POLITECNICO
MILANO 1863

Numerical Analysis for Machine Learning
Academic Year 2021 - 2022

---

movieRecommendation

Sofia Martellozzo (10623060)
Matteo Nunziante (10670132)

Professor

Edie Miglio

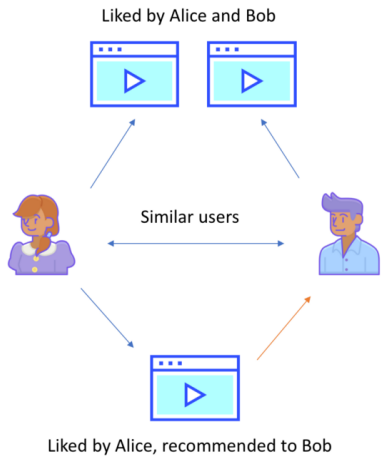
Contents

1 Introduction

With the advent of the internet today, we are witnessing an enormous information overload. This exponential growth in data results in difficulty organizing and analyzing this basic information but opens up new avenues on the paths of knowledge. The question is no longer to have the information but to find the relevant information simultaneously; from there, recommendation systems were born.

Recommender System is a system that seeks to predict or filter preferences according to the user's choices. Recommender systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. Recommender systems produce a list of recommendations in any of the two ways :

- **Collaborative filtering**: Collaborative filtering approaches build a model from the user's past behavior as well as similar decisions made by other users. This model is then used to predict ratings for items that users may have an interest in.



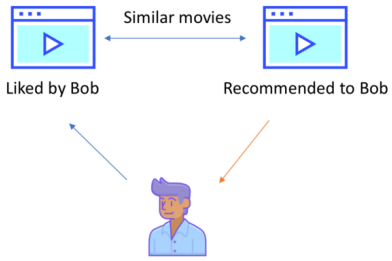
The advantages of this approach are:

  - Domain knowledge not required: The system does not required a domain knowledge because is based only on item ratings.
  - Serendipity: The model can help users discover new interests. In isolation, the ML system may not know the user is interested in a given item, but the model might still recommend it because similar users are interested in that item.

The limitations of this approach are:

- Cold start problem: The prediction of the model for a given (user, item) pair is the dot product of the corresponding embeddings. So, if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item.
- Sparsity: The system may find problems on predicting evaluation because of a situation in which the users evaluate a little of the total number of items available in a dataset. This creates a sparse matrix with a high number of missing values.

- **Content-based filtering**: Content-based filtering approaches uses a series of discrete characteristics of an item in order to recommend additional items with similar properties. Content-based filtering methods are totally based on a description of the item and a profile of the user's preferences. It recommends items based on the user's past preferences.



The advantages of this approach are:

- User autonomy:The model doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to a large number of users.
  The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.
- Immediate consideration of a new item: The model does not need the evaluation of all movie by a user, because it can be recommended without being evaluated.

The limitations of this approach are:

- Content too specific: The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.
- Big scale information: Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of

domain knowledge. The items must be enouth detailed descripted and a user must evaluate several items before the system can iterpret its preferences.

It is also possible to combine these two class of recommendation in order to overcome some limitations they faced. This type of approach is called **Hybrid Recommendation**.
In this project the items that has been evaluated are movies, and their scores are the rating that the users gave after they whatched them.

2 Objectives

The purpose of this project is to develop a recommender system, which predicts the ratings of a user towards a domain-specific item. In this case, this domain-specific item is a movie, and the main focus of our recommendation system is to filter and predict only those movies that a user would prefer. The approach chosen to develop this system is a hybrid one. At first, a collaborative filtering technique has been used to make predictions on movies' ratings deducted from similarities between users. Then with the content-based filtering technique some movies' ratings are predicted, based on similarities between movies. A second approach uses the power of the machine learning developing neural networks to make predictions. The based knowledge of this system derives a data set that is described thoroughly in the next section, while in the fourth section there is a step-by-step description of the algorithm, followed by an evaluation of its performance and an example of output it generates.

3 Dataset description

The data provided are stored in two CSV files named movies.csv and ratings.csv.

