ỨNG DỤNG DỮ LIỆU LỚN

PROJECT 1: RECOMMENDER SYSTEMS

1.Giới thiệu:

1.1 Mô tả đồ án:

Xây dựng hệ thống tư vấn cơ bản.

1.2 Ngôn ngữ:

Python

1.3 Source code:

Link source code: [final PA1.ipynb - Colaboratory (google.com)]

1.4Thông tin nhóm :

MSSV	Họ và tên
18120505	Đào Quốc Phong
18120525	Đoàn Thanh Quang
18120528	Nguyễn Như Quang
18120533	Dương Đoàn Bảo Sơn

2.Nội dung thực hiện:

2.1 Tập dữ liệu:

MovieLen-1M

Movielens là một nhóm nghiên cứu cung cấp các bộ dữ liệu cho các bài toán xây dựng hệ thống gợi ý. Các bộ dữ liệu trong tập này bao gồm thông tin đánh giá xếp hạng của người dùng tới các bộ phim. Những thông tin về người dùng hay các bộ phim cũng được cung cấp.

Bộ dữ liệu này bao gồm xấp xỉ 1 triệu bộ (user, movie, rating) từ khoảng 3900 bộ phim và 6040 người dùng.

Sparsity : 4,24%

EachMoive

Trung tâm Nghiên cứu Hệ thống Compaq đã chạy dịch vụ EachMovie trong 18 tháng.

61265 người dùng đã nhập tổng số 2811983 xếp hạng số trên 1623 phim, tức là khoảng 2,4% mục nhập được xếp hạng từ 0 đến 5 sao.

Chỉ số người dùng nằm trong khoảng từ 1 đến 74424.

Chỉ số phim từ 1 đến 1648.

2.2 Triển khai hệ thống:

2.3 Thư viện:

Dùng hàm SVD của thư viện surprise

Dự đoán $\hat{r}_{ui}$ được đặt là: $\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$

Nếu người dùng ***u*** là không xác định, thì độ lệch b_u và các factors p_u được giả định là 0. Điều tương tự cũng áp dụng cho mặt hàng ***i*** với b_i và q_i

Để ước tính tất cả những điều chưa biết, chúng tôi giảm thiểu lỗi bình phương sau:

$$\sum_{r_{ui} \in R_{train}} \left(r_{ui} - \hat{r}_{ui}\right)^2 + \lambda \left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2\right)$$

Việc thu nhỏ được thực hiện bởi **stochastic gradient descent** đơn giản:

$$\begin{aligned} b_u &\leftarrow b_u &+ \gamma(e_{ui} - \lambda b_u) \\ b_i &\leftarrow b_i &+ \gamma(e_{ui} - \lambda b_i) \\ p_u &\leftarrow p_u &+ \gamma(e_{ui} \cdot q_i - \lambda p_u) \\ q_i &\leftarrow q_i &+ \gamma(e_{ui} \cdot p_u - \lambda q_i) \end{aligned}$$

Trong đó: $e_{ui} = r_{ui} - \hat{r}_{ui}$. Các bước này được thực hiện trên tất cả các xếp hạng của tập hợp và lặp lại **n_epochs** lần. Đường cơ sở được khởi tạo bằng **0**. Yếu tố người dùng và mục được khởi tạo ngẫu nhiên theo phân phối chuẩn, có thể được điều chỉnh bằng cách sử dụng các tham số **init_mean** và **init_std_dev**.

Bạn cũng có quyền kiểm soát tốc độ học tập γ và thời hạn chính quy hóa. λ. Cả hai đều có thể khác nhau đối với từng loại thông số. Theo mặc định, tốc độ học tập được đặt thành 0,005 và các điều khoản chính quy được đặt thành 0,02.

Bạn có thể chọn sử dụng phiên bản không bias của thuật toán này, chỉ cần dự đoán:

$$\hat{r}_{ui} = q_i^T p_u$$

Điều này tương đương với Hệ số hóa ma trận xác suất và có thể đạt được bằng cách đặt tham số **biased** thành **False**

2.4 Độ Đo:

❖ RMSE:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(y_i - \tilde{y}_i\right)^2}{n}}$$

❖ Hit-rate:

Tính hit rate cho mỗi user:

- Tìm tất cả các items trong lịch sử của user đó.
- Bỏ đi một item trong những item trước.
- Sử dụng toàn bộ items còn lại để đưa ra tư vấn cho user đó.
- Nếu item được bỏ ở bước 2 nằm trong danh sách các item tư vấn thì được xem là 1 hit.
- Hit rate của cả hệ thống là số lượng hit chia cho tổng số các thử nghiệm. Điều đó có nghĩa là đo tần suất hệ thống có thể đề xuất các rating đã xóa,thông số càng cao càng tốt.

