1. 서론

A. 제안 배경 및 필요성

i. 사용자 맞춤형 서비스의 부상

1. 2013 년까지 전세계적으로 4.4 제타바이트의 데이터가 형성되었으며, 2020 년까지는 그 10 배인 44 제타바이트까지 그 양이 증가할 예정이다[1]. 한 명의 사람은 수많은 데이터를 남기고, 그러한 데이터를 모아 여러 기업 및 정부가 가공하여 유의미한 정보를 읽으려고 노력하고 있다.
2. 특히 이러한 데이터는 개인의 니즈를 정확히 파악하여 개인에게 딱 맞는 서비스를 제공하기 위한 수단으로 사용되고 있다. 많은 기업들이 소비자의 변화하는 기호를 인지하기 위하여 소비자가 제품 및 서비스와 관련하여 남긴 데이터를 적극적으로 수집하고 있다.

ii. 소비자 행동특성에 기반한 접근

1. 넷플릭스(Netflix)의 부회장 토드 옐린(Todd Yellin)이 최근 인터뷰에서 “주소, 나이, 성별과 같은 정보는 모두 쓰레기다”라는 다소 과격한 표현을 통해 기존의 빅데이터 분석 방식의 틀을 비판했다[2]. 소비자의 니즈를 정확히 파악하기 위해서는, 더이상 전통적인 인구통계학적 데이터에 얽매여서는 안 된다는 의미로 풀이된다. 전체 시청되는 콘텐츠 중 75%가 추천 시스템에 의하여 소비되고 있는 넷플릭스에서 이러한 발언을 한 것은 상당한 의미가 있다[3].
2. 기존의 인구통계학적 데이터는 계속해서 수집되고 있으나, 소비자의 행동패턴에 대한 관심이 더욱 커지고 있는 것도 사실이다. 어떠한 콘텐츠에 주목하고, 소비하고, 재구매하는지에 대한 정보를 꾸준히 추적하여 소비자 의견을 반영하려는 기업들이 많아지고 있다.

iii. 추천 시스템의 비교

1. 소비자의 여러 요소를 파악하여 이들이 가장 원하는 제품 및 서

비스를 제공하려는 추천 시스템은 다양한 알고리즘으로 이루어져 있다. 본 연구는 이러한 알고리즘들을 동일한 입력데이터값과 동일한 기준으로 비교하여 어떠한 알고리즘이 가장 정확한 추천 시스템인지를 밝히고자 한다.

iv. 오픈 소스의 활용

1. 개발자들 사이의 더 나은 결과물을 내기 위한 협력은 지속되어왔다. 오픈 소스는 이러한 협력을 더욱 편리하고 효과적으로 돕고 있다. 소스를 공개하여 다른 개발자들에게 조언을 받고 그에 기반한 수정사항들을 반영한다. 본 연구 역시 그렇게 공개된 오픈 소스들을 비교하고자 한다.

B. 졸업작품의 목표

i. 본 연구는 무비렌즈(http://movielens.org/)에서 수집된 정보를 입력데이터로 활용한다. 특정 영화에 대한 소비자들의 평가이력을 주축으로 하는 다양한 정보를 활용하여, 동일한 입력데이터에 대해 가장 높은 정확도를 보이는 추천 알고리즘을 밝혀내고자 한다.

ii. 비교 대상은 다음과 같다.

1. Surprise(http://surpriselib.com)
2. Python-recsys(http://ocelma.net/software/python-recsys/build/html/)

2. 비교 알고리즘 소개

A. Surprise

i. Surprise 는 Python Scikit 에 기반한 추천시스템 분석툴로, 강력한 API 가 제공된다. 머신 러닝 전문가 Nicolas Hug 가 개발했으며, 꾸준히 관련 가이드가 업데이트 되고 있다.

ii. Dataset handling 에 특히 강점을 가지고 있으며, SVD(Singular-value decomposition)뿐만 아니라 PMF, NMF 와 같은 다양한 Matrix-factorzation 으로도 분석할 수 있다.

iii. Baseline algorithm, neighborhood algorithm, neighborhood methods

및 여러 유사성 검정 모듈이 포함되어 있다.

B. Python-recsys

i. Python-recsys 는 SVD 형태의 데이터를 분석하여 아이템을 추천하는 추천시스템으로, 스페인 출신 개발자인 Oscar Celma 가 개발한 라이브러리이다.

ii. Divisi2 와 csc-pysparse 기반으로 제작되었으며, 두 라이브러리를 위해 Numpy 와 Scipy 역시 요구된다.

iii. 일반적으로 활용되는 SVD 형태로 분석으르 진행함으로써, 다른 라이브러리에서의 명령어와 상당히 유사하게 개발되었다. 그러한 점에서 특별한 튜토리얼 없이 바로 이용할 수 있다는 장점이 있다.

