hyper-parameters.
7. Nadam – Nesterov – accelerated Adaptive Moment Estimation is a combination of NAG and Adam. It is employed for noisy or high curvature gradients. The learning process is accelerated by summing up the exponential decay of the moving averages for the previous and current gradient.
8. Ftrl – Follow the Regularized Leader optimizer is a per-coordinate learning rate system.

* A Metric function is used to judge the performance of a model. Functions mse() and mae() are Regression Metrics which used for loss determination or to calculate accuracy of the model.

**SECTION 3.3 – IMPLEMENTATION**

**3.3.1 NDCG**

* After training the model with SVD, we take the "top n" values from the predictions object which is a key-value pair with the uid as "keys" and an array of recommendation items as the "value" and sort each array in decreasing order of their predicted rating. In each of the array we take a cumulated sum of the ratio between the "ground truth" and "predicted truth" of each item in the corresponding array to get the individual discounted cumulative gain for each "uid".
* We then normalize this DCG value by taking the average of the sum of the individual DCG values to finally get the Normalised Discounted Cumulative Gain or NDGC value.

**3.3.2 MATRIX FACTORISATION**

* A GMF takes the concatenation of two vectors.
* The user and item columns of the dataset are embedded into their corresponding user vector (u\_id) and product vector (p\_id) respectively, considering latent factor = 20.
* The product of the vectors produces a low-rank utility matrix, which is the approximation of the original matrix (user column × product column of the dataset).
* The dot product of the two vectors gives a real number which represents the ratings of user u for item i.

**rui = quT . pi**

* A loss function measures the accuracy of the approximation and the regularization term helps prevent overfitting.
  + 1. **MULTI-LAYER PERCEPTRON**
* MLP is a neural network which consists of different layers, labeled as input, hidden and output layers, where the number of hidden layers differs from model-to-model depending upon the complexity of the model.
* MLP concatenates the user (u\_id) and item (p\_id) embeddings using the *keras.layers.concatenate()* function and feeds it as the input into the input layer.
* Each layer has an attached weight which can be calculated through either linear or non-linear activation functions such as softmax, ReLU etc. The softmax layer maps a vector of ratings to a probability distribution. To compare the output of softmax layer, a loss function is defined which also represents the items user has interacted with. This can be represented as a normalized multi-hot distribution (a probability vector). A Batch Normalization function is used to normalize the nodes value in the layers. Also, to prevent over-fitting of the matrix, Dropout function is used that drops off some percentage of random values.
* The hidden layers take into consideration a particular number of latent factors (features) and perform complex calculations accordingly for the next neighboring layers. The hidden layer loop runs from n (number of hidden layers) to 0 numbers of times.
* The final or output layer produces a modified version of rating matrix which is fed into the training and test model for making effective predictions.
* The keras.getlayer() function is used to get the embedded layer .
  + 1. **NEURAL MATRIX FACTORISATION**
* A NeuMF is a fusioned model of two subnetworks, GMF and MLP which forms the recommendation system.
* A GMF takes the concatenation of two vectors and produces a utility matrix. The user and item columns of the dataset are embedded into u\_id and p\_id vectors respectively, considering latent factor = 20. The prediction value is obtained by applying optimiser to the product of output matrix and weights.
* MLP feeds the concatenation of u\_id and p\_id into the input layer. The MLP is implemented by considering 6 hidden layers. Also, *relu* is added as the non-linear activation function, along with *kernel-initializer = ‘he\_uniform’*. The hidden layer is responsible for estimating the complicated interactions between the users and items. The output layer uses *softmax* as the activation function and *kernel-initializer = ‘lecun\_uniform’* to produce a modified matrix.
* To fuse the results of GMF and MLP, NeuMF concatenates the second last layers of the two subnetworks to create a feature vector which can be passed to the further layers. Afterwards, the outputs are projected with weight matrix and an activation function.

**SECTION 3.4 – FINDINGS AND RESULTS**

**3.4.1 GMF MODEL**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Optimizer | Loss at start | Loss at end | Mae at start | Mae at end | Time for each epoch(approx.) |
| Adagrad | 18.4499 | 18.4477 | 4.1624 | 4.1622 | 6s |
| Adam | 18.0541 | 0.3843 | 4.1090 | 0.4550 | 155s |
| RMSprop | 18.4485 | 11.6930 | 4.1623 | 3.0608 | 96s |
| Adadelta | 18.4498 | 18.4496 | 4.1624 | 4.1624 | 7s |
| Adamax | 18.4499 | 18.3033 | 4.1624 | 4.1446 | 57s |
| Nadam | 18.4242 | 7.8622 | 4.1592 | 2.4254 | 146s |
| Ftrl | 18.4499 | 18.4499 | 4.1624 | 4.1624 | 7s |
| SGD | 18.4499 | 18.4423 | 4.1624 | 4.1624 | 6s |

Table 3.1 : To select the best optimiser for the GMF model

From the above table it is inferred that Adam is the best optimiser as it greatly minimises the loss. The figure below shows the rate of minimization of the losses with the increasing number of epochs.