❖ NDCG:

$$CG_p = \sum_{i=1}^{p} rel_i$$

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

Where

$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

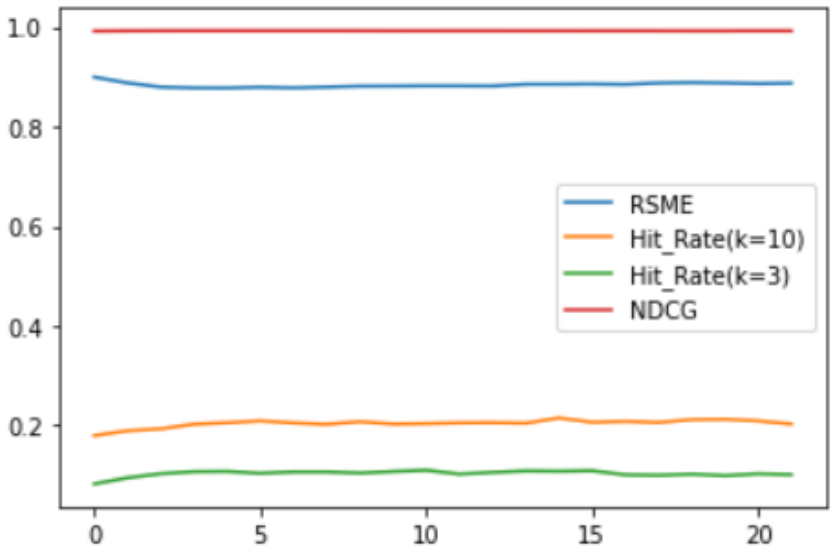
2.5 Kết Quả dữ liệu MovieLen_1M:

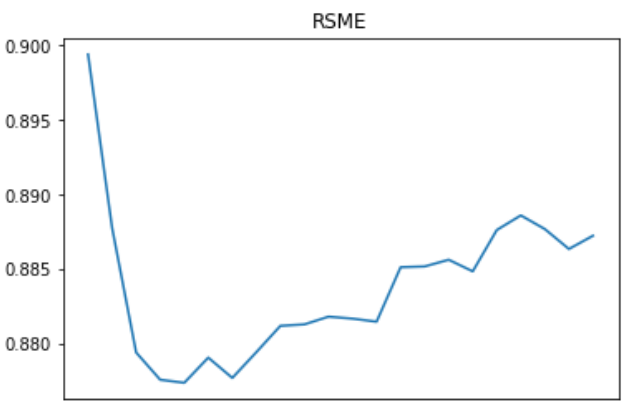
Tự code thuật toán Hit Rate dựa vào code mẫu cô đưa và tự code thêm.
Vì hàm build_anti_testset() của surprise tạo tất cả các rating bị thiếu, dữ liệu quá lớn đã dẫn đến tràn bộ nhớ colab nên nhóm đã code lại phần này dùng dataframe của thư viện pandas.
Dùng thuật toán NDCG của scikit-learn vì thuật toán này đưa ra giá trị tốt hơn, số thập phân gần bằng một trong khi thuật toán của recommender đưa giá trị làm tròn là 1.0

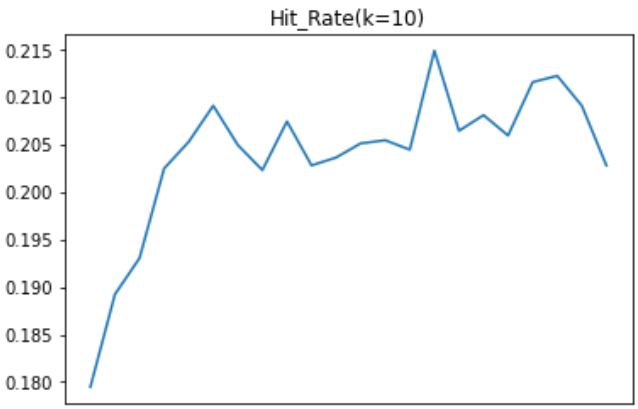
Chạy Thuật toán SVD với tham số n_factor thay đổi từ 1 tới 200:

