DMT HW2: Recommendation systems

Alessandro Quattrociochi - 1609286
Tansel Simsek - 1942297

Part 1.1

Recommendation Algorithms Performances on the first dataset

Evaluating RMSE of algorithm NormalPredictor on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.5118	1.5162	1.5136	1.5117	1.5158	1.5138	0.0019
Fit time	0.28	0.46	0.66	0.58	0.45	0.49	0.13
Test time	0.80	0.89	0.68	0.61	0.44	0.68	0.16

1. Normal Predictor

Evaluating RMSE of algorithm BaselineOnly on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9167	0.9176	0.9094	0.9215	0.9163	0.9163	0.0039
Fit time	2.30	2.44	2.55	2.48	1.37	2.23	0.44
Test time	0.56	0.54	0.49	0.43	0.29	0.46	0.10

Out[46]:

2. BaselineOnly

Evaluating RMSE of algorithm CoClustering on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9330	0.9448	0.9361	0.9559	0.9331	0.9406	0.0088
Fit time	4.24	4.45	4.39	4.27	2.33	3.94	0.81
Test time	0.62	0.64	0.60	0.53	0.30	0.53	0.12

3. CoClustering

Evaluating RMSE of algorithm SlopeOne on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9241	0.9240	0.9185	0.9279	0.9221	0.9233	0.0031
Fit time	6.59	6.67	6.49	5.98	3.50	5.84	1.20
Test time	30.21	29.26	28.50	29.20	12.30	25.90	6.82

4. SlopeOne

Evaluating RMSE of algorithm NMF on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9410	0.9338	0.9312	0.9422	0.9344	0.9365	0.0043
Fit time	25.18	25.61	24.86	25.39	15.61	23.33	3.87
Test time	0.64	0.64	0.67	0.46	0.91	0.66	0.14

5. NMF

Evaluating RMSE of algorithm KNNBasic on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.0048	1.0114	1.0080	1.0129	1.0053	1.0069	0.0047
Fit time	19.53	19.95	19.22	18.84	7.13	16.93	4.91
Test time	36.97	37.92	36.79	36.99	14.74	32.68	8.98

6. KNNBasic

Evaluating RMSE of algorithm KNNWithMeans on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9190	0.9188	0.9112	0.9206	0.9151	0.9169	0.0034
Fit time	18.55	18.73	18.95	18.83	7.15	16.44	4.65
Test time	39.24	39.68	39.17	39.82	13.98	34.38	10.20

7. KNNWhitMeans

Evaluating RMSE of algorithm KNNWithZScore on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9215	0.9216	0.9136	0.9229	0.9175	0.9194	0.0035
Fit time	15.62	16.08	16.10	16.19	7.30	14.26	3.48
Test time	39.23	39.13	39.98	38.88	15.80	34.60	9.41

8. KNNWhitZScore

Evaluating RMSE of algorithm KNNBaseline on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9157	0.9184	0.9077	0.9199	0.9126	0.9148	0.0044
Fit time	16.00	16.69	16.45	16.45	7.23	14.56	3.67
Test time	41.09	42.14	41.12	41.11	15.75	36.24	10.25

9. KNNBaseline

Evaluating RMSE of algorithm SVD on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9399	0.9398	0.9326	0.9405	0.9352	0.9374	0.0030
Fit time	69.61	69.13	68.92	69.02	30.06	61.35	15.64
Test time	1.01	0.99	1.03	1.01	0.47	0.90	0.22

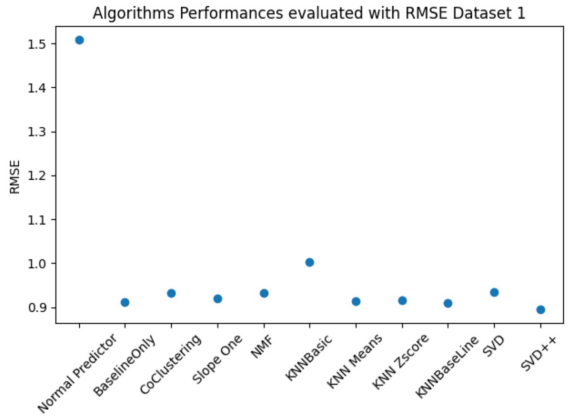
10.SVD

Evaluating RMSE of algorithm SVDpp on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.8991	0.8998	0.8928	0.9030	0.8987	0.8987	0.0033
Fit time	6189.90	6213.03	6219.05	6233.61	2654.24	5501.97	1423.93
Test time	31.51	32.80	31.71	23.53	15.46	27.80	6.66

