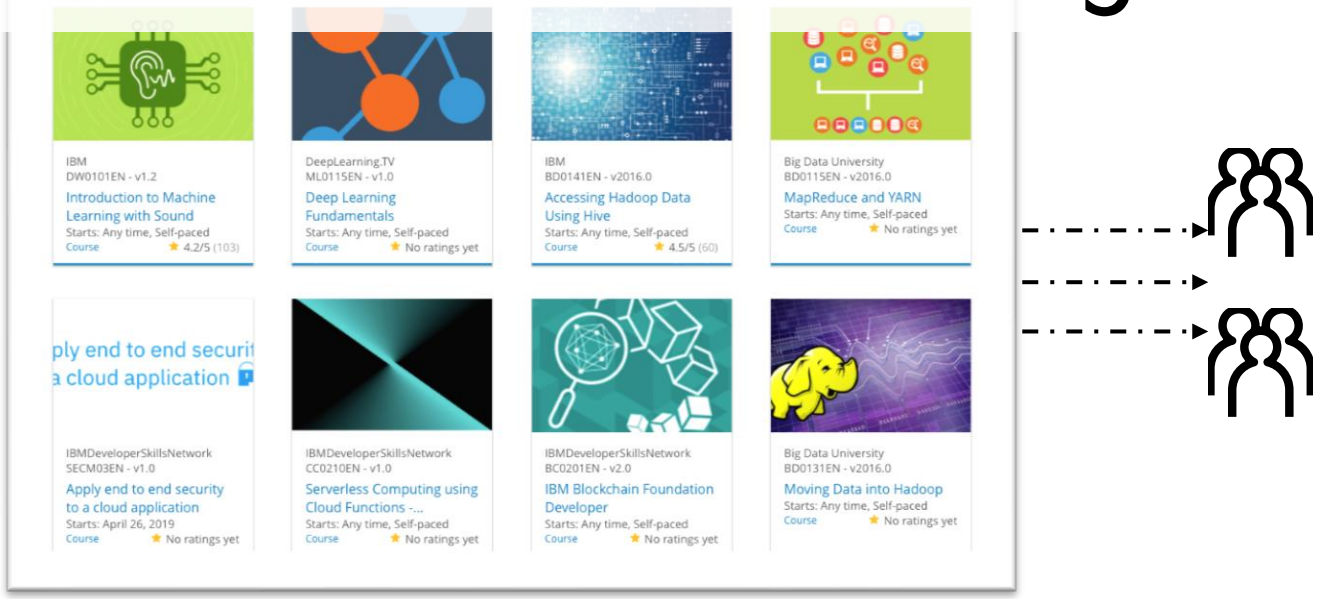
Build a Personalized Online Course Recommender System with Machine Learning

Tianxiang Yang
2024-11-18



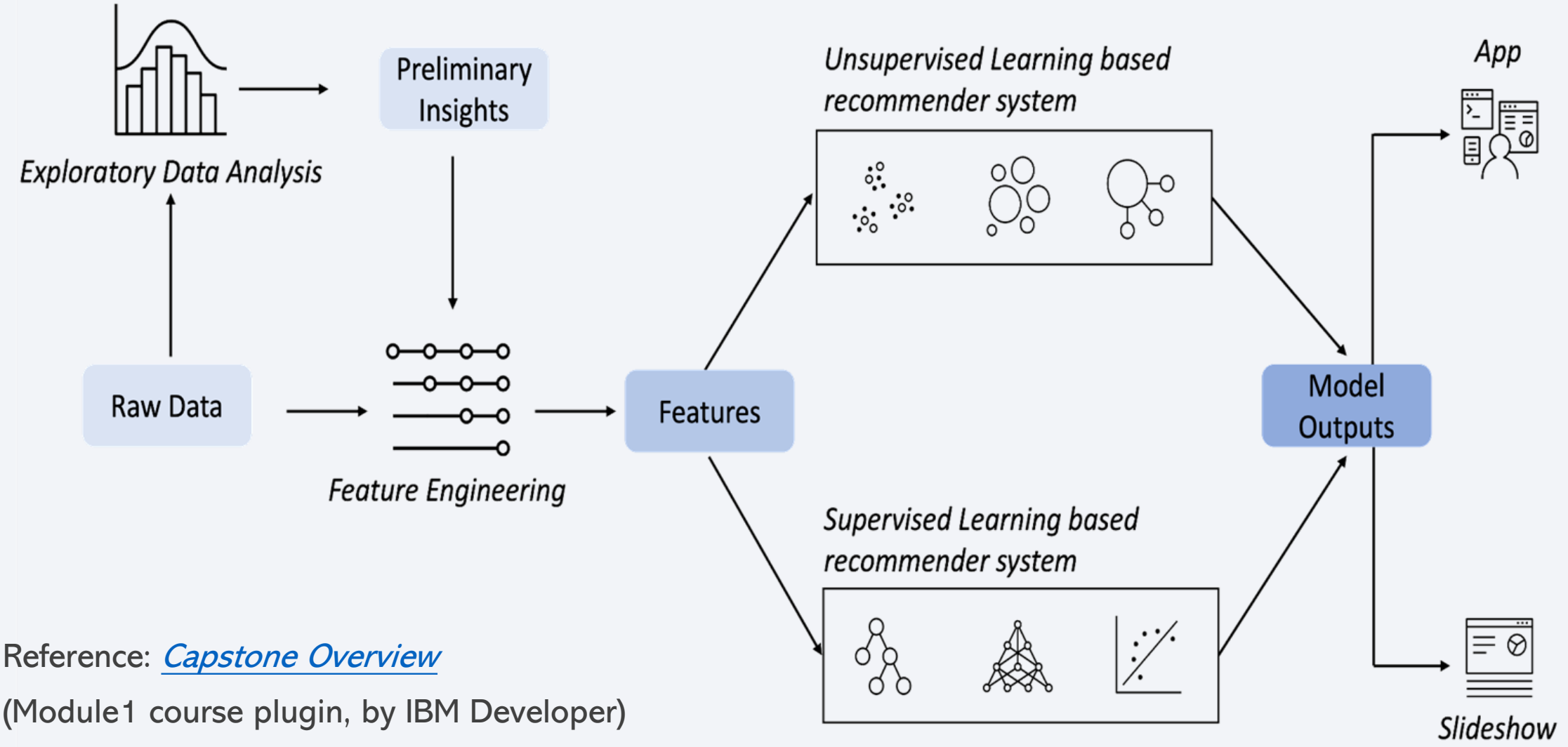
Outline

- Introduction and Background
- Exploratory Data Analysis
- Content-based Recommender System using Unsupervised Learning
- Collaborative-filtering based Recommender System using Supervised learning
- Conclusion
- Appendix