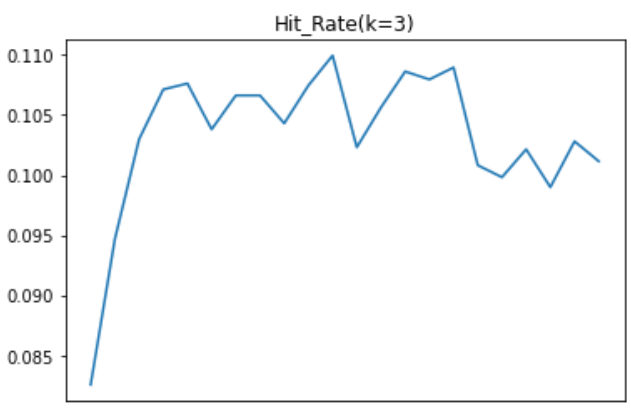
Name	RSME	Hit_Rate(k=10)	Hit_Rate(k=3)	NDCG
n_factors = 1	0.899409	0.179470199	0.082615894	0.991829
n_factors = 5	0.88778	0.189238411	0.094701987	0.992231
n_factors = 10	0.879425	0.193046358	0.102980132	0.992404
n_factors = 20	0.877597	0.202483444	0.107119205	0.992453
n_factors = 30	0.877391	0.205298013	0.107615894	0.992498
n_factors = 40	0.879079	0.20910596	0.103807947	0.992452
n_factors = 50	0.877719	0.204966887	0.106622517	0.992459
n_factors = 60	0.879447	0.202317881	0.106622517	0.992444
n_factors = 70	0.881211	0.207450331	0.104304636	0.992404
n_factors = 80	0.881308	0.20281457	0.107450331	0.992298
n_factors = 90	0.881825	0.203642384	0.109933775	0.992324
n_factors = 100	0.881693	0.20513245	0.102317881	0.992364

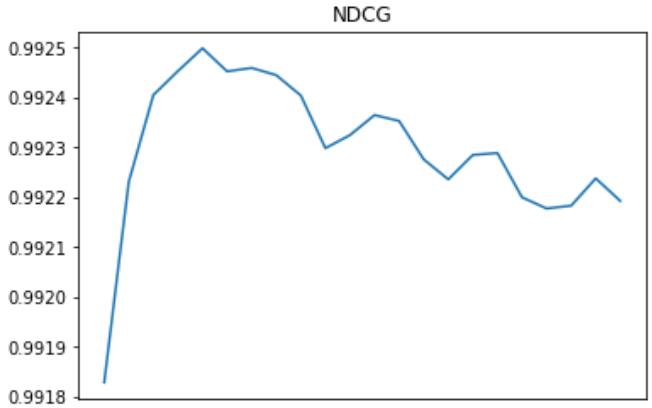
n_factors = 110	0.881489	0.205463576	0.105629139	0.992352
n_factors = 120	0.885142	0.204470199	0.108609272	0.992276
n_factors = 130	0.885192	0.214900662	0.10794702	0.992235
n_factors = 140	0.885641	0.206456954	0.108940397	0.992284
n_factors = 150	0.884863	0.208112583	0.100827815	0.992288
n_factors = 160	0.887635	0.205960265	0.099834437	0.992199
n_factors = 170	0.888615	0.211589404	0.102152318	0.992177
n_factors = 180	0.887696	0.212251656	0.099006623	0.992183
n_factors = 190	0.886363	0.20910596	0.10281457	0.992237
n_factors = 200	0.887251	0.20281457	0.10115894	0.992192