3. 비교 대상 데이터셋

A. 데이터셋 메타데이터

i. 데이터셋은 무비렌즈(http://movielens.org)에서 수집된 실사용자들의 평가이력을 미네소타 대학 연구진이 모은 것을 활용한다.

ii. 현재 확보된 데이터셋의 종류는 세 가지이며, 세 가지 중 결과값이 가장 명확하게 산출되는 데이터셋을 최종보고서에서 활용한다.

iii. 세 데이터셋의 메타데이터

1. Ratings : 사용자가 한 영화에 부여한 평가점수로, 5 점 만점이다. 특정 사용자-영화-평가이력 정보에는 타임스탬프가 부여된다.
2. Users : 무비렌즈 서비스의 사용자로, 최소 20 개 영화에 평가를 남긴 사용자 중 연령, 성별, 직업에 편중되지 않고 무작위로 선정하였다. 각 사용자는 고유의 id 를 가지고 있으며, 그 외에 다른 정보는 없다.
3. Movies : 무비렌즈에서 서비스되는 영화들로, 각 영화는 고유한 id 와 복수의 장르를 가지고 있다. 장르는 어드벤처, 애니메이션, 아동, 코미디, SF, 로맨스, 드라마, 범죄, 호러, 미스테리, 스릴러 등으로 분류된다. 각 영화당 영화 데이터베이스인 imbd 와 tmbd 의 id 와도 연동이 되어있다. 해당 연결은 'links' 에서 확인할

수 있다.

4. Tags : 사용자가 영화에 부여한 free-text tagging 을 의미한다. 특정 사용자-영화-태그 정보에는 타임스탬프가 부여된다.
5. From/Until : 해당 데이터셋의 최초 평가이력과 최신 평가이력의 기간을 의미한다.
6. Generated : 해당 데이터셋이 수집 및 최초 가공된 최종 일자를 의미한다.
7. Last Update : 해당 데이터셋이 재가공 및 배포된 최종 일자를 의미한다.
8. DataSet2('ml-lastest')와 DataSet3('ml-latest-small')은 계속해서 업데이트 되는 데이터셋으로, 본 연구 이후에 데이터가 변동될 수 있으며, 그에 따라 산출값과 정확도도 변할 수 있다.

	DataSet1 ('ml-20m')	DataSet2 ('ml-latest')*	DataSet3 ('ml-latest-small')*
N(Ratings)	20000263	26024289	1000004
N(Users)	138493	270896	671
N(Movies)	27278	45843	9125
N(Tags)	465564	753170	1296
From	Jan 09, 1995	Jan 09, 1995	Jan 09, 1995
Until	Mar 31, 2015	Aug 04, 2017	Oct 16, 2016
Generated	Mar 31, 2015	Aug 04, 2017	Oct 17, 2016
Last Update	Oct 17, 2016	Aug 04 2017	Oct 17, 2016

표 1 : 데이터셋의 메타데이터

iv. 세 데이터셋의 공통적인 구조

userId	movieId	Rating	timestamp
1	2	3.5	1112486027
1	29	3.5	1112484676

1	32	3.5	1112484819
1	47	3.5	1112484727
…	…	…	…

표 2 : Ratings

userId	movieId	tag	timestamp
18	4141	Mark Waters	1240597180
65	208	dark hero	1368150078
65	353	dark hero	1368150079
65	521	noir thriller	1368149983
…	…	…	…

표 3: Tags

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
…	…	…

표 4 : Movies

movieId	imdbId	tmdbId
1	114709	862
2	113497	8844
3	113228	15602
4	114885	31357

…	…	…

표 5: Links

4. 구현 방식

A. 분석 기준

i. 입력데이터를 어떠한 방식으로 매트릭스화 할 것인지에 대해서는 여러 방법론이 있으나, 해당 연구에서는 가장 일반적인 프레임워크인 SVD(Singular-value decomposition) 방식을 활용한다. 이는 Missing Data 가 많은 User-item 데이터셋의 특성을 극복하고, 적절한 데이터를 채워넣기 위함이다.

ii. 데이터셋 전체를 동일한 규모로 나누어 서로 그 정확도를 검정하는 Cross-validation 방식을 활용하며, 일괄적으로 2 개, 3 개, 5 개, 10 개, 20 개의 데이터셋으로 나누어 검정하고 그 값을 비교한다.

iii. 정확도를 분석하는 방식으로는 RMSE 와 MAE 를 활용하며, 이에 대한 자세한 설명은 다음 항목에서 논한다.

B. RMSE 와 MAE : RMSE 와 MAE 는 Continuous variable 의 정확도를 분석할 때 가장 일반적으로 활용되는 메트릭스 구조이다. 많은 연구 결과물에서 그 정확도를 RMSE 나 R-squared 형식으로 표현한다.

i. RMSE(Root Mean Squared Error)

1. RMSE 의 정의는 다음과 같다.

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

2. 평균적인 에러값들을 계산하는 2 차 방정식이다.

ii. MAE(Mean Absolute Error)

1. MAE 의 정의는 다음과 같다.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

2. 예측값과 실제값의 차에 대한 절대값들을 평균낸 값이다.

iii. RMSE 와 MAE 비교

1. 분석하려고 하는 값들의 예측 에러를 평균 낸다는 점에서 두 수치는 매우 유사하다.
2. 둘 다 0 에서 무한 사이의 값을 가지고 있으며, 에러값이 양인지 음인지는 영향으르 주지 않는다.
3. RMSE 는 평균값을 내기 전에 각 에러를 제곱하기 때문에 큰 에러값이 전체적으로 큰 영향을 미치게 된다.
4. 그에 따라, 분산값이 증가할수록 RMSE 는 그 값이 더욱 커지는 반면, MAE 는 보다 안정적인 수치를 보여준다.