3.1 Movies

In this dataset are stored information about the movies. The content is structured in 3 different columns :

- the **id**
- the **title**
- a list of **genres**

each row represents a movie.
In it are stored 10329 movies with 20 different genres:
Western, Documentary, Children, Crime, Film-Noir, Comedy, Adventure, Fantasy, Horror, Thriller, Mystery, Sci-Fi, Musical, Romance, Action, Animation, IMAX, Drama, War, not-defined.
In the following image there is a representation of it:

3.2 Ratings

Here are stored all the ratings that the users gave to the movies from 0.5 to 5. The 4 columns are filled with respectively:

- the id of a **user**
- the id of a **movie**
- the value of the **rate** given by the user on the movie

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
10324	146684	Cosmic Scrat-tastrophe (2015)	Animation|Children|Comedy
10325	146878	Le Grand Restaurant (1966)	Comedy
10326	148238	A Very Murray Christmas (2015)	Comedy
10327	148626	The Big Short (2015)	Drama
10328	149532	Marco Polo: One Hundred Eyes (2015)	(no genres listed)

10329 rows × 3 columns

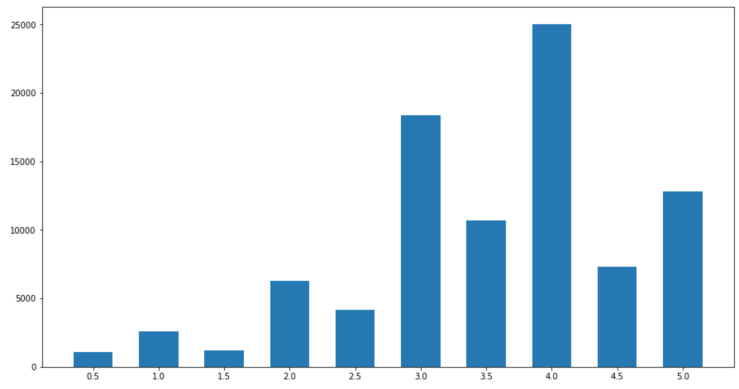
- the **timestamp** (day and time) of the evaluation

Counting the different userId, there are 668 users that overall made 105339 ratings.
To follow a representation of it:

	userId	movieId	rating	timestamp
0	1	16	4.0	1217897793
1	1	24	1.5	1217895807
2	1	32	4.0	1217896246
3	1	47	4.0	1217896556
4	1	50	4.0	1217896523
...	...	...	...	...
105334	668	142488	4.0	1451535844
105335	668	142507	3.5	1451535889
105336	668	143385	4.0	1446388585
105337	668	144976	2.5	1448656898
105338	668	148626	4.5	1451148148

105339 rows × 4 columns

It's also reported the plot of the distribution of the rating data:



4 Algorithm description

In this section a detailed step-by-step description of the algorithm adopted is provided.
At first, the data has been loaded using the *pandas* library and then they have been organized in matrices.
Before building the matrices, the data has been split:

- 80% as training set
- 20% as testing set

It is necessary to shuffle the data before, or in the test set would be only the data about lasts users; the analysis would not be correct.

The first matrix called **ratings matrix** have all the 668 **user ids** as rows, 10329 **movie ids** as columns and as values the **ratings** that users gave to the movies (the ones missing are set to 0.0).

	1	2	3	...	148626	149532
1	0.0	0.0	0.0		0.0	0.0
2	0.0	0.0	2.0		0.0	0.0
3	0.0	0.0	0.0		0.0	0.0
4	0.0	0.0	0.0		0.0	0.0
...						
668	0.0	3.0	0.0		4.5	0.0

The second matrix keeps track of all the genres associated with a movie: it has **movie ids** as rows, **genres** as columns, and as **values** "1" if the movie is associated to a genre, 0 otherwise.