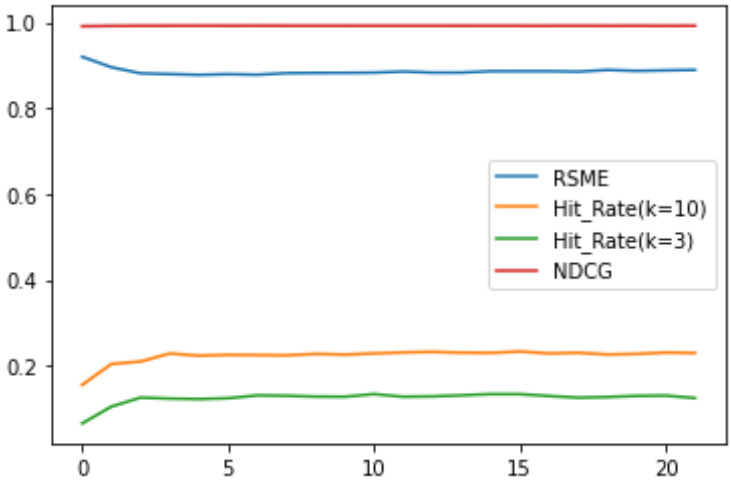


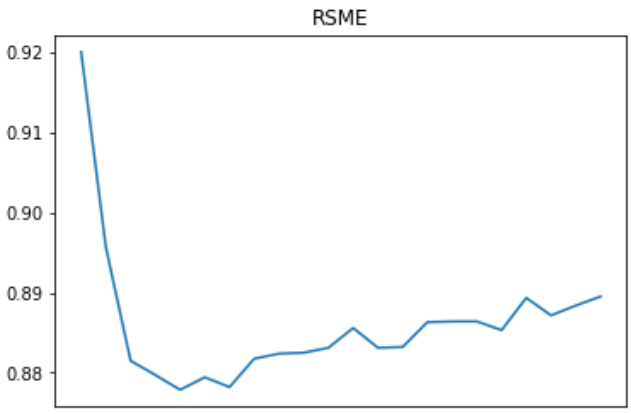
- Tổng quát, ta thấy n_factors không làm cho độ hiệu quả của thuật toán thay đổi quá nhiều.
- Cụ thể thì ở mức n_factors bằng 1 thì độ hiệu quả của SVD giảm mạnh, RSME cao, Hit Rate và NDCG thấp hơn nhiều so với số n_factors lớn hơn.
- Tới một số n_factors cụ thể, SVD đạt RSME thấp nhất, số này ở mức 20-30.
- Càng tăng n_factors thì RSME tăng NDCG giảm, nhưng ngược lại độ Hit Rate cũng tăng. Nhưn độ tăng của Hit Rate không nhanh bằng độ tăng của RSME và độ giảm của NDCG.
- Nên SVD hiệu quả nhất với bộ dữ liệu MovieLen_1M khi n_factors ở mức 20-30.

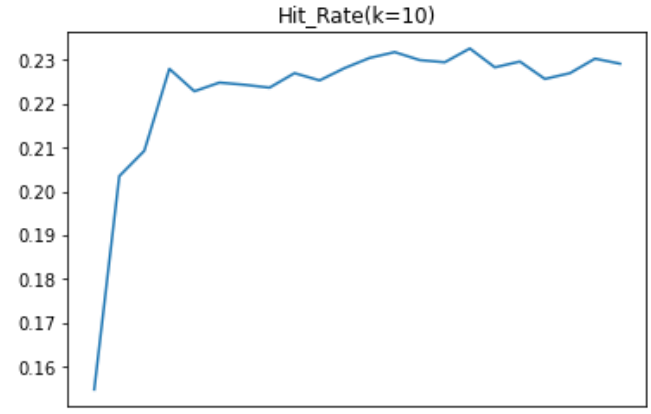
Chạy thuận toán SVD với biased = True, không tính độ lệch cho tham số n_factors.

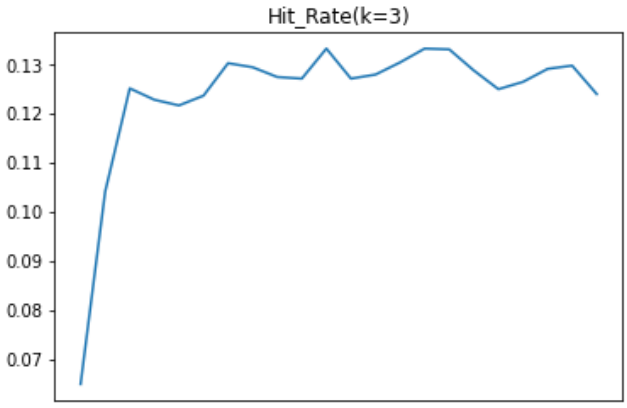
Name	RSME	Hit_Rate (k=10)	Hit_Rate(k=3)	NDCG
n_factors = 1	0.920085	0.154801325	0.064900662	0.991345
n_factors = 5	0.895664	0.203476821	0.104139073	0.992061
n_factors = 10	0.881477	0.209271523	0.125165563	0.992393
n_factors = 20	0.8797	0.227980132	0.122847682	0.99247
n_factors = 30	0.87785	0.222847682	0.121688742	0.992562
n_factors = 40	0.879418	0.224834437	0.123675497	0.992458
n_factors = 50	0.878192	0.224337748	0.130298013	0.992527
n_factors = 60	0.881744	0.223675497	0.129470199	0.992463
n_factors = 70	0.882371	0.226986755	0.127483444	0.992454
n_factors = 80	0.882487	0.225331126	0.127152318	0.992434

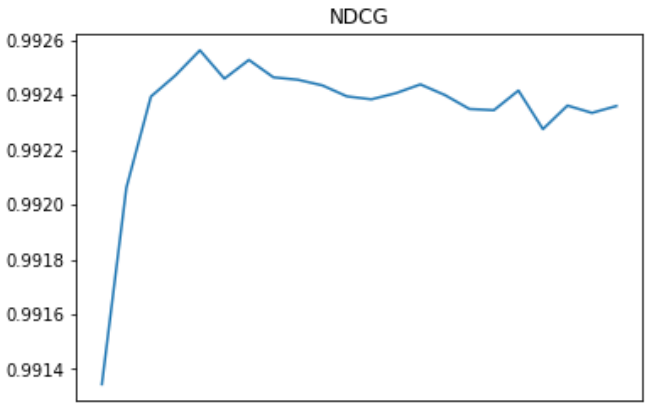
n_factors = 90	0.883102	0.228145695	0.133278146	0.992394
n_factors = 100	0.885584	0.230463576	0.127152318	0.992383
n_factors = 110	0.883103	0.231788079	0.127980132	0.992406
n_factors = 120	0.883206	0.229966887	0.130463576	0.992438
n_factors = 130	0.886315	0.229470199	0.133278146	0.992399
n_factors = 140	0.886396	0.232615894	0.133112583	0.992348
n_factors = 150	0.8864	0.228311258	0.128807947	0.992344
n_factors = 160	0.885311	0.229635762	0.125	0.992415
n_factors = 170	0.889343	0.225662252	0.126490066	0.992275
n_factors = 180	0.88715	0.226986755	0.129139073	0.99236
n_factors = 190	0.888386	0.230298013	0.129801325	0.992334
n_factors = 200	0.889506	0.229139073	0.124006623	0.992359