5. 구현 결과

A. Surprise

i. 첫번째 데이터 셋 ml-200m

1. Cross Validation (fold:2)

	RMSE	MAE
FOLD 1	0.8626	0.6624
FOLD 2	0.8611	0.6609
MEAN	0.8619	0.6616

표 6 : Surprise 의 ml-200m 분석 (fold=2)

2. Cross Validation (fold:3)

	RMSE	MAE
FOLD 1	0.8441	0.6470
FOLD 2	0.8455	0.6471
FOLD 3	0.8475	0.6492
MEAN	0.8457	0.6478

표 7 : : Surprise 의 ml-200m 분석 (fold=3)

3. Cross Validation (fold:5)

	RMSE	MAE
FOLD 1	0.8349	0.6387
FOLD 2	0.8351	0.6387
FOLD 3	0.8344	0.6383
FOLD 4	0.8302	0.6353
FOLD 5	0.8335	0.6379
MEAN	0.8336	0.6378

표 8 : : Surprise 의 ml-200m 분석 (fold=5)

4. Cross Validation (fold:10)

	RMSE	MAE
FOLD 1	0.8297	0.6330
FOLD 2	0.8254	0.6317
FOLD 3	0.8257	0.6305
FOLD 4	0.8200	0.6265
FOLD 5	0.8253	0.6319
FOLD 6	0.8282	0.6335
FOLD 7	0.8259	0.6319
FOLD 8	0.8261	0.6305
FOLD 9	0.8236	0.6295
FOLD 10	0.8256	0.6320
MEAN	0.8256	0.6311

표 9 : : Surprise 의 ml-200m 분석 (fold=10)

5. Cross Validation (fold : 20)

	RMSE	MAE
FOLD 1	0.8221	0.6285
FOLD 2	0.8186	0.6259
FOLD 3	0.8253	0.6318

FOLD 4	0.8255	0.6294
FOLD 5	0.8243	0.6303
FOLD 6	0.8215	0.6286
FOLD 7	0.8195	0.6277
FOLD 8	0.8199	0.6265
FOLD 9	0.8208	0.6286
FOLD 10	0.8209	0.6275
FOLD 11	0.8186	0.6272
FOLD 12	0.8226	0.6277
FOLD 13	0.8203	0.6291
FOLD 14	0.8255	0.6309
FOLD 15	0.8199	0.626
FOLD 16	0.8229	0.628
FOLD 17	0.8218	0.629
FOLD 18	0.8232	0.6276
FOLD 19	0.8188	0.6258
FOLD 20	0.8283	0.6333
MEAN	0.822	0.6285

표 10 : : Surprise 의 ml-200m 분석 (fold=20)

ii. 두번째 데이터셋 : ml–latest

1. Cross Validation (fold:2)

	RMSE	MAE
FOLD 1	0.8609	0.6579
FOLD 2	0.8625	0.6588
MEAN	0.8617	0.6584

표 11 : : Surprise 의 ml-latest 분석 (fold=2)

2. Cross Validation (fold:3)

	RMSE	MAE
FOLD 1	0.8463	0.6452
FOLD 2	0.8447	0.6441
FOLD 3	0.8435	0.6433
MEAN	0.8448	0.6442

표 12 : : Surprise 의 ml-latest 분석 (fold=3)

3. Cross Validation (fold:5)

	RMSE	MAE
FOLD 1	0.8319	0.6335
FOLD 2	0.8335	0.6342
FOLD 3	0.8311	0.633
FOLD 4	0.8349	0.6368
FOLD 5	0.8346	0.6359
MEAN	0.8332	0.6347

표 13 : : Surprise 의 ml-latest 분석 (fold=5)

4. Cross Validation (fold:10)

	RMSE	MAE
FOLD 1	0.8211	0.6254
FOLD 2	0.8242	0.63
FOLD 3	0.8241	0.6278
FOLD 4	0.8272	0.6277
FOLD 5	0.8228	0.6272
FOLD 6	0.8273	0.6283
FOLD 7	0.8275	0.6293
FOLD 8	0.826	0.6281
FOLD 9	0.8268	0.629

FOLD 10	0.8248	0.6287
MEAN	0.8252	0.6281

표 14 : : Surprise 의 ml-latest 분석 (fold=10)

5. Cross Validation (fold : 20)

	RMSE	MAE
FOLD 1	0.8184	0.6223
FOLD 2	0.8254	0.6285
FOLD 3	0.8189	0.6239
FOLD 4	0.8296	0.6299
FOLD 5	0.8226	0.6243
FOLD 6	0.8231	0.6257
FOLD 7	0.8195	0.625
FOLD 8	0.8209	0.6243
FOLD 9	0.824	0.6269
FOLD 10	0.818	0.6227
FOLD 11	0.8215	0.6251
FOLD 12	0.8211	0.6249
FOLD 13	0.8223	0.624
FOLD 14	0.8215	0.624
FOLD 15	0.8236	0.6264
FOLD 16	0.8241	0.6281
FOLD 17	0.8224	0.6236
FOLD 18	0.818	0.6237
FOLD 19	0.8211	0.6234
FOLD 20	0.8223	0.6262
MEAN	0.8219	0.6252