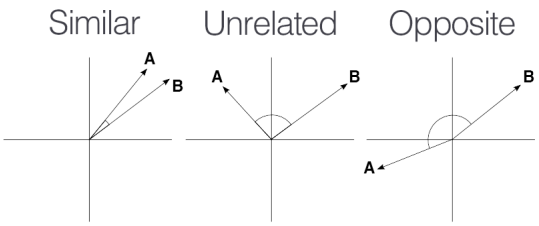
$$CorrelationMatrix_{i,j} = \begin{cases} 1 & \text{if movie i is of genre j} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

	Comedy	Drama	Horror	...	Fantasy	Sci-Fi
1	1	0	0		1	0
2	0	0	0		1	0
3	1	0	0		0	0
4	1	1	0		0	0
...						
149532	0	0	0		0	0

Binary search is an algorithm that find an element in an array with a complexity of O(log n). It searches in a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise, narrow it to the upper half. Repeatedly check until the value is found or the interval is empty. This algorithm is been used to speed up the creation of the training lists with values the indexes they refer to.

4.1 Cosine similarity

It gives a measure of similarity between two non-zero vectors of an inner product space. It calculate the cosine of the angle(θ) between them, which is also the same as the inner product of the same vectors normalized to both have lenght 1.



Because of that it is bounded in the interval [-1,1] for any angle θ :

- two vectors with the same orientation have a cosine similarity of **1**
- two vectors oriented at right angle relative to each other have a similarity of **0**
- two vectors diametrically opposed have a similarity of **-1**

By using the Euclidean dot product formula: $\mathrm{A} \cdot \mathrm{B} = \|A\| \, \|B\| \cos\theta$
We obtain the cosine similarity formula:

$$S_c(A, B) = \cos\theta = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2}$$

where A_i anf B_i are components of vector A and B respectively.

4.2 User-User & Movie-Movie similarity

For the evaluation of the similarities between movies, it's been used the cosine similarity formula between rows of the **correlation matrix**. Instead, for the users' similarities, the formula is been applied using rows of the **ratings matrix** but considering only movies rated by both the users (in order to be considered similar they must have at least five common ratings).

4.3 Clustering

In order to maintain all the similarities between users and movies to avoid later calculation, user clusters and movie clusters were created.
The algorithm used for the building of the user's clusters(analogous for the movie):

- Create a list of user id
- Take the first element of the list and check which one of the others in the list is similar
- Add all the similar found in a new cluster and delete them from the list
- Repeat until the list is empty

In particular, to store the clusters, two dictionaries have been used: one containing all the user's clusters and one containing all the movie's clusters. Their structure is the same, the key is the number of the cluster and the value is a list containing the ids of all the elements belonging to the cluster. Furthermore, other two dictionaries have been created for maintaining direct access to clusters given the id of an element(user or movie): the key is the id and the value is the number of its cluster.
This technique's cheaper from the computation and from the memory point of view and it allows to reduce the workload of the future operations since all the similarities have already been calculated and stored.

4.4 Collaborative filtering

Fill the **ratings matrix**, predicting some missing values given similar behavior between users:

- Given a user, take all the similar users in his cluster
- First variant: for each movie, calculate the mean rating between all the ratings that are given by similar users and assign it to the user.
  Second variant: for each movie, calculate the mean rating between the most common ratings given by similar users and assign it to the user
- Repeat for each user

4.5 Content-based filtering

Given the matrix filled with some values due to the collaborative filtering, try to predict other ratings with a content-based filtering technique between movies:

- Take a user (a row of the **ratings matrix**) and retrieve all the movies' clusters
- For each cluster
  - Calculate the average rating given by the user (ratings ≥ 0.5) only to movies that belong to the cluster;
  - Insert the average in the position corresponding to movies of the current cluster not rated yet by the user or, if it's not rated but predicted by the collaborative filtering, insert the mean between them

4.6 SVD

In linear algebra, the singular value decomposition (SVD) is a factorization of a real or complex matrix. It generalizes the eigendecomposition of a square normal matrix with an orthonormal eigenbasis to any m $\times$ n matrix.

$$S = U\Sigma V^T \tag{3}$$

Specifically, the singular value decomposition of an m $\times$ n complex matrix S is a factorization of the form UΣV^T, where **U** is an m $\times$ m complex unitary matrix, Σ m $\times$ n rectangular diagonal matrix with non-negative real numbers on the diagonal, and **V** is an n $\times$ n complex unitary matrix.
The diagonal entries $\sigma_i = \Sigma_{ii}$ of Σ are known as the singular values of S. The number of non-zero singular values is equal to the rank of S. The columns of U and the columns of V are called the left-singular vectors and right-singular vectors of S, respectively.