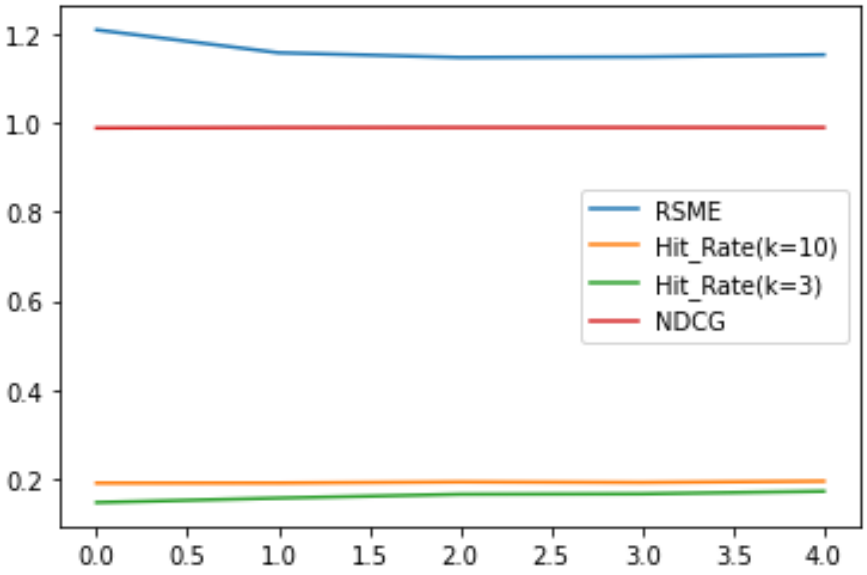
Tương tự như thí nghiệm trước, ta thấy:

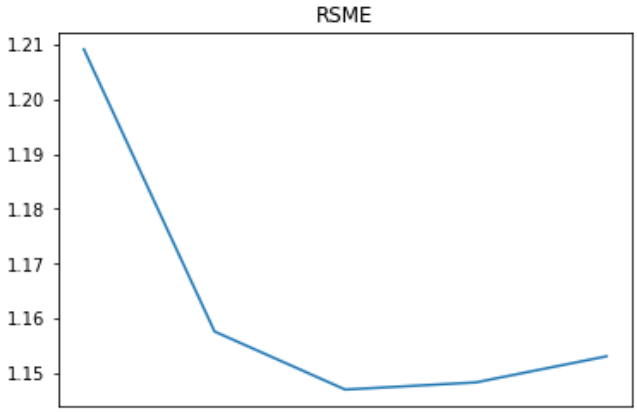
- Khi n_factors quá thấp, hiệu quả của thuật toán bị giảm nhiều.
- Khi n_factors quá cao, hiệu quả của thuật toán cũng giảm đi.
- Sơ đồ chi tiết cho thấy khi biased = False, tức là ta không quan tâm đến độ chệch thì hiệu quả của thuật toán ổn định hơn.

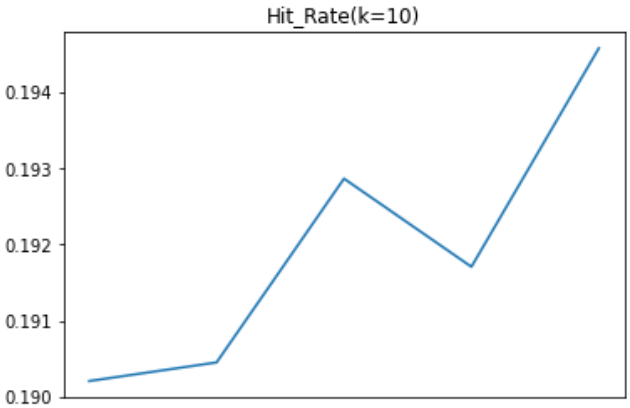
2.6 Kết Quả dữ liệu EachMovie:

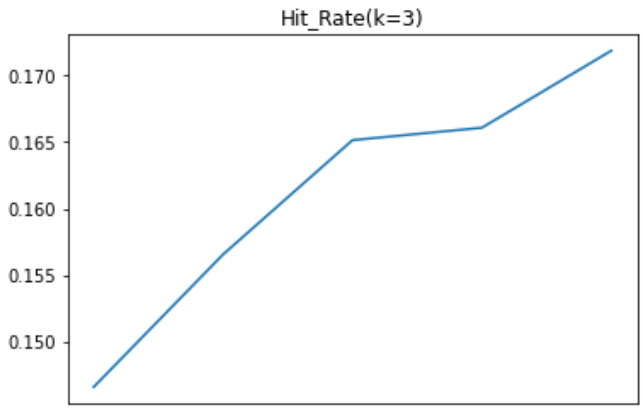
Chạy thuật toán với biến n_factors.