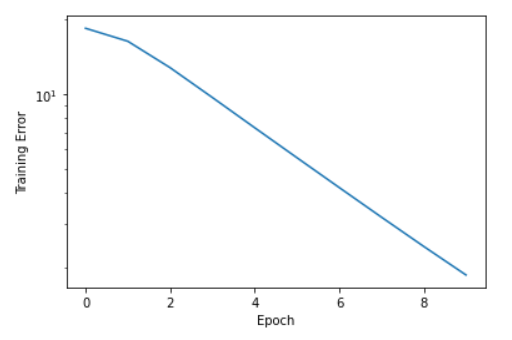


Fig 3.1 – Training Error vs Epoch for ‘Adam’ optimiser in the GMF model

The loss seen after evaluating the test data with our model comes out to be 7.0950.

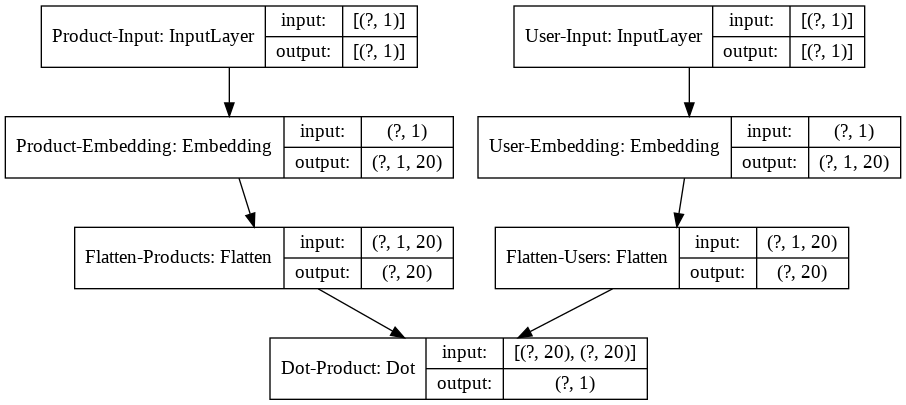


Fig 3.2 – GMF model

**3.4.2 MLP MODEL**

Latent Factors = 20

Total number of epochs = 10

Total number of layers in MLP = 6

Activation = ReLU (for the inner layers) and Softmax (for the final layer)

Kernel Initializer = lecun\_uniform (for the final layer) and he\_uniform (for the rest of the layers)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Optimizer | Loss at start | Loss at end | Mae at start | Mae at end | Time for each epoch(approx.) |
| Adagrad | 11.1252 | 11.1252 | 3.1627 | 3.1627 | 237.6s |
| Adam | 11.1250 | 11.1250 | 3.1627 | 3.1627 | 244.5s |
| RMSprop | 11.1250 | 11.1250 | 3.1627 | 3.1627 | 358.3s |
| Adadelta | 11.1252 | 11.1252 | 3.1627 | 3.1627 | 258.9s |
| Adamax | 11.1250 | 11.1250 | 3.1627 | 3.1627 | 228.8s |
| Nadam | 11.1250 | 11.1250 | 3.1627 | 3.1627 | 406.7s |
| Ftrl | 11.1250 | 11.1250 | 3.1627 | 3.1627 | 281.2s |
| SGD | 11.1252 | 11.1252 | 3.1627 | 3.1627 | 218.7s |

Table 3.2 : To select the best optimiser for the MLP model

Here, we see that the loss remains the same for each optimiser. So we choose the best optimiser in terms of the least time taken for each epoch for a particular optimiser i.e. SGD.

The loss after evaluating the test data with our model comes out to be 11.0873.