11.SVDpp

Algorithms Ranking Dataset 1



	Algorithms	RMSE	std	RMSE-std
10	SVD++	0.8987	0.0033	0.8954
8	KNNBaseLine	0.9148	0.0044	0.9104
1	BaselineOnly	0.9163	0.0039	0.9124
6	KNN Means	0.9169	0.0034	0.9135
7	KNN Zscore	0.9194	0.0035	0.9159
3	Slope One	0.9233	0.0031	0.9202
2	CoClustering	0.9406	0.0088	0.9318
4	NMF	0.9365	0.0043	0.9322
9	SVD	0.9374	0.0030	0.9344
5	KNNBasic	1.0069	0.0047	1.0022
0	Normal Predictor	1.5138	0.0054	1.5084

Figure 5. Recommendation Algorithms Performances

Recommendation algorithms performances on the second dataset

Evaluating RMSE of algorithm NormalPredictor on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	3.2364	3.1995	3.2412	3.2464	3.1822	3.2211	0.0255
Fit time	0.03	0.05	0.06	0.04	0.04	0.04	0.01
Test time	0.06	0.06	0.06	0.05	0.03	0.05	0.01

1. Normal Predictor

Evaluating RMSE of algorithm BaselineOnly on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.9850	1.9770	1.9739	2.0015	1.9851	1.9845	0.0096
Fit time	0.32	0.26	0.25	0.21	0.11	0.23	0.07
Test time	0.06	0.05	0.04	0.06	0.02	0.05	0.01

2. BaselineOnly

Evaluating RMSE of algorithm CoClustering on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	2.0362	2.0031	2.0413	2.0582	2.0348	2.0347	0.0179
Fit time	0.46	0.44	0.46	0.41	0.22	0.40	0.09
Test time	0.06	0.06	0.06	0.07	0.02	0.06	0.02

3. CoClustering

Evaluating RMSE of algorithm SlopeOne on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.9835	1.9814	1.9779	1.9989	1.9697	1.9823	0.0096
Fit time	0.06	0.05	0.06	0.04	0.05	0.05	0.01
Test time	0.36	0.37	0.37	0.36	0.15	0.32	0.08

4. SlopeOne

Evaluating RMSE of algorithm NMF on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.9641	1.9779	1.9731	2.0309	2.0172	1.9927	0.0264
Fit time	1.98	2.25	2.13	1.98	1.19	1.91	0.37
Test time	0.05	0.03	0.04	0.08	0.02	0.05	0.02

5. NMF

Evaluating RMSE of algorithm KNNBasic on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	2.3865	2.3554	2.3727	2.4226	2.3854	2.3845	0.0221
Fit time	0.03	0.03	0.05	0.05	0.05	0.04	0.01
Test time	0.57	0.58	0.57	0.56	0.26	0.51	0.12

6. KNNBasic

Evaluating RMSE of algorithm KNNWithMeans on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.9886	1.9622	1.9727	1.9975	1.9693	1.9781	0.0130
Fit time	0.05	0.06	0.06	0.08	0.05	0.06	0.01
Test time	0.60	0.68	0.63	0.61	0.32	0.57	0.13

7. KNNWhitMeans

Evaluating RMSE of algorithm KNNWithZScore on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	2.0029	1.9724	1.9792	1.9997	1.9797	1.9868	0.0122
Fit time	0.11	0.16	0.17	0.15	0.08	0.13	0.03
Test time	1.41	1.37	1.43	1.41	0.35	1.19	0.42

8. KNNWhitZScore

Evaluating RMSE of algorithm KNNBaseline on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.9924	1.9776	1.9801	2.0128	1.9864	1.9898	0.0126
Fit time	0.15	0.10	0.07	0.06	0.04	0.08	0.04
Test time	0.89	0.92	0.91	0.85	0.38	0.79	0.21

9. KNNBaseline

Evaluating RMSE of algorithm SVD on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.9151	1.9134	1.8617	1.9243	1.8864	1.9002	0.0230
Fit time	5.31	5.22	5.20	5.27	2.41	4.68	1.14
Test time	0.07	0.07	0.07	0.07	0.03	0.06	0.02