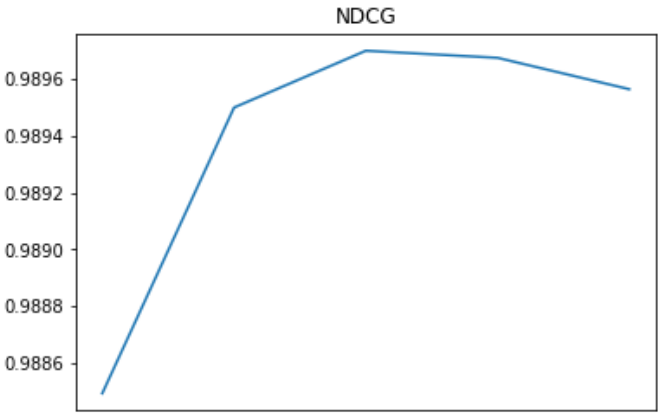
Name	RSME	Hit_Rate (k=10)	Hit_Rate (k=3)	NDCG
n_factors = 1	1.2091	0.19020648	0.146625316	0.988492
n_factors = 5	1.157623	0.190451318	0.156565739	0.9895
n_factors = 10	1.147029	0.192867053	0.165135069	0.9897
n_factors = 20	1.148336	0.191708153	0.166065453	0.989675
n_factors = 50	1.153095	0.194580919	0.17184363	0.989565


1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
RSME
Hit_Rate(k=10)
Hit_Rate(k=3)
NDCG