Introduction

- Nowadays, the commercial opportunities are coming in huge volume, variety and velocity, so does it with its associated data. To get more market penetration and share, a well informed discussion is necessary, especially for such emerging business as open online courses. For another, the mushrooming technologies of machine learning, either supervised, unsupervised or deep, have been inspiring data-driven wisdom for the industry players.
- Assuming a working scenario for MOOCs, to provide better and more sufficient idea for marketing and R&D, a project of ‘recommender systems’ is under way, which is to help learners quickly find new interested courses and better pave their learning paths. Data drivers can be leveraged by this system to decide how to keep and enroll more learners interacting with more courses, thus increasing the competitiveness of business lines.

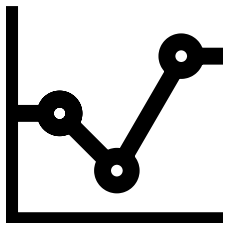
Tasks and roadmaps



Reference: *Capstone Overview*

(Module1 course plugin, by IBM Developer)

Exploratory Data Analysis



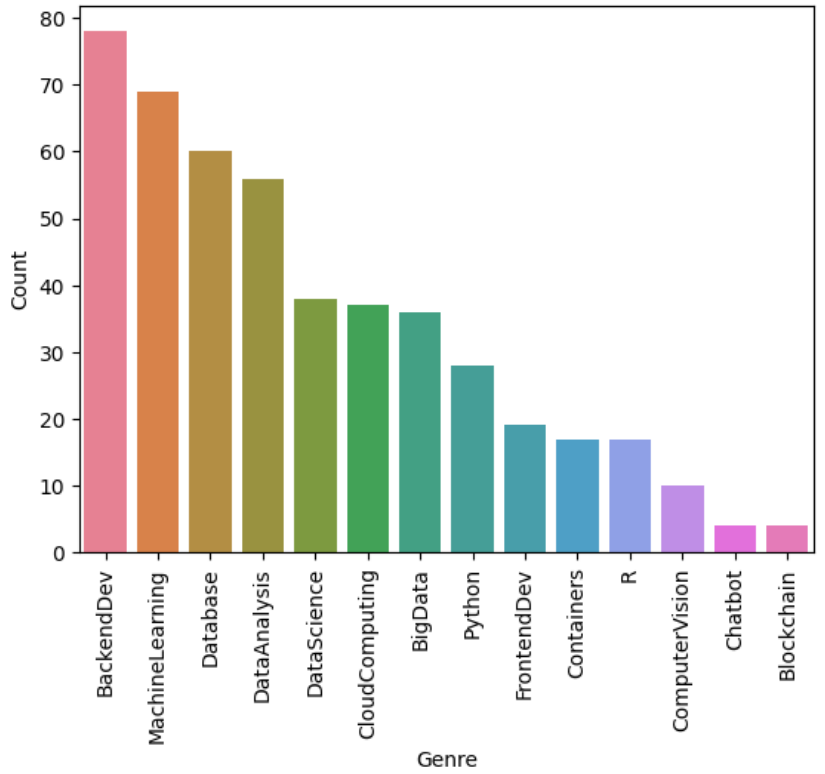
Course counts per genre

- Barchart of course counts per genre

- Brief explanations:

a) The course genres are some popular topics related to courses, which may come from the broken descriptions of courses

b) As we can see, BackendDev, MachineLearning, Database, DataAnalysis, etc. gain more popularity over the rest



Course enrollment distribution

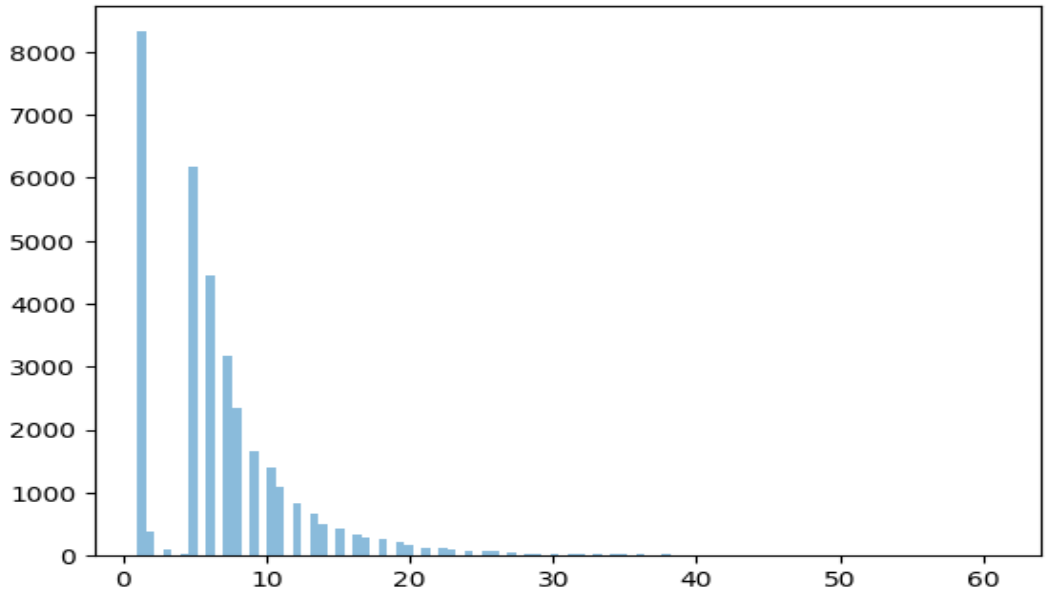
- Histogram showing the enrollment distributions

- Brief explanations:

a) It clearly shows how many users enrolled in 1 course. The grain size of '1 course' is controlled by calling:

```
plt.hist(userenrollments_df, bins=100)
```

a) As it depicts, courses' enrollments roughly fit exponential distribution, which is considered one typical sort of patterns.