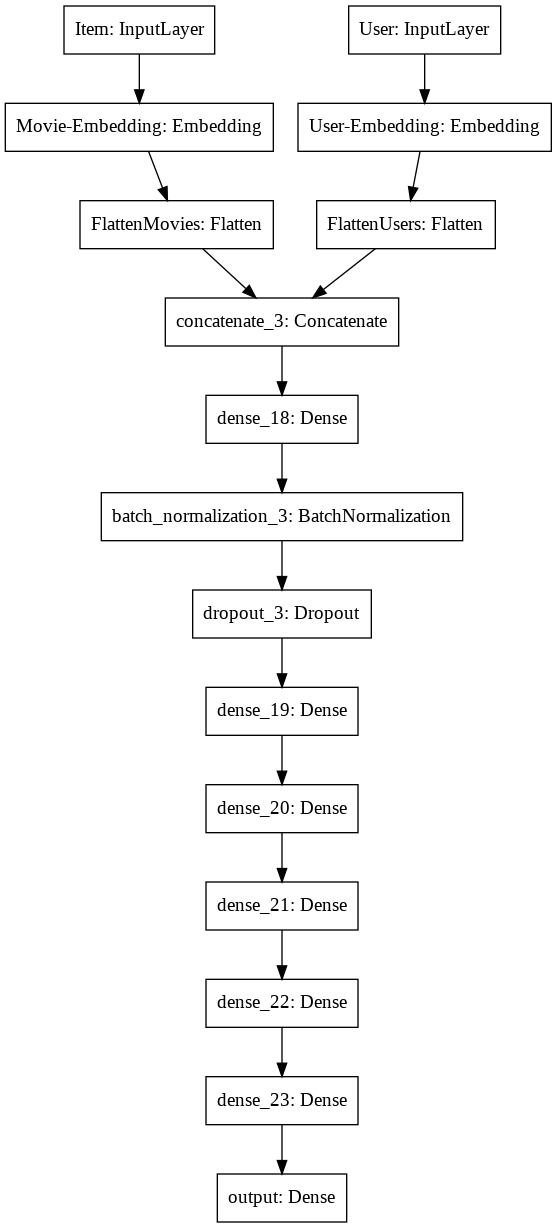


Fig 3.3– MLP model

**3.4.3 NeuMF MODEL**

Latent Factors = 20

Total number of epochs = 10

Total number of layers in MLP = 6

Activation = ReLU (for the inner layers) and Softmax (for the final layer)

Kernel Initializer = lecun\_uniform (for the final layer) and he\_uniform (for the rest of the layers)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Optimizer | Loss at start | Loss at end | Mae at start | Mae at end | Time for each epoch(approx.) |
| Adagrad | 1.2364 | 1.1253 | 0.8528 | 0.8203 | 263.8s |
| Adam | 1.1589 | 0.0972 | 0.8280 | 0.2371 | 456.2s |
| RMSprop | 1.1551 | 0.2127 | 0.8289 | 0.2975 | 438.6s |
| Adadelta | 7.7832 | 1.1253 | 2.3172 | 0.8203 | 259.8s |
| Adamax | 1.1505 | 0.8543 | 0.8259 | 0.6916 | 278.8s |
| Nadam | 1.1562 | 0.0792 | 0.8276 | 0.2006 | 475.1s |
| Ftrl | 1.4667 | 1.1244 | 0.9079 | 0.8203 | 278.7s |
| SGD | 1.1427 | 1.1286 | 0.8239 | 0.8202 | 193.7s |

Table 3.3 : To select the best optimiser for the NeuMF model

Here, we can see that Nadam optimiser gives the best results as it minimizes the loss up to 0.0792 after the 10th epoch.

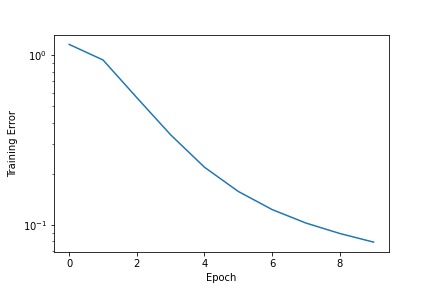


Fig 3.4 – Training Error vs Epoch for ‘Nadam’ optimiser in the NeuMF model

The loss after evaluating the test data with our model comes out to be 1.5200.