표 15 : : Surprise 의 ml-latest 분석 (fold=20)

iii. 세번째 데이터셋 : ml-latest-small

1. Cross Validation (fold:2)

	RMSE	MAE
FOLD 1	0.9140	0.7057
FOLD 2	0.9083	0.7019
MEAN	0.9111	0.7038

표 16 : : Surprise 의 ml-latest-small 분석 (fold=2)

2. Cross Validation (fold:3)

	RMSE	MAE
FOLD 1	0.9046	0.6962
FOLD 2	0.9032	0.6973
FOLD 3	0.9024	0.6967
MEAN	0.9034	0.6967

표 17 : : Surprise 의 ml-latest-small 분석 (fold=3)

A. Cross Validation (fold:5)

	RMSE	MAE
FOLD 1	0.9021	0.6904
FOLD 2	0.9106	0.6988
FOLD 3	0.8898	0.6841
FOLD 4	0.8986	0.6919
FOLD 5	0.8876	0.6875
MEAN	0.8977	0.6905

표 18 : Surprise 의 ml-latest-small 분석 (fold=5)

3. Cross Validation (fold:10)

	RMSE	MAE
FOLD 1	0.8758	0.6758
FOLD 2	0.9095	0.7015
FOLD 3	0.8804	0.6742
FOLD 4	0.8759	0.6736

FOLD 5	0.8869	0.6859
FOLD 6	0.8983	0.6926
FOLD 7	0.9096	0.6984
FOLD 8	0.8929	0.6835
FOLD 9	0.9085	0.6988
FOLD 10	0.8941	0.6898
MEAN	0.8932	0.6874

표 19 : : Surprise 의 ml-latest-small 분석 (fold=10)

4. Cross Validation (fold : 20)

	RMSE	MAE
FOLD 1	0.8913	0.6833
FOLD 2	0.885	0.6827
FOLD 3	0.8963	0.6904
FOLD 4	0.8831	0.6801
FOLD 5	0.8744	0.6734
FOLD 6	0.9031	0.6941
FOLD 7	0.8875	0.6796
FOLD 8	0.8882	0.6846
FOLD 9	0.886	0.6808
FOLD 10	0.8894	0.6805
FOLD 11	0.9191	0.7068
FOLD 12	0.8989	0.6912
FOLD 13	0.877	0.6716
FOLD 14	0.8875	0.6787
FOLD 15	0.8904	0.6861
FOLD 16	0.8808	0.6795
FOLD 17	0.8776	0.6737
FOLD 18	0.8877	0.68

FOLD 19	0.8978	0.6939
FOLD 20	0.8906	0.6893
MEAN	0.8896	0.684

표 20 : : Surprise 의 ml-latest-small 분석 (fold=20)

B. Python–recsys

i. 첫번째 데이터셋 : ml–200m

1. Cross Validation (fold:2)

	RMSE	MAE
FOLD 1	0.8667	0.6664
FOLD 2	0.868	0.665
MEAN	0.8662	0.6657

표 21 : : Python-recsys 의 ml-200m 분석 (fold=2)

2. Cross Validation (fold:3)

	RMSE	MAE
FOLD 1	0.8606	0.6616
FOLD 2	0.8621	0.6619
FOLD 3	0.8596	0.6606
MEAN	0.8608	0.6613

표 22 : Python-recsys 의 ml-200m 분석 (fold=3)

3. Cross Validation (fold:5)

	RMSE	MAE
FOLD 1	0.8569	0.6587
FOLD 2	0.8556	0.657
FOLD 3	0.857	0.6582
FOLD 4	0.8602	0.6603
FOLD 5	0.8597	0.66
MEAN	0.8579	0.6588

표 23 : Python-recsys 의 ml-200m 분석 (fold=5)

4. Cross Validation (fold:10)

	RMSE	MAE
FOLD 1	0.8550	0.6565
FOLD 2	0.8525	0.6552
FOLD 3	0.8551	0.6571
FOLD 4	0.8581	0.6596
FOLD 5	0.8562	0.6577
FOLD 6	0.8661	0.6581
FOLD 7	0.8558	0.6565
FOLD 8	0.8581	0.6583
FOLD 9	0.8545	0.6568
FOLD 10	0.8571	0.6579
MEAN	0.8559	0.6574

표 24 : Python-recsys 의 ml-200m 분석 (fold=10)

5. Cross Validation (fold : 20)

	RMSE	MAE
FOLD 1	0.8586	0.6612
FOLD 2	0.8577	0.6596
FOLD 3	0.8570	0.6567
FOLD 4	0.8552	0.6578
FOLD 5	0.8574	0.6584
FOLD 6	0.8506	0.6537
FOLD 7	0.8490	0.6525
FOLD 8	0.8527	0.6545
FOLD 9	0.8554	0.6562
FOLD 10	0.8543	0.6561

FOLD 11	0.8548	0.6565
FOLD 12	0.8564	0.6572
FOLD 13	0.8564	0.6581
FOLD 14	0.8516	0.6545
FOLD 15	0.8519	0.6551
FOLD 16	0.8593	0.6589
FOLD 17	0.8552	0.6556
FOLD 18	0.8544	0.6559
FOLD 19	0.862	0.6598
FOLD 20	0.8543	0.6554
MEAN	0.8552	0.6567