20 most popular courses

- Table showing the most popular 20 courses

- Brief explanations:

a) It lists items grouped and ranked by the counts.

b) The most popular course is PY0101EN, DS0101EN, BD0101EN, etc.

c) We implement it by calling:

```
itemdf = ratings_df.groupby('item')
itemenrollments_df = items_df.size().reset_index()
itemenrollments_df.columns = ['course','Ratings']
itemenrollments_df = itemenrollments_df.sort_values
(by='Ratings', ascending=False)
itemenrollments_df.head(20)
```

	course	Ratings
101	PY0101EN	14936
54	DS0101EN	14477
4	BD0101EN	13291
5	BD0111EN	10599
42	DA0101EN	8303
55	DS0103EN	7719
82	ML0101ENv3	7644
18	BD0211EN	7551
56	DS0105EN	7199
1	BC0101EN	6719
63	DV0101EN	6709
86	ML0115EN	6323
24	CB0103EN	5512
103	RP0101EN	5237
112	ST0101EN	5015
27	CC0101EN	4983
36	CO0101EN	4480
46	DB0101EN	3697
6	BD0115EN	3670
61	DS0301EN	3624

Word cloud of course titles

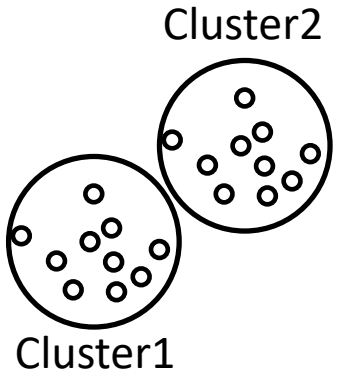
- Word cloud created from course titles

- Brief explanations:

a) It depicts the frequency of title words of courses, with common stop words and less meaningful ones filtered.

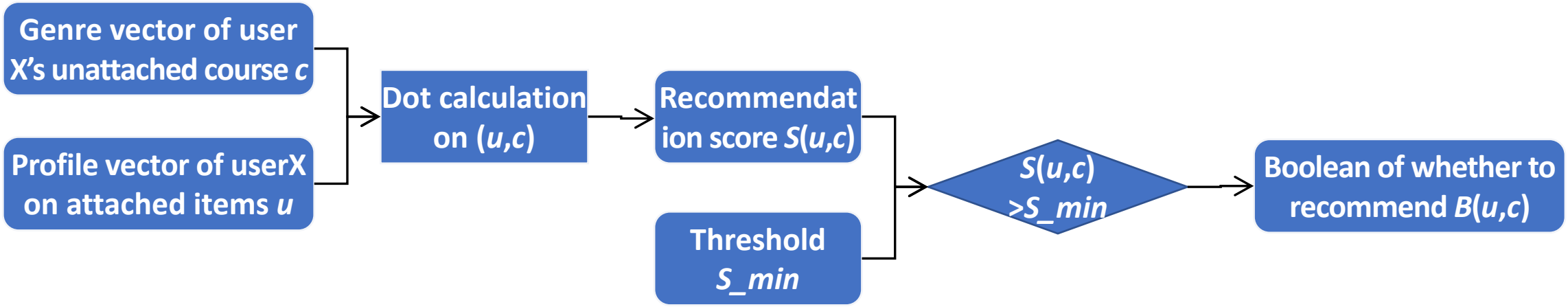
b) As we can see, python, data science, machine learning, etc are in phenomenal shape, indicating their popularity.

Content-based Recommender System using Unsupervised Learning



Flowchart of content-based recommender system using user profile and course genres

- Flowchart to implement the content-based recommender system



a) Vector *u* can be based on item's genre, etc. to get the 'appetite' by userX on courses.

b) Vector *c* is in 1 col and vector *u* in 1 row, values of both must focus on the same comparable features.

c) It is just to illustrate 1 user on 1 course; For *N* courses, set *C* as matrix with *N* cols, and for *M* users, set *U* as matrix with *M* rows.

Evaluation results of user profile-based recommender system

Hyper-parameter settings:

a) ***S_min*** 'score_threshold'=10

As 'res_df' with 1500424 rows, there are 44.3 (=1500424/33901) new/unseen courses have been recommended per user on average

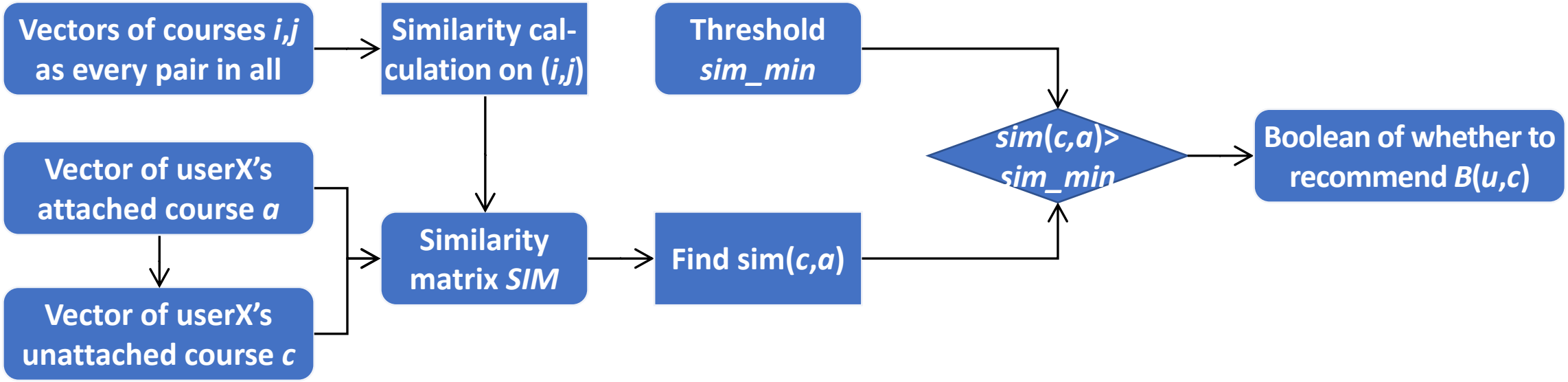
USER	COURSE_ID	SCORE
2	ML0201EN	43
2	GPXX0ZG0EN	43
2	GPXX0Z2PEN	37
2	DX0106EN	47
2	GPXX06RFEN	52
...	...	...
2102680	excourse62	15
2102680	excourse69	14
2102680	excourse77	14
2102680	excourse78	14
2102680	excourse79	14