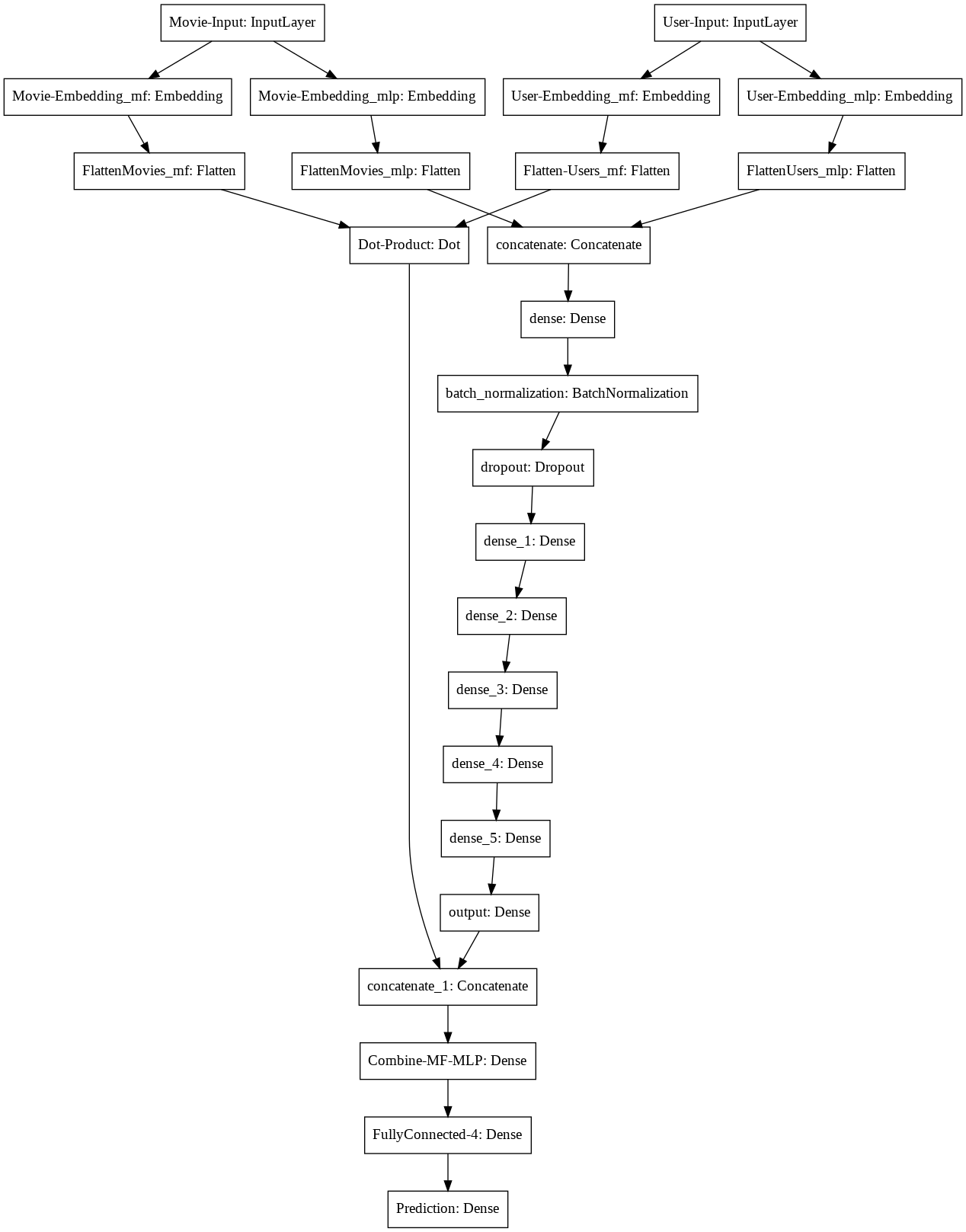


Fig 3.5 – NeuMF model

**CHAPTER 4 – CONCLUSION AND FUTURE SCOPE**

**SECTION 4.1 – CONCLUSION**

Through the project, we have researched extensively about recommendation systems and also built codes on the same. We have implemented the system using three different models. We found that in the General Matrix Factorization (GMF) model, the loss on the test set was found to be around 7.08. Also, for the Multi-Layer Perceptron model, the loss was found to be 11.0873. Here we have inferred that a Deep Learning model like the MLP is not much efficient when it comes to recommendation systems. So, we combined both the GMF and the MLP models to form the NeuMF model. For this model, we found that the losses in terms of mean squared error and mean absolute error decreased drastically i.e., the loss was found to be 1.5200. We ultimately came to the conclusion that NeuMF model is a comparatively much better model which can help us to predict the output more accurately. In the end, with the help of this model, we found five product recommendations for a particular user.

**SECTION 4.2 – FUTURE WORK**

The goal to achieve a better recommendation system still remains. There are various other methods like node2vec, establishing a better neural network and also making better hybrid model with the help of which we can increase the accuracy of the predictions and improve the overall efficiency of the code. Apart from this, we can also find better values for the hyperparameters such as the number of latent factors, number of epochs, activation etc., which would help us in achieving a more precise model. Also, the code can be better presented with the help of GUI or in website in order to visualise the working of the models. Further, better and more cost-efficient metrics could be implemented to check the overall preciseness of our model.

Eventually, the code can even be deployed with other websites to calculate recommendations for various products.