표 25 : Python-recsys 의 ml-200m 분석 (fold=20)

ii. 두번째 데이터셋 : ml-latest

1. Cross Validation (fold:2)

	RMSE	MAE
FOLD 1	0.8657	0.6622
FOLD 2	0.8645	0.6612
MEAN	0.8651	0.6617

표 26 : Python-recsys 의 ml-latest 분석 (fold=2)

2. Cross Validation (fold:3)

	RMSE	MAE
FOLD 1	0.8584	0.6562
FOLD 2	0.8604	0.6577
FOLD 3	0.8603	0.6571
MEAN	0.8597	0.6570

표 27 : Python-recsys 의 ml-latest 분석 (fold=3)

3. Cross Validation (fold:5)

	RMSE	MAE
FOLD 1	0.8540	0.6525
FOLD 2	0.8563	0.6545
FOLD 3	0.8565	0.6544
FOLD 4	0.8591	0.6552
FOLD 5	0.8557	0.6533
MEAN	0.8563	0.6540

표 28 : Python-recsys 의 ml-latest 분석 (fold=5)

4. Cross Validation (fold:10)

	RMSE	MAE
FOLD 1	0.8528	0.6519
FOLD 2	0.8529	0.6522
FOLD 3	0.8559	0.6539
FOLD 4	0.8569	0.6546
FOLD 5	0.8543	0.6530
FOLD 6	0.8524	0.6510
FOLD 7	0.8552	0.6535
FOLD 8	0.8539	0.6513
FOLD 9	0.8552	0.6527
FOLD 10	0.8566	0.6542
MEAN	0.8546	0.6528

표 29 : Python-recsys 의 ml-latest 분석 (fold=10)

5. Cross Validation (fold : 20)

	RMSE	MAE
FOLD 1	0.8566	0.6535
FOLD 2	0.8560	0.6527

FOLD 3	0.8512	0.6494
FOLD 4	0.8523	0.6521
FOLD 5	0.8574	0.6534
FOLD 6	0.8520	0.6494
FOLD 7	0.8513	0.6508
FOLD 8	0.8479	0.6480
FOLD 9	0.8526	0.6503
FOLD 10	0.8603	0.6559
FOLD 11	0.8555	0.6517
FOLD 12	0.8514	0.6509
FOLD 13	0.8576	0.6548
FOLD 14	0.8523	0.6495
FOLD 15	0.8554	0.6528
FOLD 16	0.8544	0.6535
FOLD 17	0.853	0.6515
FOLD 18	0.8529	0.6498
FOLD 19	0.8556	0.6539
FOLD 20	0.8478	0.649
MEAN	0.8537	0.6516

표 30 : Python-recsys 의 ml-latest 분석 (fold=20)

iii. 세번째 데이터셋 : ml-latest-small

1. Cross Validation (fold:2)

	RMSE	MAE
FOLD 1	0.9058	0.6978
FOLD 2	0.9122	0.7056
MEAN	0.9090	0.7017

표 31 : Python-recsys 의 ml-latest-small 분석 (fold=2)

2. Cross Validation (fold:3)

	RMSE	MAE
FOLD 1	0.8606	0.6616
FOLD 2	0.8621	0.6619
FOLD 3	0.8596	0.6606
MEAN	0.8608	0.6613

표 32 : Python-recsys 의 ml-latest-small 분석 (fold=3)

3. Cross Validation (fold:5)

	RMSE	MAE
FOLD 1	0.8923	0.6862
FOLD 2	0.9062	0.6970
FOLD 3	0.8945	0.6895
FOLD 4	0.8959	0.6919
FOLD 5	0.9029	0.6940
MEAN	0.8984	0.6917

표 33 : : Python-recsys 의 ml-latest-small 분석 (fold=5)

4. Cross Validation (fold:10)

	RMSE	MAE
FOLD 1	0.8948	0.6939
FOLD 2	0.9018	0.6888
FOLD 3	0.8929	0.6864
FOLD 4	0.8932	0.6874
FOLD 5	0.8888	0.6837
FOLD 6	0.8905	0.6875
FOLD 7	0.9058	0.6988
FOLD 8	0.9057	0.6956
FOLD 9	0.9040	0.6923
FOLD 10	0.8884	0.6845

MEAN	0.8966	0.6899

표 34 : : Python-recsys 의 ml-latest-small 분석 (fold=10)

5. Cross Validation (fold : 20)

	RMSE	MAE
FOLD 1	0.8854	0.6862
FOLD 2	0.8886	0.6872
FOLD 3	0.8845	0.6803
FOLD 4	0.8940	0.6874
FOLD 5	0.8834	0.6861
FOLD 6	0.8941	0.6935
FOLD 7	0.8866	0.6826
FOLD 8	0.8892	0.6863
FOLD 9	0.8906	0.6882
FOLD 10	0.8774	0.6748
FOLD 11	0.8899	0.6821
FOLD 12	0.9010	0.6917
FOLD 13	0.8909	0.6840
FOLD 14	0.9078	0.6966
FOLD 15	0.8850	0.6780
FOLD 16	0.9164	0.7016
FOLD 17	0.9100	0.6999
FOLD 18	0.9101	0.7013
FOLD 19	0.8995	0.6919
FOLD 20	0.9085	0.7020
MEAN	0.8946	0.6891