As 'res_df' grouped by 'COURSE_ID' and sorted desc by avg 'score' shows, the most frequently recommended courses is excourse72/73

COURSE_ID	SCORE
excourse72	28.83573
excourse73	28.83573
TMP0105EN	28.178474
RP0105EN	26.586875
excourse31	26.232664
TA0106EN	25.667773
SC0103EN	25.634683
excourse21	25.634683
excourse22	25.634683
ML0122EN	25.634683

Flowchart of content-based recommender system using course similarity

- Flowchart to implement the course similarity based recommender system



a) The similarity can be measured by Cosine, Euclidean, Jaccard index, etc. on genres, BOWs, etc.

b) Values of vector *i*/*j* and vector *a*/*c* must focus on the same comparable features such as genres/BoWs.

c) It is just to illustrate 1 user on 1 unattached course consulting 1 attached course; For *L* courses of *a*, max ***sim*(*c,a*) among a's in** *L* cols can measure degree; *M* users at *N* courses is same to our former case.

Evaluation results of course similarity based recommender system

Hyper-parameter settings:

a) ***sim_min*** 'threshold'=0.6

As 'res_df' with 289738 rows, there are 8.55 (=289738/33901) new/unseen courses have been recommended per user on average

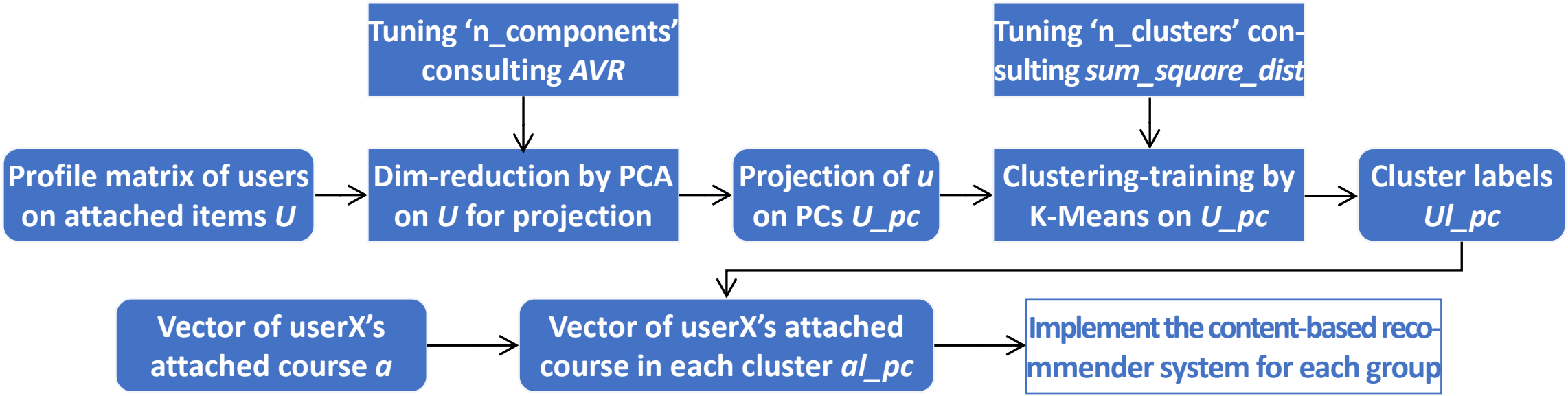
USER	COURSE_ID	SCORE
2	ML0120ENv3	1
2	DX0106EN	0.947623
2	CB0101EN	0.923381
2	TMP0101EN	0.889499
2	excourse23	0.739704
...	...	...
2102680	excourse65	0.638641
2102680	excourse28	0.623754
2102680	excourse25	0.600535
2102983	DAI101EN	0.668994
2103039	DAI101EN	0.668994

As 'res_df' grouped by 'COURSE_ID' and sorted desc by avg 'score' shows, the most frequently recommended courses is ML0120ENv3

COURSE_ID	SCORE
ML0120ENv3	0.997113
ML0120EN	0.99649
ML0120ENv2	0.993685
ML0122ENv1	0.982873
CB0103EN	0.923381
CB0101EN	0.923381
TMP0101EN	0.889499
GPXX0JLHEN	0.877866
excourse23	0.737565
Excourse36	0.737565

Flowchart of clustering-based recommender system

- Flowchart to performed user profile clustering based recommender system



a) Matrix *U* can be based on item's genre, etc. to get the 'appetite' by userX on courses.

b) It is to illustrate all users on all courses during clustering steps; Then, supposing *M_g* users in group *g*, implement the content-based recommender system iteratively on *M_g* users and get recommendation scores based on enrollments.