RSME
1.21
1.20
1.19
1.18
1.17
1.16
1.15

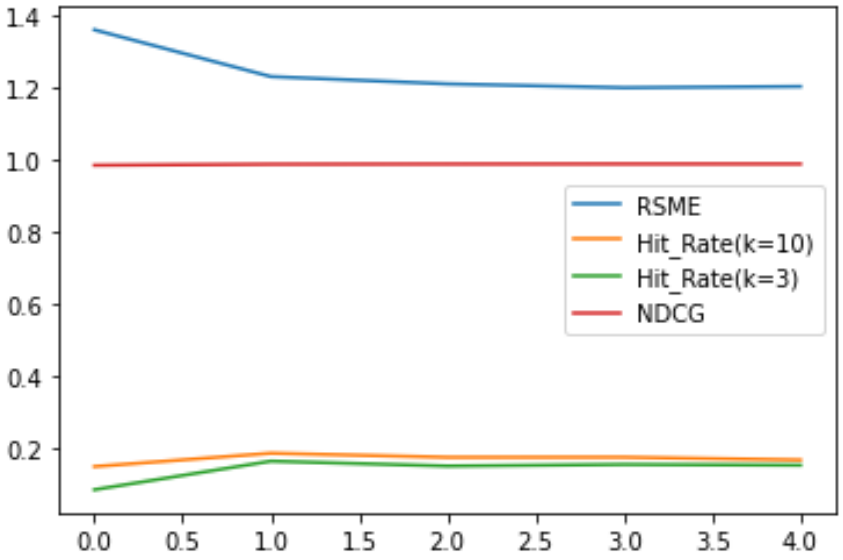

Hit_Rate(k=10)
0.194
0.193
0.192
0.191
0.190

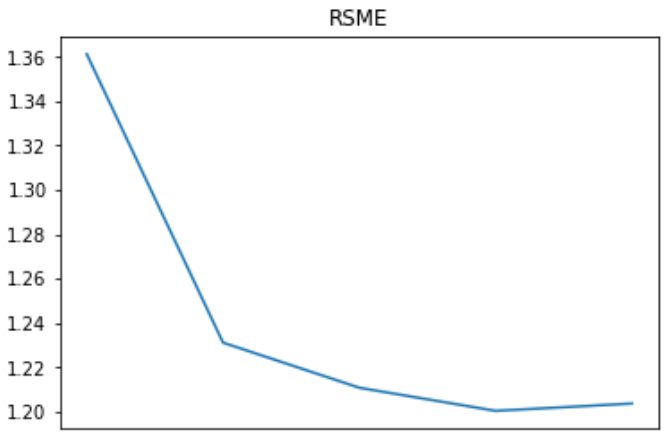

Hit_Rate(k=3)
0.170
0.165
0.160
0.155
0.150

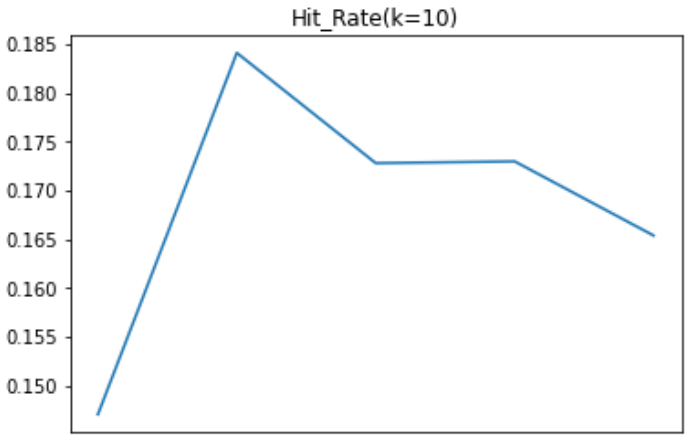

NDCG
0.9896
0.9894
0.9892
0.9890
0.9888
0.9886

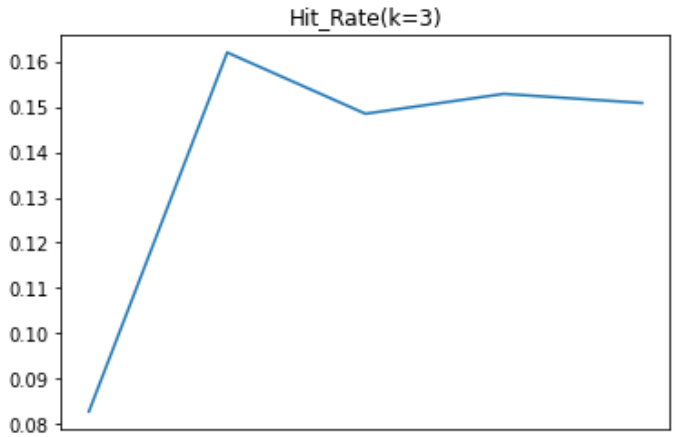
Biến n_factors và biased = False

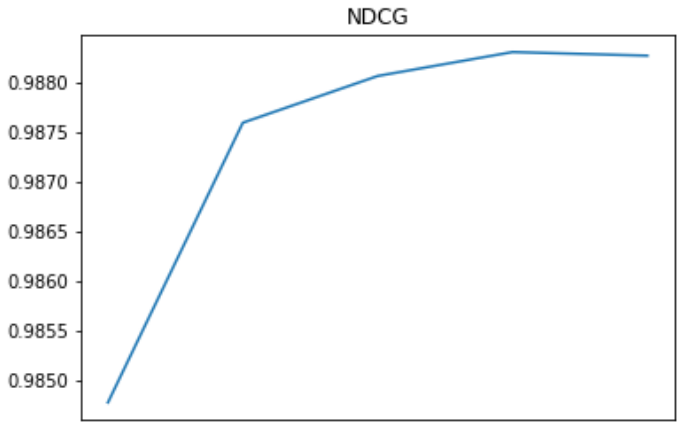
Name	RSME	Hit_Rate(k=10)	Hit_Rate(k=3)	NDCG
n_epochs = 1	1.332087	0.096449849	0.093838244	0.985661
n_epochs = 5	1.219401	0.144079001	0.125438668	0.988427
n_epochs = 10	1.160092	0.173231045	0.151620011	0.989548
n_epochs = 20	1.158576	0.197763813	0.171729372	0.98946