표 35 : : Python-recsys 의 ml-latest-small 분석 (fold=20)

6. 결과 분석

A. 결과값 비교

i. RMSE MAE 평균값

A. Surprise

Cross validation		ml-20m	ml-latest	ml-latest-small
Nfolds=2	RMSE	0.8619	0.8617	0.9111
	MAE	0.6616	0.6584	0.7038
Nfolds=3	RMSE	0.8457	0.8448	0.9034
	MAE	0.6478	0.6442	0.6967
Nfolds=5	RMSE	0.8336	0.8332	0.8977
	MAE	0.6378	0.6347	0.6905
Nfoods=10	RMSE	0.8256	0.8252	0.8932
	MAE	0.6311	0.6281	0.6874
Nfolds=20	RMSE	0.8220	0.8219	0.8996
	MAE	0.6285	0.6252	0.6840

표 36 : Surprise 의 정확도 비교

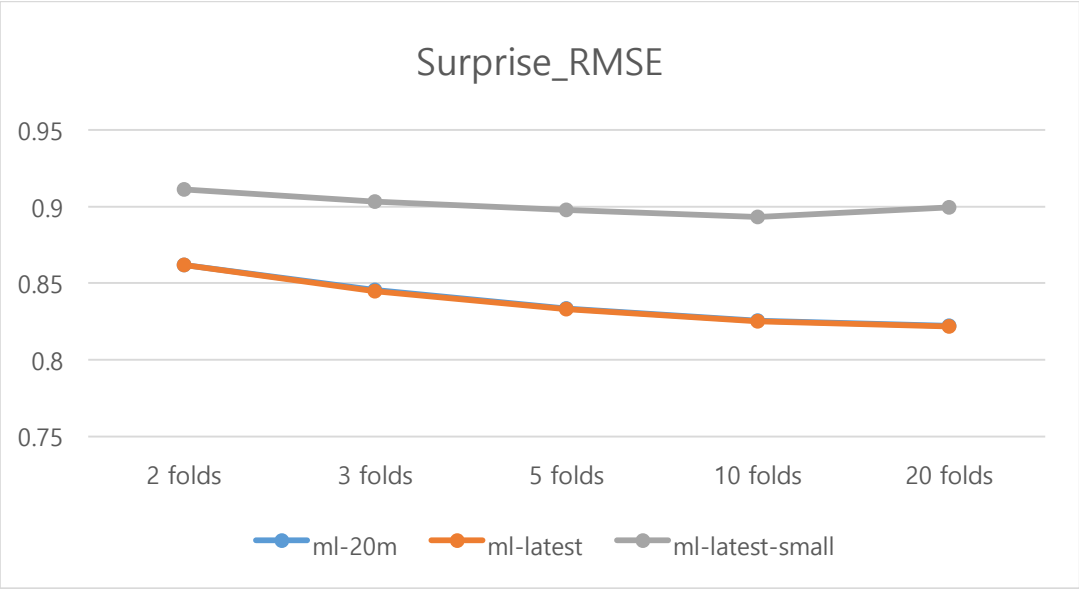
ii. Python-recsys

Cross validation		ml-20m	ml-latest	ml-latest-small
Nfolds=2	RMSE	0.8662	0.8651	0.9090
	MAE	0.6657	0.6617	0.7017
Nfolds=3	RMSE	0.8608	0.8597	0.9023
	MAE	0.6613	0.6571	0.6954
Nfolds=5	RMSE	0.8579	0.8563	0.8984
	MAE	0.6588	0.6540	0.6917
Nfoods=10	RMSE	0.8559	0.8546	0.8966
	MAE	0.6574	0.6528	0.6899
Nfolds=20	RMSE	0.8552	0.8537	0.8946
	MAE	0.6567	0.6516	0.6891