Evaluation results of clustering-based recommender system

Hyper-parameter settings:

a) ‘n_components’=9; b) ‘n_clusters’=15; c) ‘threshold’(of enrollments)=10

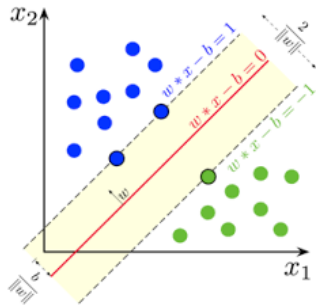
As ‘res_pop_df’ has ‘USER’ duplicated, we group it by ‘USER’ and get mean size. It turns out to be 65.89 if calling `res_pop_df.groupby(by=['USER']).size().mean()`

USER	COURSE_ID	enrollments
2	DS0105EN	107
2	DE0205EN	51
2	ML0111EN	33
2	ML0120ENv3	11
2	SW0201EN	44
...	...	...
2064387	ML0101ENv3	12
2088382	BC0101EN	11
2088382	ML0101ENv3	12
2088528	BC0101EN	11
2088528	BD0101EN	11

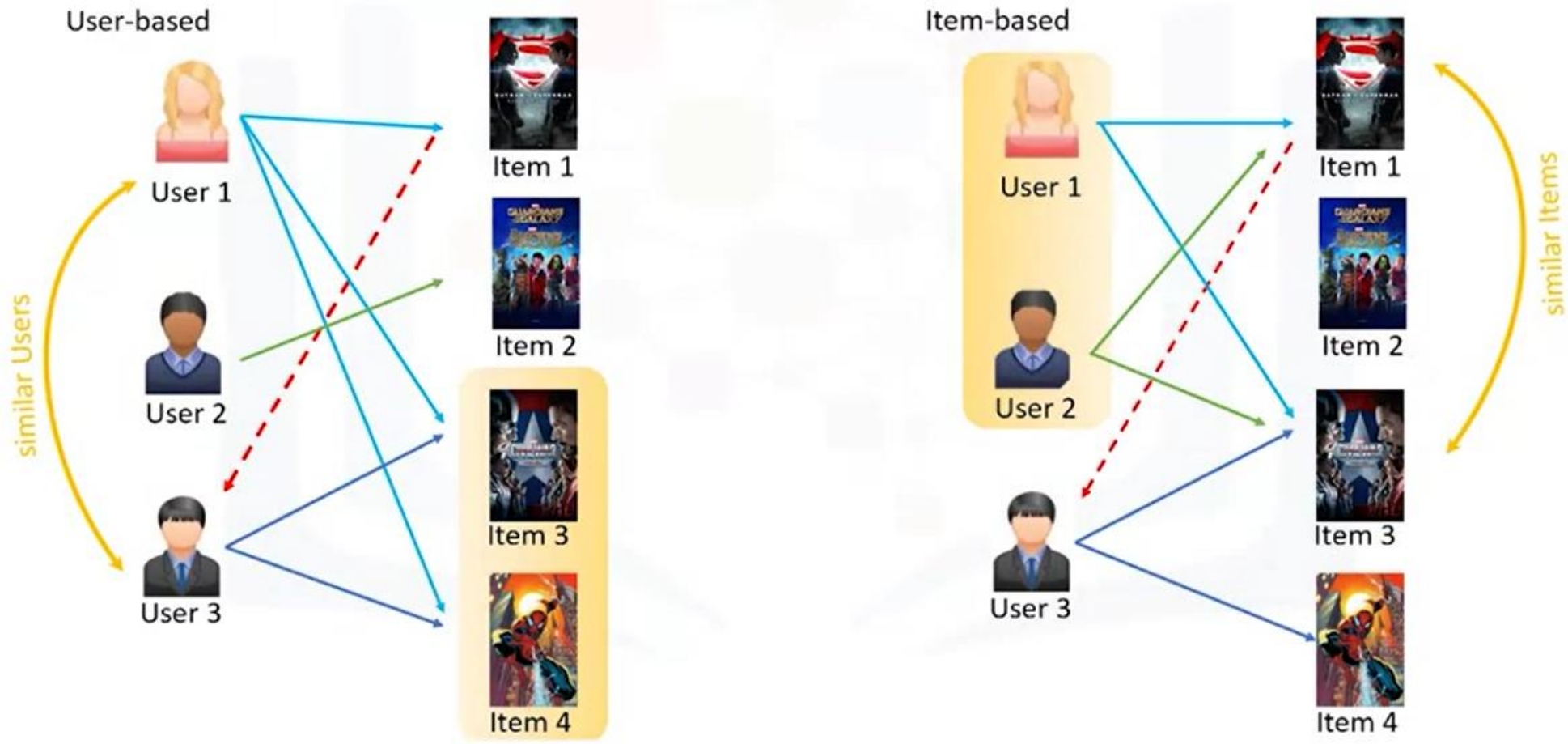
As ‘res_pop_df’ grouped by ‘COURSE_ID’ and sorted desc by avg ‘enrollments’ shows, the most frequently recommended courses is DS0101EN

COURSE_ID	SCORE
DS0101EN	1480.2673
BD0101EN	1161.7016
PY0101EN	825.73404
CNSC02EN	755.54256
DS0103EN	699.64438
BD0111EN	687.89812
DS0105EN	626.49223
ML0115EN	526.04379
ML0101ENv3	498.29592
ST0101EN	476.25938