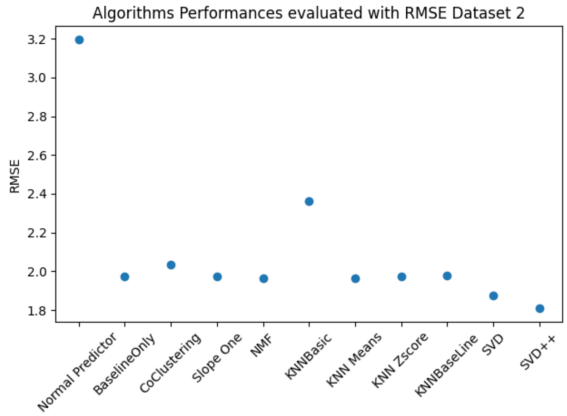
10.SVD

Evaluating RMSE of algorithm SVDpp on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.8289	1.8278	1.8105	1.8574	1.8112	1.8271	0.0170
Fit time	61.01	61.82	61.14	61.56	33.59	55.82	11.12
Test time	0.44	0.33	0.48	0.40	0.25	0.38	0.08

11.SVDpp

Algorithms Ranking dataset 2



	Algorithms	RMSE	std	RMSE-std
10	SVD++	1.8271	0.01700	1.81010
9	SVD	1.9002	0.02300	1.87720
6	KNN Means	1.9781	0.01300	1.96510
4	NMF	1.9927	0.02640	1.96630
3	Slope One	1.9823	0.00960	1.97270
7	KNN Zscore	1.9868	0.01220	1.97460
1	BaselineOnly	1.9845	0.00960	1.97490
8	KNNBaseLine	1.9898	0.01260	1.97720
2	CoClustering	2.0347	0.00179	2.03291
5	KNNBasic	2.3845	0.02210	2.36240
0	Normal Predictor	3.2211	0.02550	3.19560

Figure 10. Recommendation Algorithms Performances

CPU cores settings

For all the simulations, 4 cores were used in the *cross_validate* function, imposing the variable **n_jobs** =4.

```
cross_validate(current_algo, data, measures=['RMSE'], cv=kf,n_jobs=4,
    verbose=True)
```

Part 1.2

Dataset 1: Random-Search-Cross-Validation process for tuning the hyper-parameter of the KNNBaseline algorithm

```
search_params ={"k": [20,25,30,35,40,45,50],
                "min_k": [1,3,5],
                "sim_options": {
                    "name": ["cosine","pearson_baseline"],
                    "user_based":[True, False],
                    "min_support": [2,3,4]
                },
                "bsl_options":{
                    'method': ["sgd","als"],
                    'learning_rate': [0.001,0.005,0.01],
                    'n_epochs': [10,20,50],
                    'reg': [0.01,0.02,0.03],
                }

}
```

Hyper Parameters	Tested	Selected
$k1$	$[20, 25, 30, 35, 40, 45, 50]$	35
min_k	$[1, 3, 5]$	1
$name$	$[$"$cosine$","$pearson\_baseline$"$]$	"$pearson\_baseline$"
$user\_based$	$[True, False]$	$False$
$min\_support$	$[2, 3, 4]$	3
$method$	$[$"sgd","als"$]$	"sgd"
$learning\_rate$	$[0.001, 0.005, 0.01]$	0.005
n_epochs	$[10, 20, 50]$	50
reg	$[0.01, 0.02, 0.03]$	0.02

Value of **RMSE** according to the selected parameters: 0.8852871686079228.
The whole running of the code took 6.8 min.

Dataset 1: Grid-Search-Cross-Validation process for tuning the hyper-parameter of the SVD algorithm

```
param_grid = {
              'n_factors':[98,100,102,104],
              'n_epochs': [10,20,50],
              'lr_all': [ 0.4, 0.01,0.5],
              'reg_all': [0.2,0.1,0.7,0.9]}
```

Parameters	Tested	Selected
$n\_factors$	$[98, 100, 102, 104]$	100
n_epochs	$[10, 20, 50]$	50
lr_all	$[0.4, 0.01, 0.5]$	0.01
reg_all	$[0.2, 0.1, 0.7, 0.9]$	0.1

Value of **RMSE** according to the selected parameters: 0.8824775731847134.
The whole running of the code took 91.00 min

Dataset 2: Random-Search-Cross-Validation process for tuning the hyper-parameter of the KNNBaseline algorithm

```
search_params ={"k": [20,25,30,35,40,45,50],
                "min_k": [1,3,5],
                "sim_options": {
                    "name": ["cosine","pearson_baseline"],
                    "user_based":[True, False],
                    "min_support": [2,3,4]
                },
                "bsl_options":{
                    'method': ["sgd","als"],
                    'learning_rate': [0.001,0.005,0.01],
                    'n_epochs': [50,20],
                    'reg': [0.01,0.02,0.03],
                }

}
```

Hyper Parameters	Tested	Selected
$k1$	$[20, 25, 30, 35, 40, 45, 50]$	20
min_k	$[1, 3, 5]$	1
$sim\_options$	$[20, 25, 30, 35, 40, 45, 50]$	$scoring.TF\_IDF()$
$name$	$[$"$cosine$","$pearson\_baseline$"$]$	"$pearson\_baseline$"
$user\_based$	$[True, False]$	$True$
$min\_support$	$[2, 3, 4]$	3
$method$	$[$"sgd","als"$]$	"als"
$learning\_rate$	$[0.001, 0.005, 0.01]$	0.005
n_epochs	$[50, 20]$	20
reg	$[0.01, 0.02, 0.03]$	0.01

Value of **RMSE** according to the selected parameters: 1.8418355193277776.
The whole running of the code took 24s.