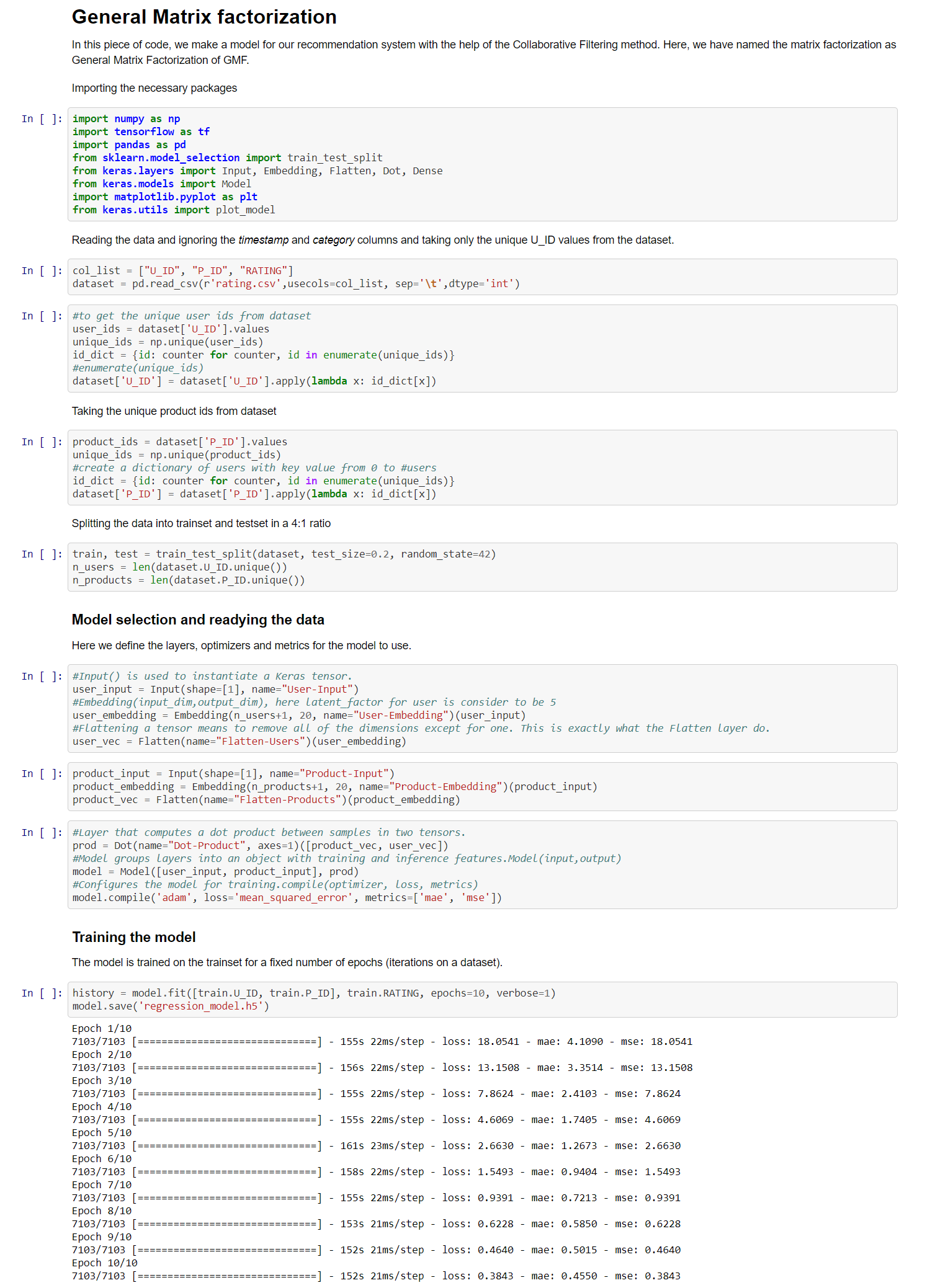
**ANNEXURE A**

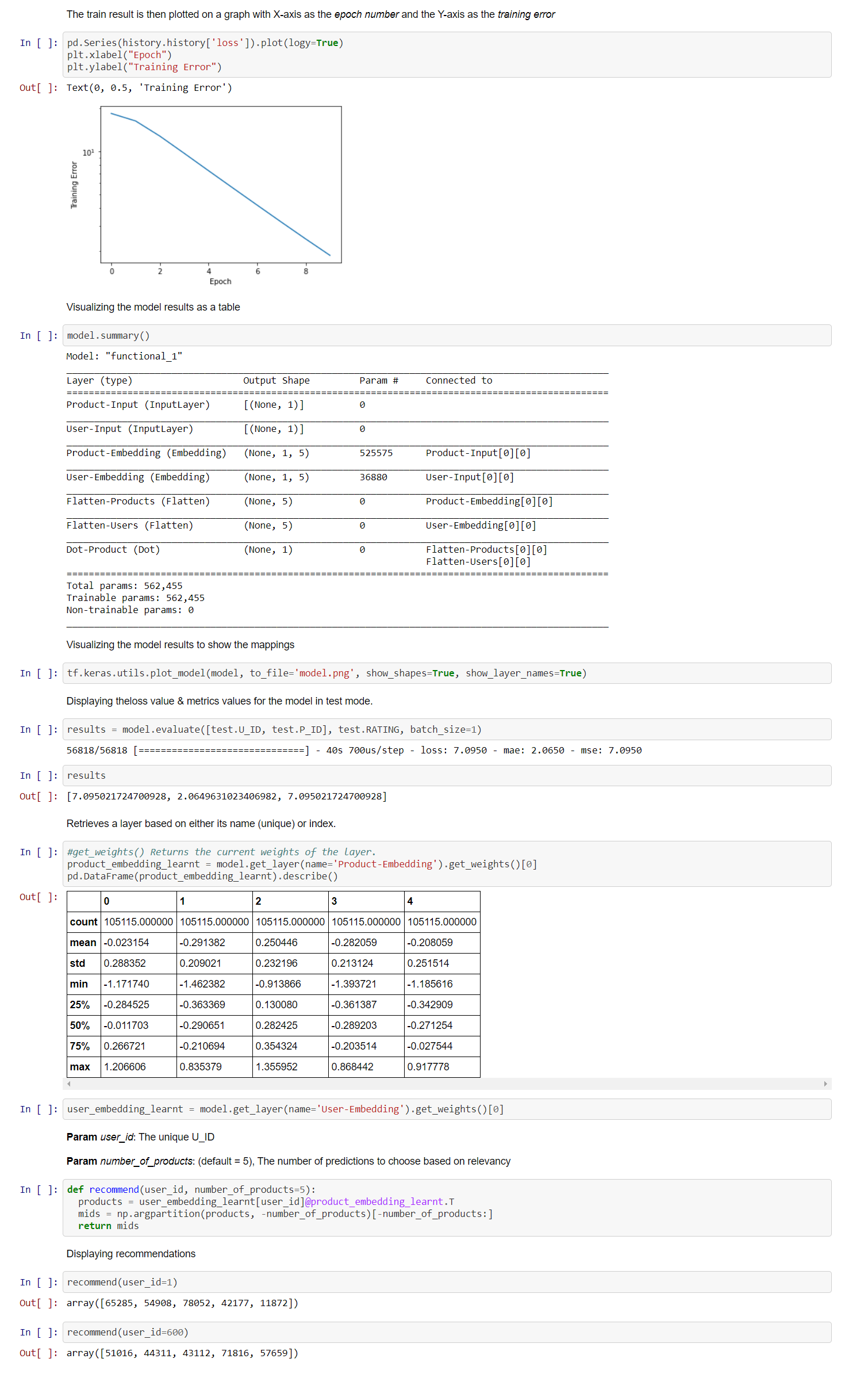
## **SVD\_model\_and\_NDCG**

****

**ANNEXURE B**

GMF Model

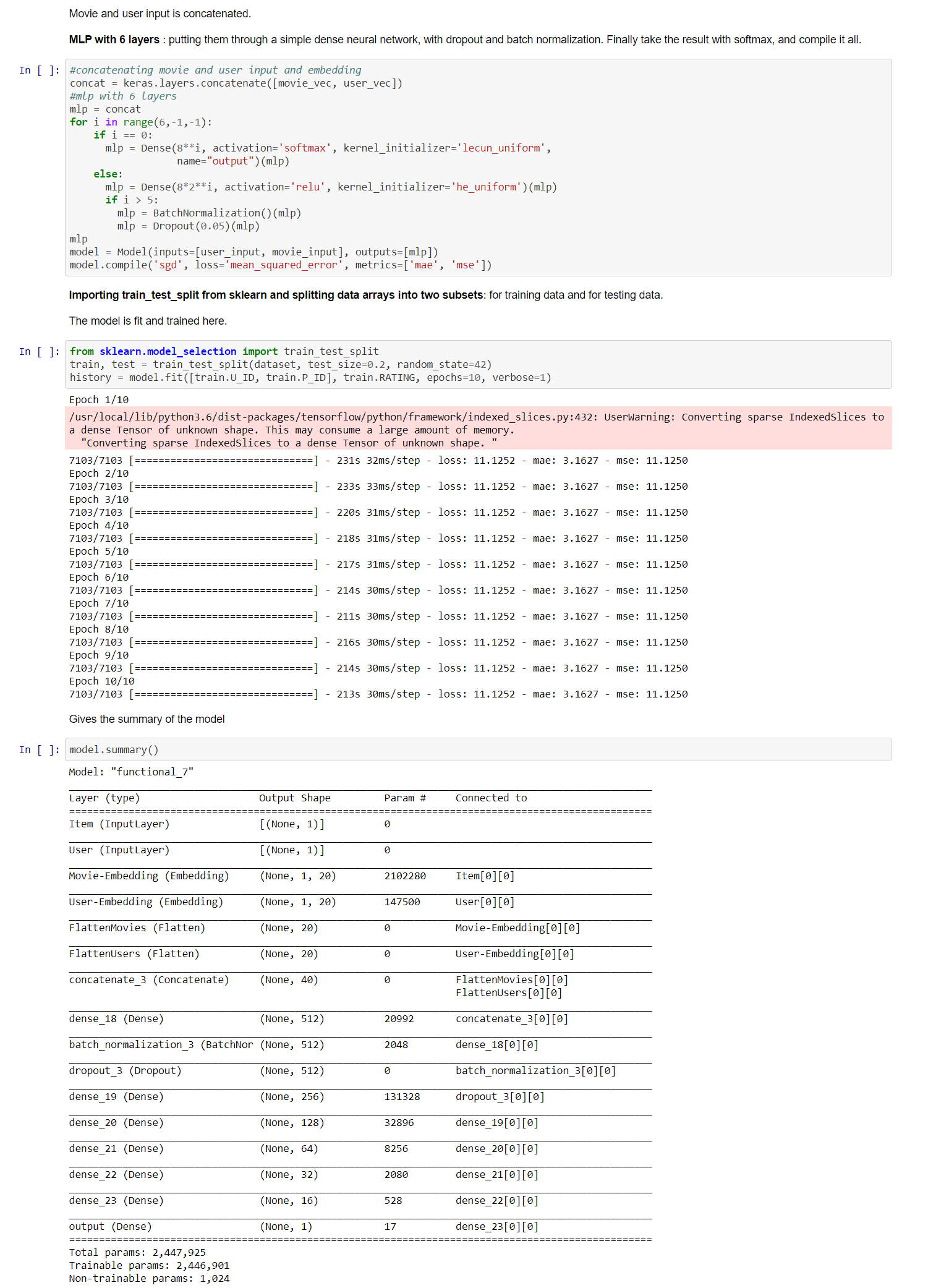
****

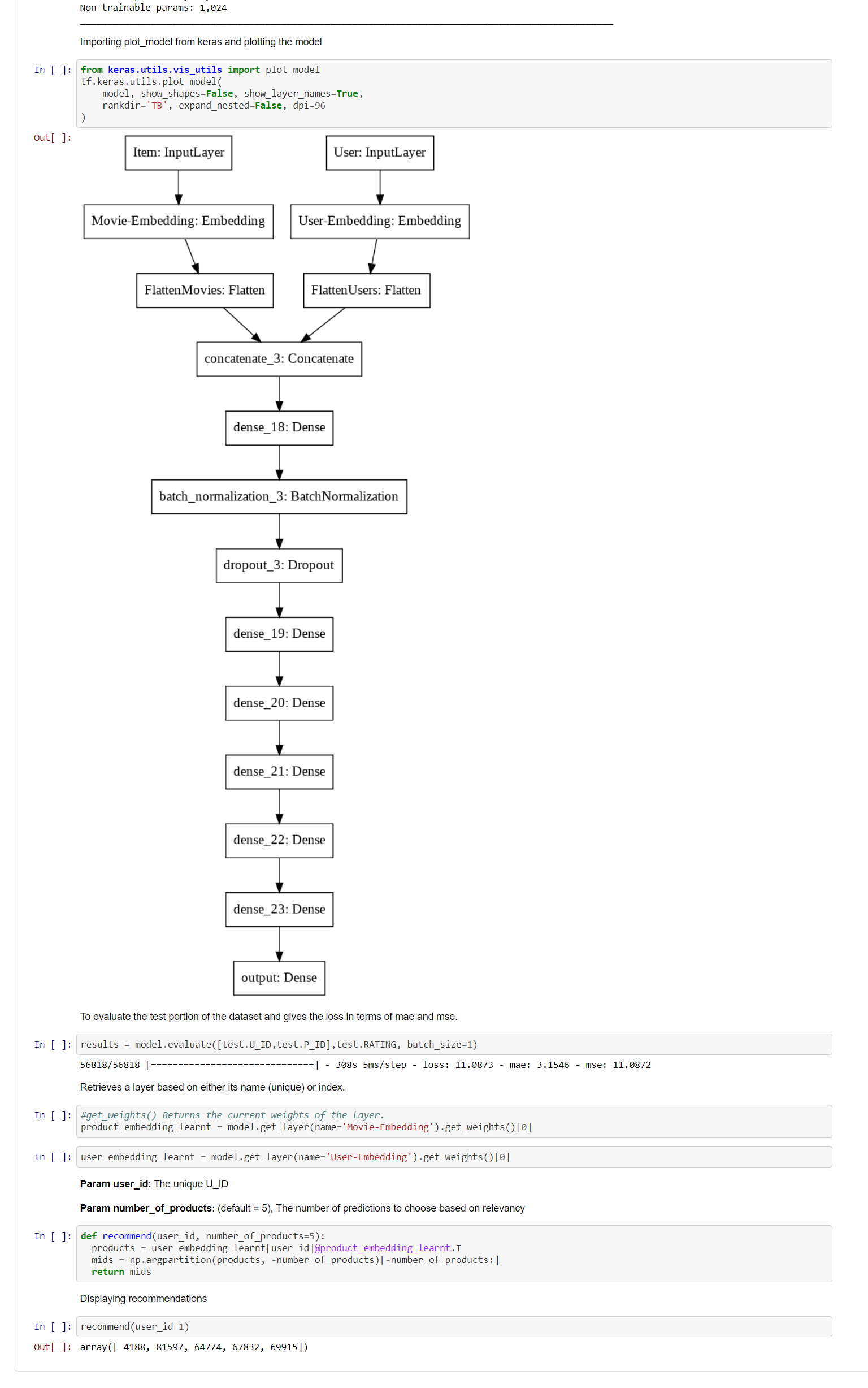
****

**ANNEXURE C**

MLP Model

****

****

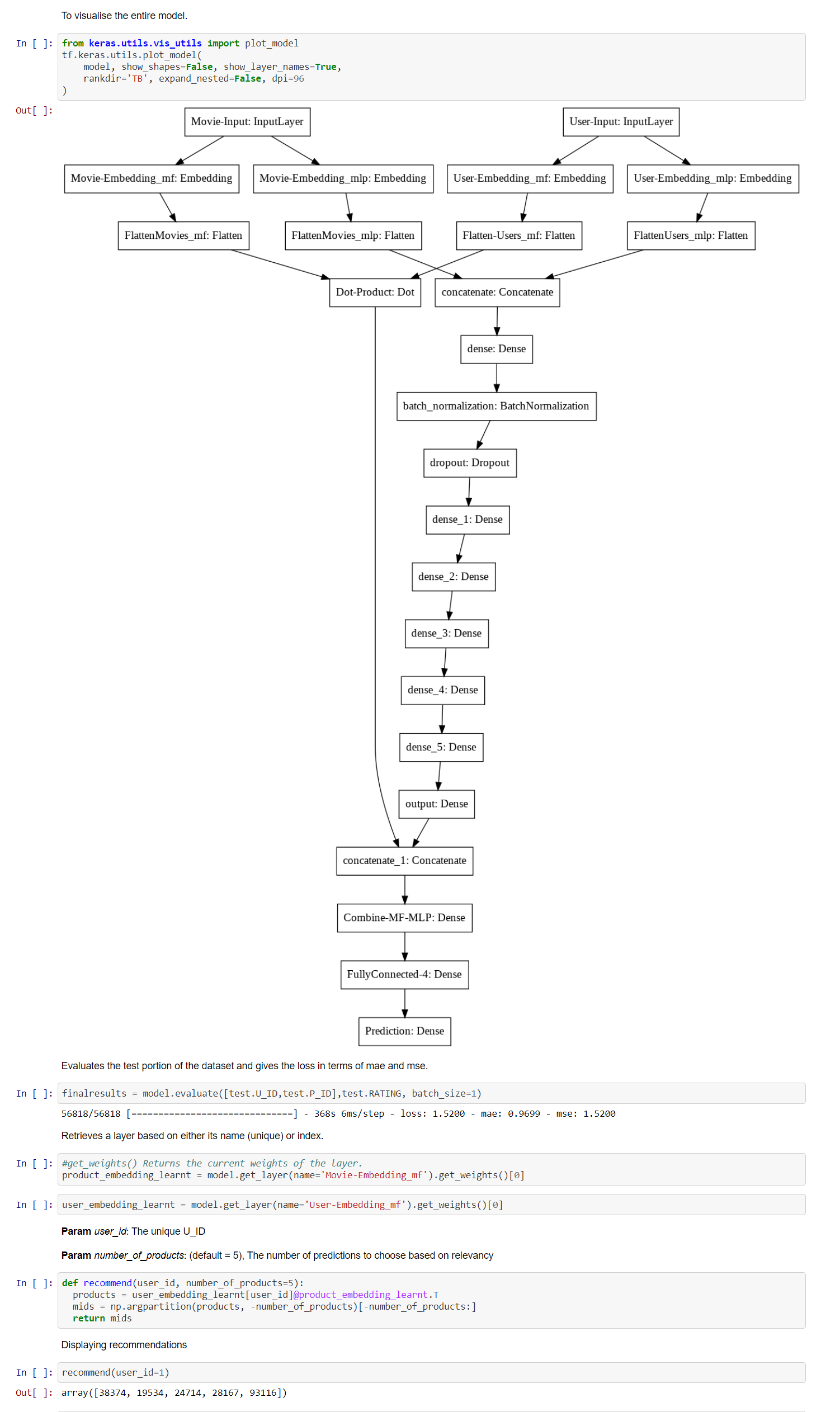
****

**ANNEXURE D**

NeuMF Model



****

****

Github repository - https://github.com/rahul-thapa/PSG\_Research\_Internship

**REFERENCES**

|  |
| --- |
|  |