Collaborative-filtering Recommender System using Supervised Learning



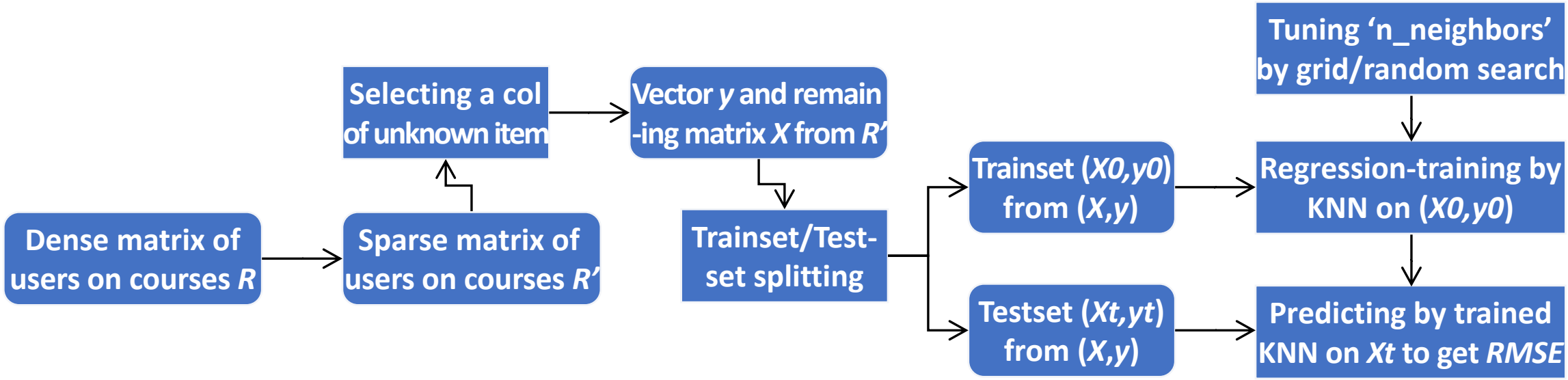
Logic behind collaborative-filtering recommender system using supervised learning



Reference: *Collaborative Filtering-Based Recommender Systems* (Module3 course video, by IBM Developer)

Flowchart of KNN based recommender system

- Flowchart to performed KNN based recommender system



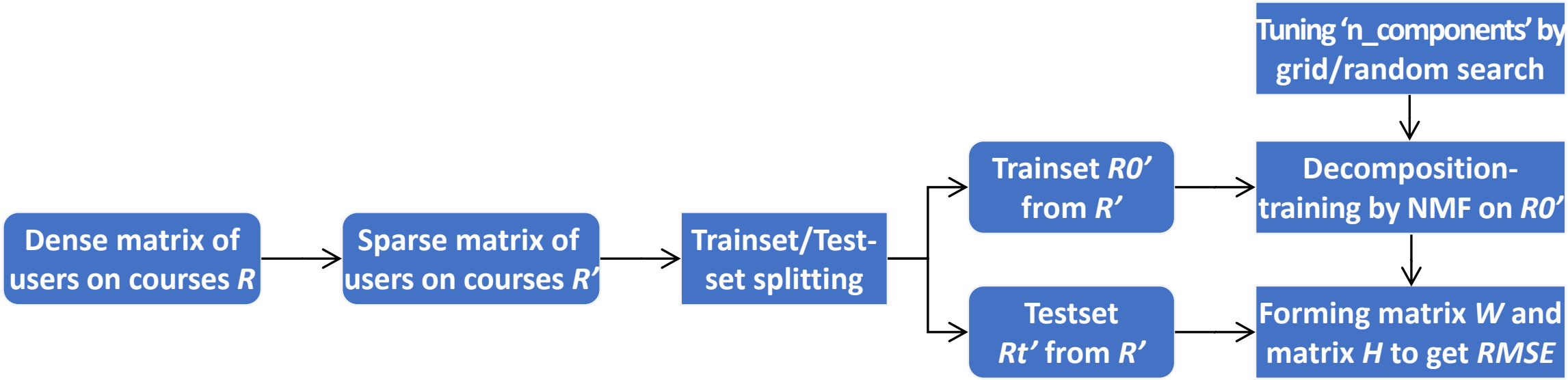
a) Matrix *R* is based on course item and have been gain the 'appetite' by all each users.

b) The selected col *y* from *R'* is supposed without 'appetite' by all each users, which has true value and is to be predicted using yet-to-be-trained models; *RMSE* makes interpretable sense.

c) It is to illustrate all users on all courses during regression steps.

Flowchart of NMF based recommender system

- Flowchart to performed NMF based recommender system



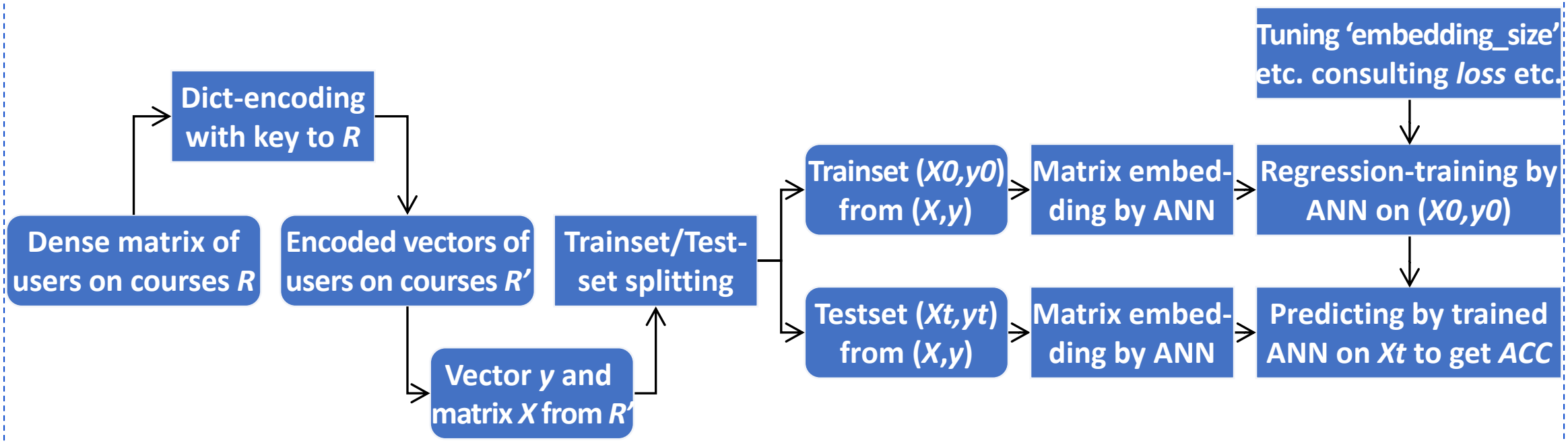
a) Matrix *R* is based on course item and have been gain the 'appetite' by all each users.

b) Matrix *W* is formed by fit_transform on *Rt'*, and matrix *H* by NMF's components; The dot calculation of (*W*,*H*) is to fit the trainset *R0'*; *RMSE* makes interpretable sense.

c) It is to illustrate all users on all courses during decomposition steps.