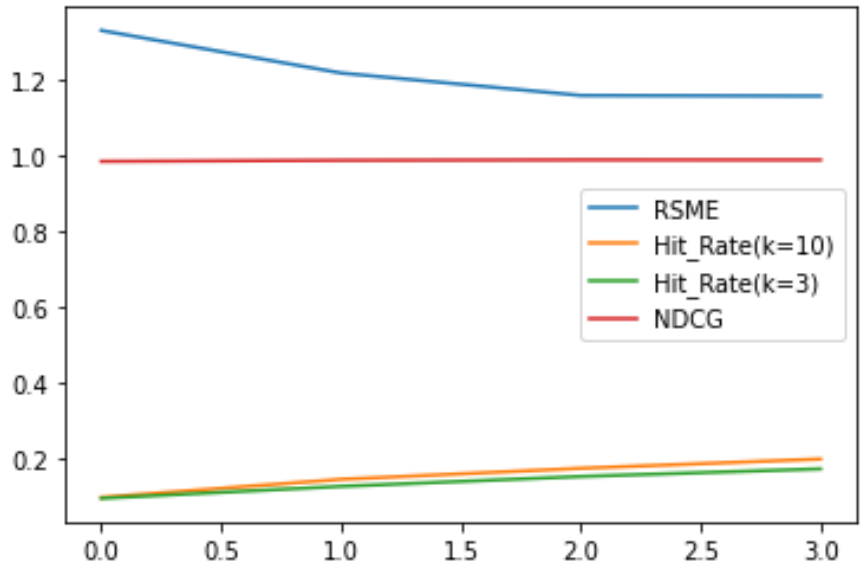


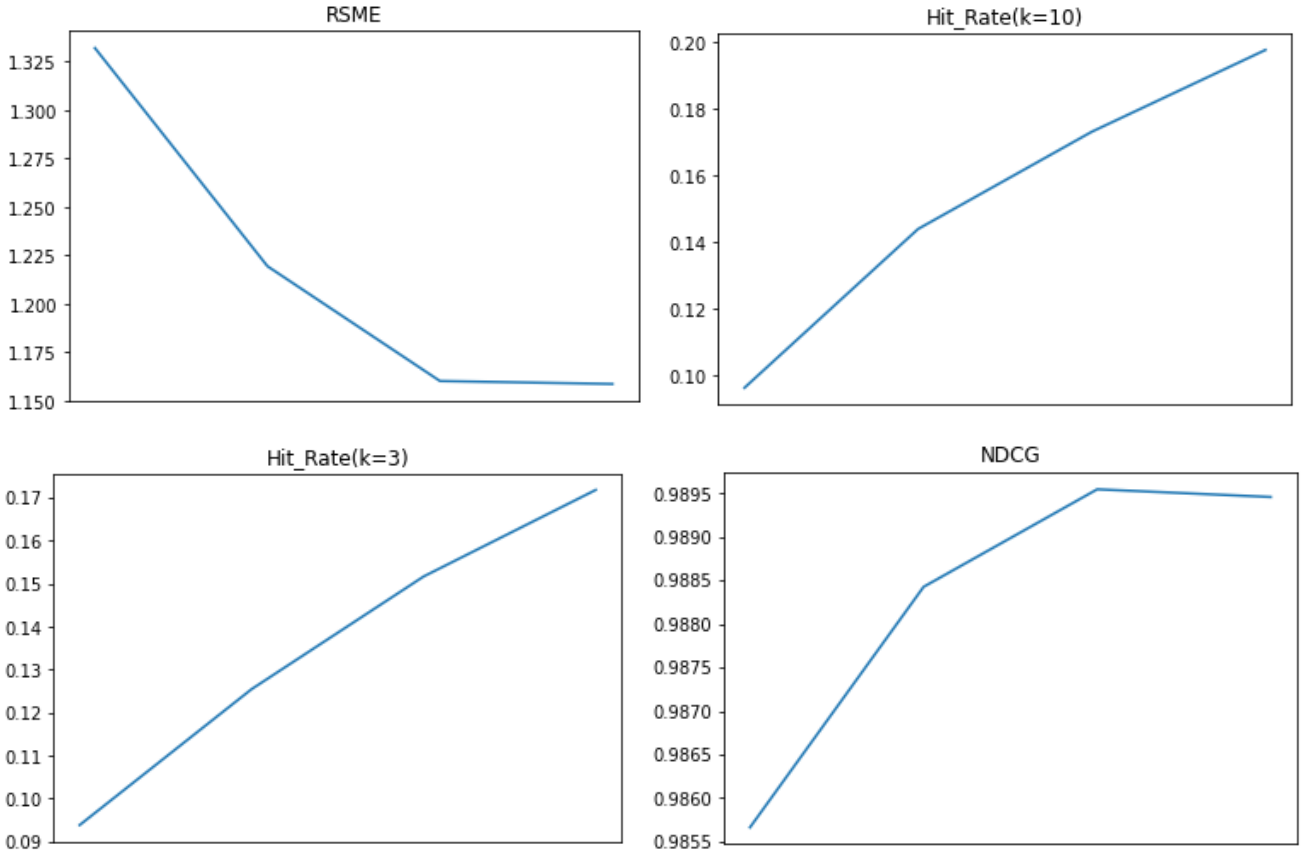




Biến n_epochs

Name	RSME	Hit_Rate(k=10)	Hit_Rate(k=3)	NDCG
n_epochs = 1	1.332087	0.096449849	0.093838244	0.985661
n_epochs = 5	1.219401	0.144079001	0.125438668	0.988427
n_epochs = 10	1.160092	0.173231045	0.151620011	0.989548
n_epochs = 20	1.158576	0.197763813	0.171729372	0.98946





3. Tài liệu tham khảo:

Bộ dữ liệu MovieLens-1M — Machine Learning cho dữ liệu dạng bảng (machinelearningcoban.com)
Machine Learning cơ bản (machinelearningcoban.com)
https://surpriselib.com/