<https://builtin.com/data-science/collaborative-filtering-recommender-system>

<https://developers.google.com/machine-learning/recommendation/overview>

<https://www.youtube.com/playlist?list=PLLssT5z_DsK9JDLcT8T62VtzwyW9LNepV>

<https://cloud.google.com/solutions/machine-learning/recommendation-system-tensorflow-overview>

<https://towardsdatascience.com/paper-summary-matrix-factorization-techniques-for-recommender-systems-82d1a7ace74>

<https://www.youtube.com/playlist?list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN>

<https://aihub.cloud.google.com/u/0/p/products%2F88c7d41c-bc75-4e36-a966-50a0aa38fa2a>

<https://datascience.stackexchange.com/questions/58728/whats-the-principal-difference-between-ann-rnn-dnn-and-cnn>

<https://en.wikipedia.org/wiki/Multilayer_perceptron>

<https://towardsdatascience.com/paper-review-neural-collaborative-filtering-explanation-implementation-ea3e031b7f96>

<https://towardsdatascience.com/paper-review-neural-collaborative-filtering-explanation-implementation-ea3e031b7f96>

<https://d2l.ai/chapter_recommender-systems/neumf.html>

<https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed>

<https://surprise.readthedocs.io/en/stable/getting_started.html>

<https://petamind.com/build-a-simple-recommender-system-with-matrix-factorization/>

<https://petamind.com/mlp-for-implicit-binary-collaborative-filtering/>

<https://www.researchgate.net/publication/264023918_A_Literature_Review_and_Classification_of_Recommender_Systems_on_Academic_Journals>