표 37 : Python-recsys 의 정확도 비교

iii. Surprise 의 결과치 분석

1. RMSE



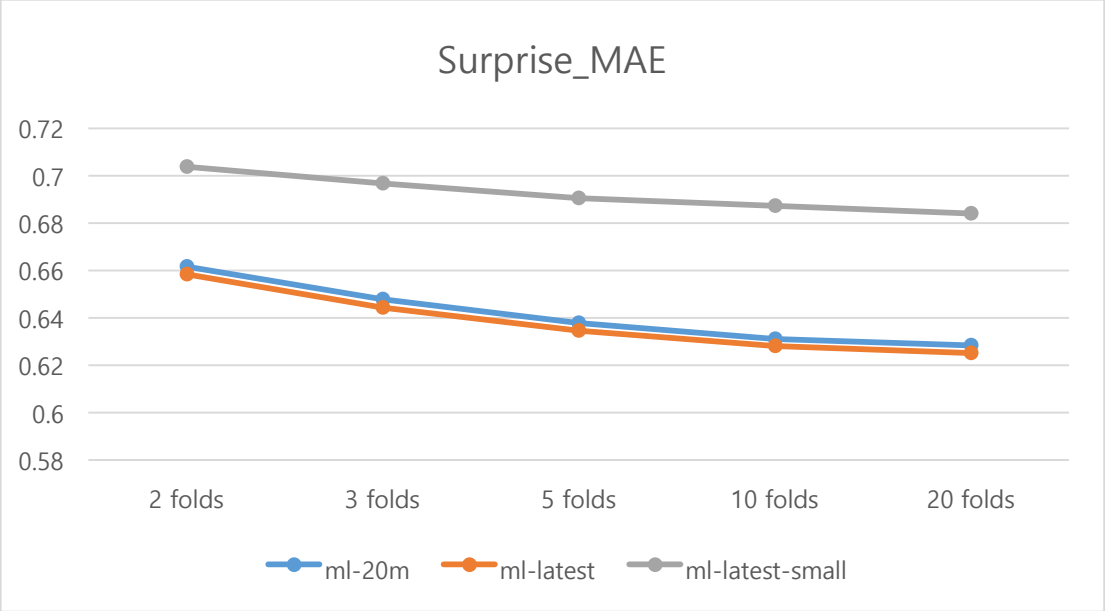
A. 동일 데이터 내에서 fold 값이 커질수록 그 정확도가 높아진다.

B. 데이터셋의 규모가 커질수록 그 정확도가 높아진다.

C. 에러가 0.8 에서 0.95 사이로 분포했다.

D. 데이터셋의 규모가 유사한 'ml-20m'과 'ml-latest'의 정확도가 각 데이터 분할 값에 따라 거의 유사하게 나타난 것으로 보아, Surprise 의 RMSE 는 분석 데이터 규모와 강력한 상관관계가 있음을 알 수 있다.

2. MAE



A. 동일 데이터 내에서 Cross Validation 을 위한 데이터 분할이 많아질수록 그 정확도가 높아진다.

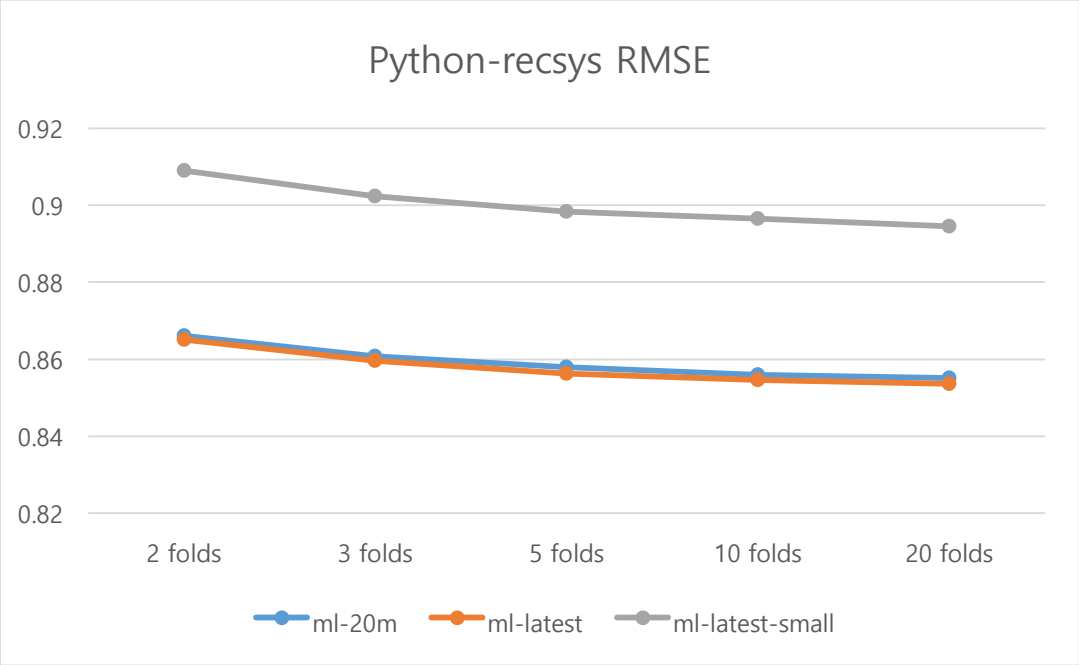
B. 데이터셋의 규모가 커질수록 그 정확도가 높아진다.