4.6.1 SVT

The singular value truncation (SVT) uses the idea of SVD to complete the original matrix. In particular, it's a recursive process that calculates the SVD of the matrix and then tries to reduce its rank by modifying the singular values in Σ. Finally, reconstruct the matrix using the same **U** and **V** but with the new Σ. In the algorithm used the singular values are modified according to an adaptive threshold: its value depends on the number of iterations done. A constant threshold is also possible, for example, threshold = 50 gave good results.

Here are reported the main steps of the algorithm used:

- Copy the current matrix (the **old matrix**)

- Perform the SVD:
$$X = U\Sigma V^T \tag{4}$$
- Calculate the threshold for this step:
$$threshold = b * e^{(-k*a)} \tag{5}$$
where k is the number of iterations already done, a = 0.01, b = 200
- To each singular value greater than zero subtract the threshold and set to zero the ones that become negative
- Build the **new matrix** performing the SVD using the same U and V obtained by the initial decomposition and the diagonal matrix Σ with the new singular values
- Update the new matrix obtained in the previous point with the values fixed, before the application of the algorithm: the true ratings and the ones predicted
- If there are negative values in the new matrix, set them to zero
- Calculate the increment: the difference in module between the **new matrix** and the **old matrix**
- Repeat until k is lesser than the maximum number of iterations available and the increment is greater than epsilon, where their values are, respectively, 100 and 0.1

4.7 Neural Network

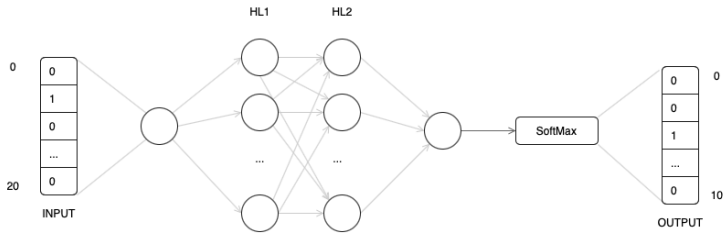
Instead of using collaborative and content-based filtering, it's also possible to use Neural Networks to do it implicitly. In the second solution provided it's been created a neural network for each user. Each NN has been trained using two different types of data:

- Input: a vector for each movie directly rated by the user (the vector correspond to a row of the **correlation matrix**).
  Output: a vector of 10 positions for each movie, where each position corresponds to a rating (from 0.5 to 5) and with all zero except in the position corresponding to the right rate where there is 1
- The same but with movies the users haven't rated. In this case, ratings of similar users are been used

4.7.1 Structure of the neural network

The structure of the neural networks is made of 3 types of layers:

- Input layer: it's the layer that receives the input vector that, in the case considered, is a vector representing a movie given its genres. Thus this layer has 20 neurons, one for each cell of the input.
- Hidden layers: the quantity of these layers and their dimensions (how many neurons each layer has) could be any. For the project, it's been decided to use 2 hidden layers with sizes, respectively, 32 and 16.
- Output layer: his size depends on the dimension of the output required, since there are 10 possible ratings from 0.5 to 5 the dimension of this layer is 10 and in each position, there is the probability to have the corresponding rate given an input.



4.7.2 Activation function

The activation function characterizes neurons in hidden and output layers. Every node in the same layer must have the same activation function, while it could change between layers. The activation functions used are:

- **Soft max**: it's been used specifically for the output layer since it allows to have a vector with only positive values and with their sum equal to 1, exactly as a probability vector.

$$\sigma(z_i) = \frac{\exp z_i}{\sum_{j=0}^{n} \exp z_i}$$

  Where n is the size of the output layer and z_i is the value corresponding to the i-th rating

- **Tanh**:

$$\sigma(z) = \tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- **ReLU**:

$$\sigma(z) = \max(0, z)$$

- **Leaky-ReLU**:
$$\sigma(z) = \max(0.1z, z)$$
- **Swish**:
$$\sigma(z) = \frac{z}{1+e^{-z}} = z \times sigmoid(z)$$

4.7.3 Optimization method

Two different optimizations method have been used for the training of the neural networks:

- Mini-batch stochastic gradient descent (MB-SGD)
$$\theta^{(0)} given$$
$$for k = 0, 1, ..., n_{epochs}$$
$$g^k = \frac{1}{|I_k|} \times \sum_{i \in I_k} \nabla_\theta J(x_i, y_i, \theta^{(k)})$$
$$\theta^{(k+1)} = \theta^{(k)} - \lambda_k \times g^{(k)}$$
- Stochastic gradient descent with momentum
$$\theta^{(0)} given, v^{(0)} = 0$$
$$for k = 0, 1, ..., n_{epochs}$$
$$g^k = \frac{1}{|I_k|} \times \sum_{i \in I_k} \nabla_\theta J(x_i, y_i, \theta^{(k)})$$
$$v^{(k+1)} = \alpha v^{(k)} - \lambda_k \times g^{(k)}$$
$$\theta^{(k+1)} = \theta^{(k)} + v^{(k+1)}$$

Where θ are the parameters, $\mathbf{g}$ is the gradient of the loss function with respect to θ divided by the dimension of the mini-batch I_k.
The mini-batch is a random subset of the inputs. Instead, $\mathbf{v}$ is the "velocity" vector, as a matter of fact its value is big if far from the minimum, while its value is small if close to the minimum.

4.7.4 Loss function

For this project two different loss functions have been evaluated:

- Mean square error (MSE):
$$J = \frac{1}{N} \times \sum_{n=1}^{N} (y_i - y_{i,pred})^2$$

- Cross entropy:

$$J = -[y \times \ln(y_{pred}) + (1 - y) \times \ln(1 - y_{pred})]$$

Where **N** is the dim of the output vector, **y** is the true output and y_{pred} is the prediction.

4.8 Possible improvements

The traditional collaborative filtering algorithms produce low temporal diversity, it recommended the same top-N items to users concluding that there was not much difference between an evaluation made many years before and one made recently. In a real scenario it is possible that the preferences of a user could change over time, therefore use the timestamp of the ratings could make a better prediction. At first it requires the traditional collaborative filtering in order to find the k-nearest neighbours for each user. The steps are:

- create user and movie cluster with cosine similarity
- create a dictionary for each user with, as key, the user id and as value the list of movies ids he evaluate
- order the movie list follow a timeline based on the timestamp on which the rating has been made
- get one user at time, call it *'target'*
- from the dictionary extract the timestamp of his last rating (t)
- get all movie dictionary of all users similar to the target one
- select only the movie evaluated by the neighbours from time t or after, that will be considered for the prediction
- the ratings prediction of the movie in the new list will be the average of the neighbours rating on that movie

In this way the accuracy is higher than the traditional collaborative filtering algorithms, but this approach would also mean losing out on other valid recommendations (excluding all the ratings made before the time 't').

4.9 Assumptions

During the development of the system, some assumption has been made:

- **Clustering**: to speed up the somputation of the similarity between user and movie, some group of these elements similar has been reated.
  - For the clusers containing users, the similarity is evaluated based on the same ratings behaviour. The algorithm generates 161 cluster for the users.

  - The other set of clusters contains movies similar for their genres. In this case the amount of clusters is 200.

  Another possibility could have been to generate a similarity matrix with user-user and item-item as rows-columns, filled with the cosine similarity calculated between each users or movies. This could be more accurate and precise, but makes the algolithm much more slower and computationally heavy.

- **Threshold** for similarity: to evaluate two elements as similar, their cosine similarity have to return a value grather or equal than 0.7 (for the movies) and 0.95 (for the users)

- **Rating recommended**: a movie is recommended by the system if the rating predictes has a value equal or grather than 3.

- **SVT**: the threshold selected to compute the truncation is calculated with two constant a and b, respectively 0.01 and 200, and k, the number of the iteration.

- **Hyperparameters**: to train the neural network the best params found are: 2000 epochs, using the Gaussian distribution to inizitialize the weights, select the learning rate with the linear decay between a minimum learning rate $\lambda_{min} = e^{-4}$ and a maximum learning rate $\lambda_{max} = e^{-2}$ and decay lenght K=1000, implement the stocastic gradient descent with momentum with alpa $\alpha = 0.95$ and minibatch of a size equal to the 30% of the input size.