Dataset 2: Grid-Search-Cross-Validation process for tuning the hyper-parameter of the SVD algorithm

```
param_grid = {
              'n_factors': [98,100,102,104]
              'n_epochs': [10,20,50],
              'lr_all': [ 0.4, 0.01,0.5,0.05,0.001],
              'reg_all': [0.2,0.1,0.05,0.1,0.9]}
```

Parameters	Tested	Selected
$n\_factors$	$[98, 100, 102, 104]$	100
n_epochs	$[10, 20, 50]$	20
lr_all	$[0.4, 0.01, 0.5, 0.05, 0.001]$	0.01
reg_all	$[0.2, 0.1, 0.05, 0.1, 0.9]$	0.1

Value of **RMSE** according to the selected parameters: 1.8406345188022115.
The whole running of the code took 9.3 min.

Part 2.1

- The set of Pokemon provided in output by the Topic-Specific-PageRank procedure as the best team of 6 Pokemon using Set_A as input.
  'Pikachu', 'Gengar', 'Dragonite', "Sirfetch'd", 'Kingdra', 'Lucario'

- The set of Pokemon provided in output by the Topic-Specific-PageRank procedure as the best team of 6 Pokemon using Set_B as input.
  'Venusaur', 'Charizard', 'Blastoise', 'Dusclops', 'Urshifu', 'Torkoal'

- The set of Pokemon provided in output by the Topic-Specific-PageRank procedure as the best team of 6 Pokemon using Set_C as input.
  'Dracovish', 'Milotic', 'Whimsicott', 'Excadrill', 'Tyranitar', 'Cinderace'

- The set of Pokemon provided in output by the Topic-Specific-PageRank procedure as the best team of 6 Pokemon using "Charizard" as input.
  'Charizard', 'Torkoal', 'Venusaur', 'Clefairy', 'Groudon', 'Grimmsnarl'

- The set of Pokemon provided in output by the Topic-Specific-PageRank procedure as the best team of 6 Pokemon using "Venusaur" as input.
  'Venusaur', 'Torkoal', 'Dusclops', 'Charizard', 'Porygon2', 'Stakataka'

- The set of Pokemon provided in output by the Topic-Specific-PageRank procedure as the best team of 6 Pokemon using "Kingdra" as input.
  'Kingdra', 'Politoed', 'Kyogre', 'Tornadus', 'Togedemaru', 'Tsareena'

- The set of Pokemon provided in output by the Topic-Specific-PageRank procedure as the best team of 6 Pokemon using set(["Charizard", "Venusaur"]) as input.
  'Venusaur', 'Charizard', 'Torkoal', 'Dusclops', 'Stakataka', 'Groudon'

- The set of Pokemon provided in output by the Topic-Specific-PageRank procedure as the best team of 6 Pokemon using set(["Charizard", "Kingdra"]) as input.
  'Kingdra', 'Charizard', 'Raichu', 'Bronzong', 'Sableye', 'Politoed'

- The set of Pokemon provided in output by the Topic-Specific-PageRank procedure as the best team of 6 Pokemon using set(["Venusaur", "Kingdra"]) as input.
  'Kingdra', 'Venusaur', 'Tapu Koko', 'Bronzong', 'Corviknight', 'Politoed'

- The number of team members inside the Team(Charizard, Venusaur) that are neither in Team(Charizard) nor in Team(Venusaur)
  0

- The number of team members inside the Team(Charizard, Kingdra) that are neither in Team(Charizard) nor in Team(Kingdra)
  3 ('Sableye', 'Bronzong', 'Raichu')

- The number of team members inside the Team(Venusaur, Kingdra) that are neither in Team(Venusaur) nor in Team(Kingdra)
  3 ('Corviknight', 'Bronzong', 'Tapu Koko')

Pokemon	Community frequency
Kartana	178
Incineroar	178
Volcarona	170
Mimikyu	170
Primarina	170

Table 1. 5 most frequent Pokemon

Pokemon	Community frequency
Roserade	44
Miltank	42
Magnezone	37
Hitmonlee	37
Armaldo	31

Table 2. 5 least frequent Pokemon