C. 정확도는 0.64 에서 0.72 사이에서 분포했다.

D. MAE 역시 RMSE 와 마찬가지로 분석 데이터의 사이즈가 유사하다면 그 정확도 역시 매우 유사하게 산출된다.

iv. Python-recsys 의 결과치 분석

1. RMSE



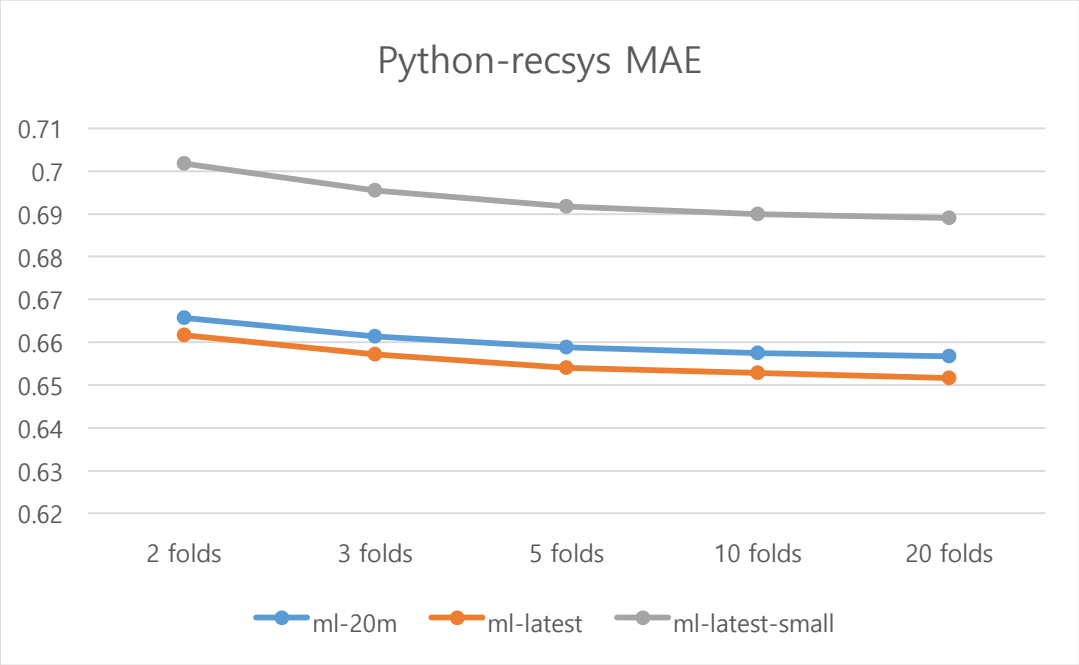
A. 동일 데이터 내에서 fold 값이 커질수록 그 정확도가 높아진다.

B. 데이터셋의 규모가 커질수록 그 정확도가 높아진다.

C. 에러가 0.84 에서 0.92 사이로 분포했다.

D. 데이터셋의 규모가 유사한 'ml-20m'과 'ml-latest'의 정확도가 각 데이터 분할 값에 따라 거의 유사하게 나타난 것으로 보아, Python-recsys 의 RMSE 는 분석 데이터 규모와 강력한 상관관계가 있음을 알 수 있다.

2. MAE



A. 동일 데이터 내에서 Cross Validation 을 위한 데이터 분할이 많아질수록 그 정확도가 높아진다.

B. 데이터셋의 규모가 커질수록 그 정확도가 높아진다.

C. 에러는 0.65 에서 0.71 사이에서 분포했다.

D. Surprise 의 MAE 에 비해 데이터 사이즈의 유사도에 따른 정확도 유사도의 관계성이 낮다.

v. 동일 Cross-validation 기준 Surprise 와 Python-recsys 양자비교

1. 데이터셋 규모의 차이

A. 규모가 큰 'ml-20m'이나 'ml-latest' 데이터 셋에서 항상 Surprise 가 Python-recsys 에 비하여 더 좋은 정확도를 나타냈다. 따라서, 데이터 셋의 규모가 20 만개 이상으로 큰 경우에는, Surprise 를 활용하는 것이 더 바람직하다.

B. Cross-validation 을 위해 데이터를 여러 사이즈로 분할했을 때, Surprise 는 분할의 영향을 상대적으로 더 많이 받았다. 그러나 Python-recsys 는 그 차이가 적어 보다 안정적인 정확도를 나타냈다.

2. RMSE 와 MAE 에 따른 차이

A. 모든 경우에 RMSE 가 MAE 보다 그 값이 컸다. 이는 RMSE 와 MAE 의 기본적 특성에서 기반하는것으로, 항상 RMSE 는 MAE 보다 크며, 에러들이 모두 같은 수준의 양을 가졌을때만 두 값이 동일하다.

7. 결론

본 연구는 협업 필터링을 활용한 추천 시스템을 제공하는 두 오픈 소스 파이썬 라이브러리 Surprise 와 Python-recsys 를 비교한다. Surprise 와 Python-recsys 모두 동일하게 데이터셋의 규모가 커질수록, 그리고 Cross-validation 을 위한 데이터 분할의 횟수가 커질수록 정확도가 높아졌다. Surprise 는 데이터셋의 규모가 커지거나 작아지더라도 그 정확도의 차이가 상대적으로 작았다. 뿐만 아니라, 데이터 분할의 횟수의 변화에 낮아지지만 상대적으로 안정적인 정확도를 보여주었다. 반면 Python-recsys 는 데이터셋의 규모가 커질수록 그 정확도의 변화가 크게 상승했다. 데이터 분할의 횟수에도 비교적 민감하게 반응했다. 하지만 데이터셋의 규모가 2 천만개 이하로 작을 경우에는 Python-recsys 에 비해 더 높은 정확도를 보였다. 따라서, 데이터셋이 계속해서 증가하는 방향으로 변화하고 있다면, Python-recsys 를 사용해야 점점 정확도가 개선된다. 반대로 데이터셋이 이미 고정되어있으며 작은 규모라면, Surprise 를 활용하는 것이 더 높은 정확도를 보여준다.