- **Hiddel layer**: the size of the hidden layer of the neural network are 32 and 16 respectivelly.

5 Results

It's been implemented a function that given a user id retrieve 10 movies the system would recommend. An example is reported below:

N°	Title	Rating predicted
1	Dream Man (1995)	5
2	Faster Pussycat! Kill! Kill! (1965)	5
3	Endless Summer 2, The (1994)	5
4	Heaven & Earth (1993)	5
5	Purple Noon (Plein soleil) (1960)	5
6	Stonewall (1995)	5
7	Little Lord Fauntleroy (1936)	5
8	American Dream (1990)	5
9	Schizopolis (1996)	5
10	Quest for Camelot (1998)	5

Table 1: Recommend movies for user with id 248 by collaborative and content filtering

Add the same but from the NN

N°	Title	Rating predicted
1	Shawshank Redemption, The (1994)	4.5
2	Balto (1995)	4
3	Power (1995)	4
4	City of Lost Children, The (Cité des enfants perdus, La) (1995)	4
5	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	4
6	Wings of Courage (1995)	4
7	Seven (a.k.a. Se7en) (1995)	4
8	Lamerica (1994)	4
9	Lawnmower Man 2: Beyond Cyberspace (1996)	4
10	Screamers (1995)	4

Table 2: Recommend movies for user with id 565 with neural network

5.1 Evaluation

5.1.1 RMSE

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how to spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

The formula is:

$$RMSE = \sqrt{\frac{1}{|\Omega_{test}|} \sum_{i,j \in \Omega_t est} (r_{i,j} - r_{i,j}^{pred})} \tag{6}$$

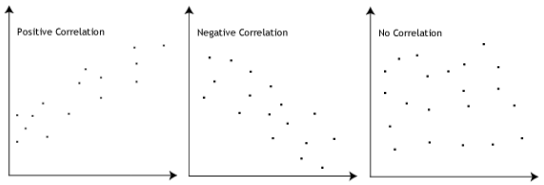
5.1.2 Rho

Pearson correlation coefficient is a measure of linear correlation between two sets of data. It is the ratio between the covariance of two variables and the product of their standard deviations; thus it is essentially a normalized measurement of the covariance, such that the result always has a value between -1 and 1. As with covariance itself, the measure can only reflect a linear correlation of variables and ignores many other types of relationship or correlation. It is calculated as follow:

$$\rho = \frac{\sum_{\Omega_{test}} (r_{ij} - \overline{r})(r_{ij}^{pred} - \overline{r^{pred}})}{\sqrt{\sum_{\Omega_{test}} (r_{ij} - \overline{r})^2} \sqrt{\sum_{\Omega_{test}} (r_{ij}^{pred} - \overline{r^{pred}})^2}} \tag{7}$$

The result is a value between [-1,1]:

- 1 = perfect match-strong positive relation; all data points lie on a line for which both variable r and r_{pred} are increasing
- 0 = random; there is no linear dependency between the variables
- -1 = strong negative relation; also here a linear equation describes the relationship between the variables r and r_{pred} and the points lie on a line in which are decreasing



5.2 Precision

Precision tries to answer the following question:
'What proportion of positive identification was actually correct?'

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

5.3 Recall

Recall tries to answer the following question:
'What proportion of actual positives was identified correctly?'

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

5.4 F1-measure

It's a measure given by the relation between precision and recall

$$F1 - score = 2 \times \frac{PrecisionxRecall}{Precision + Recall} \tag{10}$$

5.5 Collaborative & content-based filtering results

The results obtained by collaborative filtering, content-based filtering and their combination are reported in table 3.

Config	RMSE	rho	Precision	Recall	F1-score
Only Coll.F	1.347	0.279	0.874	0.768	0.817
only Con.F	0.972	0.424	0.873	0.847	0.860
Coll.F V1 + Con.F	0.926	0.481	0.869	0.900	0.884
Coll.F V2 + Con.F	0.909	0.495	0.871	0.899	0.885

Table 3: Results of collaborative and content-based filtering

From the results, it's possible to see how the hybrid model is much better than just applying only one filtering technique both in terms of error and F1-score.

5.6 Neural network results