Flowchart of Neural Network Embedding based recommender system

- Flowchart to performed Neural Network Embedding based recommender system

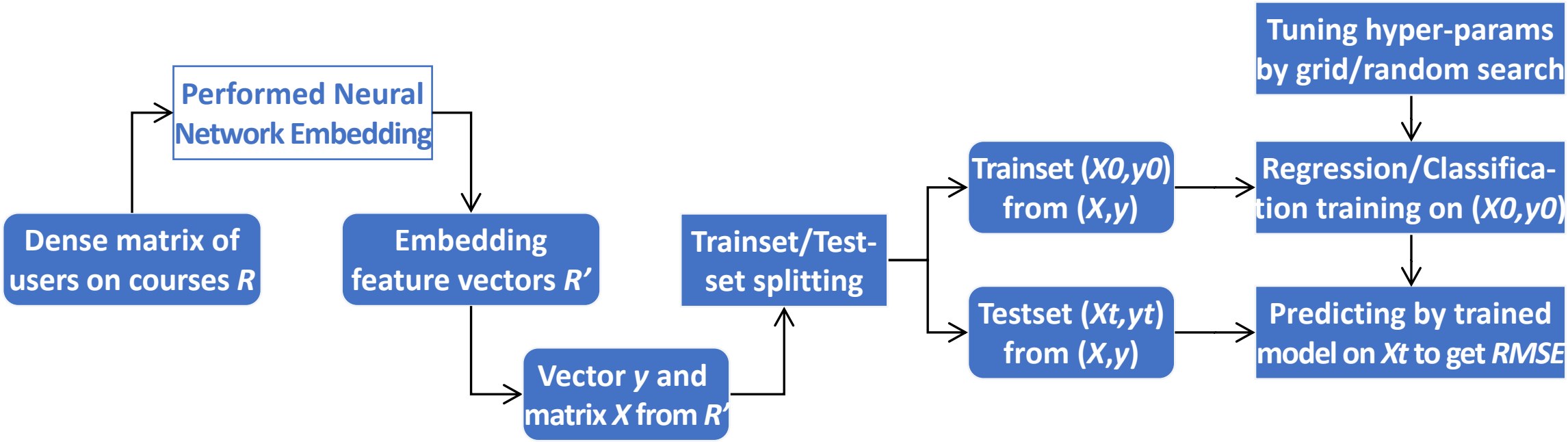


a) Length of embedding vector of users should be identical to unique-value count of users, so is it with length of embedding vector of courses; This is to formulate the inputs of regression sectors in ANN.

b) Encoded vectors were to be one-hot for categorical variables; Scaling is made for numerical variables.

Flowchart of typical regression/classification embedding based recommender system

- Flowchart to performed regression/classification embedding based recommender system



a) Length of embedding vector of users should be identical to unique-value count of users, so is it with length of embedding vector of courses; Embedding vectors are the outputs from ANN layers.

b) Suitability of models: RandomForest runs faster, and SVC more effective on high-dim features.

Compare the performance of collaborative-filtering models

- So far, we have built 10 collaborative-filtering models tested with *RMSE*:

a) **'knn_s'**: KNNBasic model on hyper default parameters set by surprise
b) **'knn'**: KNeighborsClassifier model on {'n_neighbors':10} and default hyper parameters set by sklearn
c) **'nmf_s'**: NMF model on {'init_low':0.5, 'init_high':5.0, 'n_factors':32} and default hyper parameters set by surprise
d) **'nmf'**: NMF model on {'n_components':6, 'init':'random'} and default hyper parameters set by sklearn
e) **'rn'**: RecommenderNet (ANN) model on {'embedding_size':16} and default hyper parameters self-defined
f) **'rn_1'**: RecommenderNet (ANN) model on {'embedding_size':8} and default hyper parameters self-defined
g) **'lr'**: LinearRegression model on default hyper parameters set by sklearn
h) **'best_rg'**: Ridge model on {'alpha': 10} and default hyper parameters set by sklearn
   tuned by sklearn grid search on {'alpha':[0.0001,0.001,0.01,0.1,1,10]}
i) **'best_ls'**: Lasso model on {'alpha':0.001} and default hyper parameters set by sklearn
   tuned by sklearn grid search on {'alpha':[0.0001,0.001,0.01,0.1,1,10]}
j) **'best_en'**: ElasticNet model on {'alpha':0.001, 'l1_ratio':0.5} and default hyper parameters set by sklearn
   tuned by sklearn grid search on {'alpha':[0.0001,0.001,0.01,0.1,1,10], 'l1_ratio':[0.1,0.25,0.5,0.75,0.9]}

Compare the performance of collaborative-filtering models

- Barchart visualizing the performance metric RMSE of the above 10 collaborative-filtering models

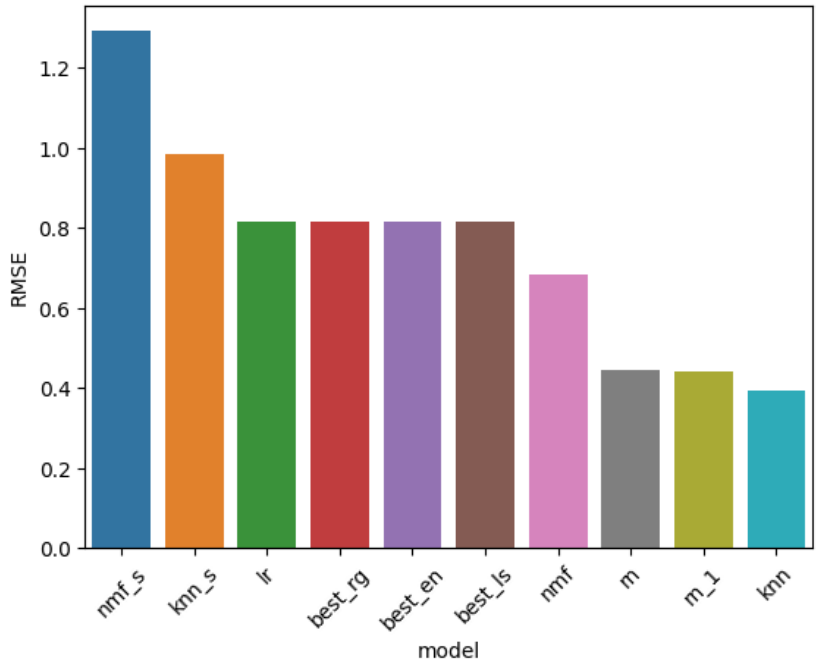
- Brief explanations:

a) *RMSE* is measured by the deviation between predict and y_test, negatively correlated to model performance:

```
RMSE = sqrt(mean( (predict – y_test)**2 ))
```

b) As we can see, most *RMSE* are rather low given the scale of targets, indicating the predicting proficiency of most trained models.

c) If case shared, models can be compared----'nmf_s' out-performs 'nmf'; 'rn_1' and 'rn' outperform 'lr', 'best_rg', 'best_ls' and 'best_en' which performs similarly.

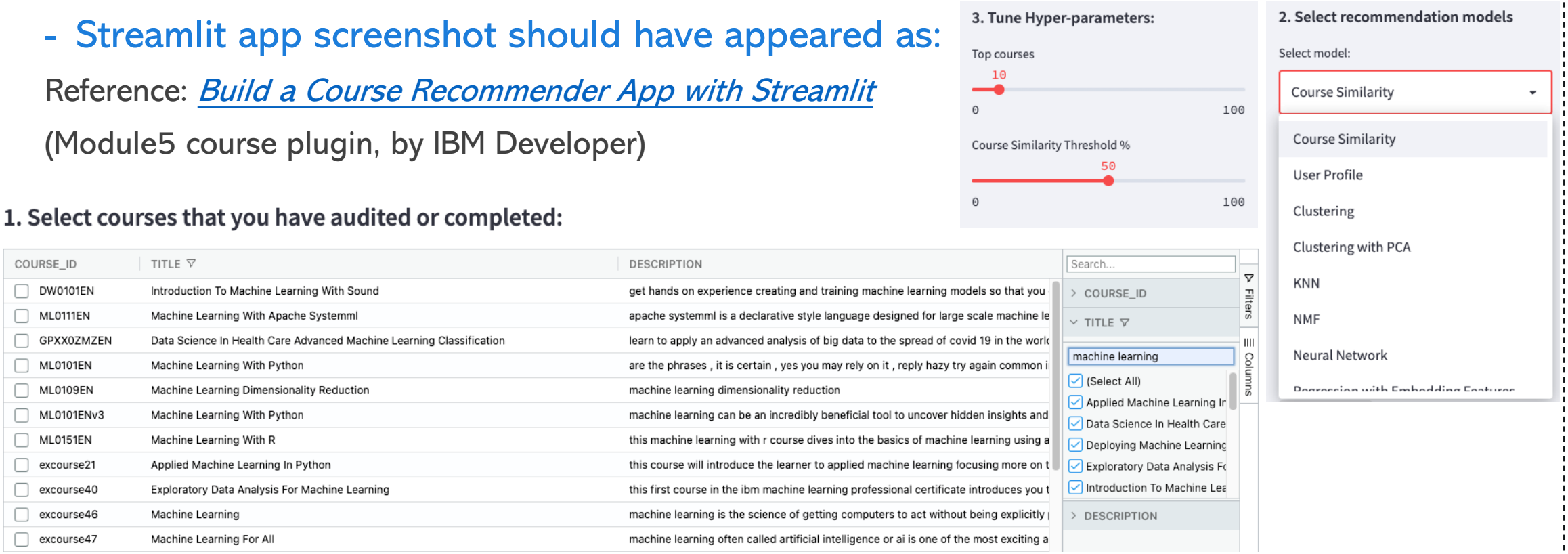


Optional: Build a course recommender system app with Streamlit

- Streamlit app screenshot should have appeared as:

Reference: *Build a Course Recommender App with Streamlit*

(Module5 course plugin, by IBM Developer)



A Streamlit App URL for a live demo (in trial):

a) https://yang20172020-8501.theia-0-labs-prod-misc-tools-us-east-0.proxy.cognitiveclass.ai/

Conclusions

- EDA does tell some narrative stories about the patterns of users and items. We can make it either on genres, titles or ids, either by rating scores or enrollments. To densely-formatted data, feature wrangling such as grouping-statistics, sorting, slicing, etc. can be used in combination, which depends on investigated issues.
- Without recommendation labels from users or items, we can label and deconstruct the existing observations for better knowledge using Unsupervised Learning. Recommendation of this type always requires threshold judgments.
- With recommendation labels from users and items, we can predict unobserved part of target using Supervised Learning. It turns out that with more hypermeters specified and tuned, models gain more potentials to decrease recommendation deviation. For more simply-modeled issues, the promotional effect may be less.
- Compared with typical regression/classification with embedded inputs, embedding-based NeuralNetworks have potentials to streamline prediction with less overfitting risk.

Appendix

- Notebook outputs:

a) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/lab_jupyter_eda.ipynb
b) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/lab_jupyter_fe_bow_solution.ipynb
c) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/lab_jupyter_fe_course_sim.ipynb
d) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/lab_jupyter_content_user_profile.ipynb
e) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/lab_jupyter_content_course_similarity.ipynb
f) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/lab_jupyter_content_clustering.ipynb
g) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/lab_jupyter_cf_knn.ipynb
h) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/lab_jupyter_cf_nmf.ipynb
i) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/lab_jupyter_cf_ann.ipynb
j) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/lab_jupyter_cf_regression_w_embeddings.ipynb
k) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/lab_jupyter_cf_classification_w_embeddings.ipynb

- Python code snippets:

a) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/recommender_app.py
b) https://github.com/yang20172020/Machine-Learning-Capstone/blob/main/backend.py