Recommender System with Machine Learning

Bo-Yan Huang
2024 1/16

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

Background:

In the era of digital education, our project aims to transform the learning landscape through an advanced recommender system. The mission is to enhance user experiences by seamlessly connecting learners with new, relevant courses, shaping personalized educational paths.

This initiative not only strives to improve user satisfaction but also anticipates a positive impact on company revenue. By facilitating user engagement with diverse courses, we envision a symbiotic relationship between learner contentment and business growth.

Introduction

Problem states:

1. Limited Course Discoverability: Users face challenges in discovering new and relevant courses that align with their interests and learning goals, leading to a suboptimal learning experience.

2. Underutilized Learning Paths: The absence of personalized recommendations may result in users not fully realizing the potential of a structured learning path, hindering their educational progression and engagement.

3. Revenue Growth Opportunities: The current lack of an effective recommender system may be limiting the company's revenue potential, as user interactions with courses could be suboptimal.

Introduction

Hypothesis:

1. Relevance Hypothesis: Implementing a recommender system will significantly improve the relevance of course recommendations for users, leading to increased user satisfaction.

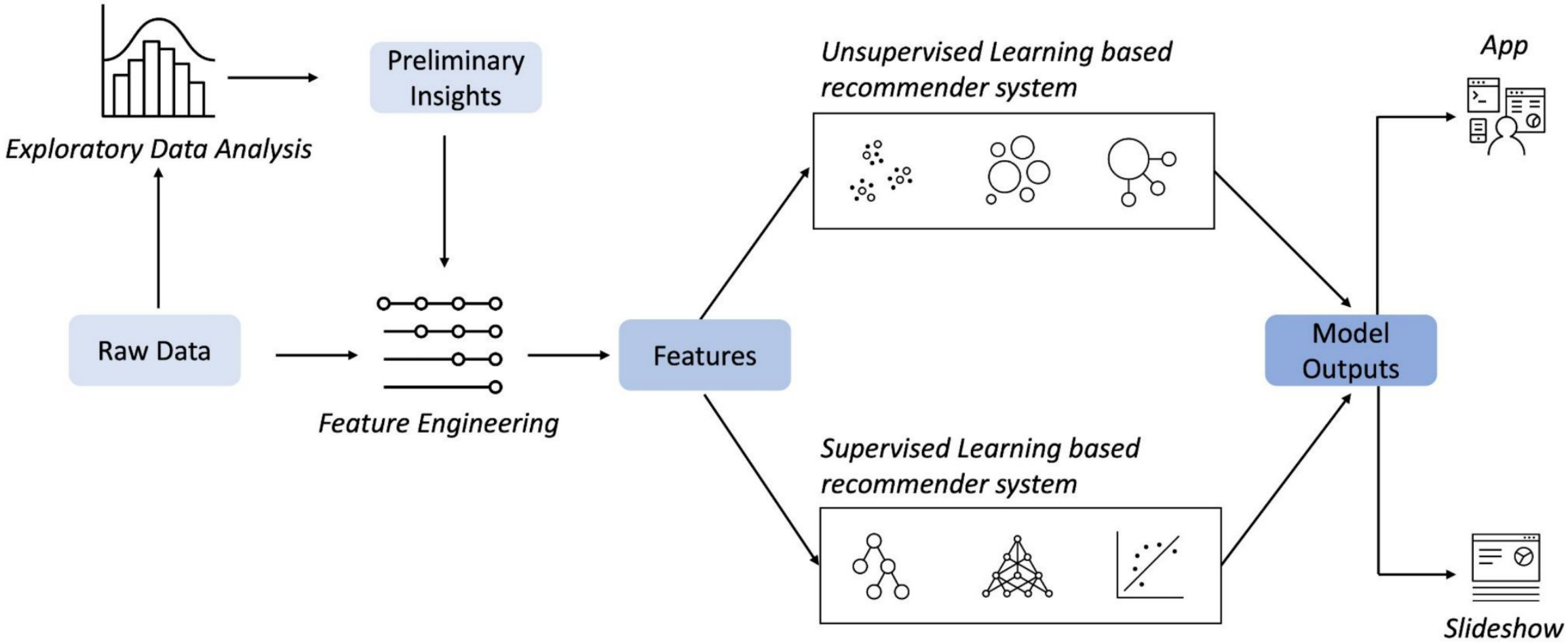
2. Engagement Hypothesis: A personalized learning experience, facilitated by the recommender system, will boost user engagement with courses, contributing to a more active and committed user base.

3. Learning Path Optimization Hypothesis: The recommender system will optimize users' learning paths, guiding them towards a more structured and effective educational journey.

4. Revenue Impact Hypothesis: Enhanced user engagement through the recommender system will positively impact company revenue by increasing course enrollments and user interactions.

5. Model Performance Hypothesis: Through the Proof of Concept (PoC) phase, it is hypothesized that the exploration and comparison of various machine learning models will identify a model with superior performance in offline evaluations, setting the stage for effective online implementation.

Machine Learning Workflow



Exploratory Data Analysis

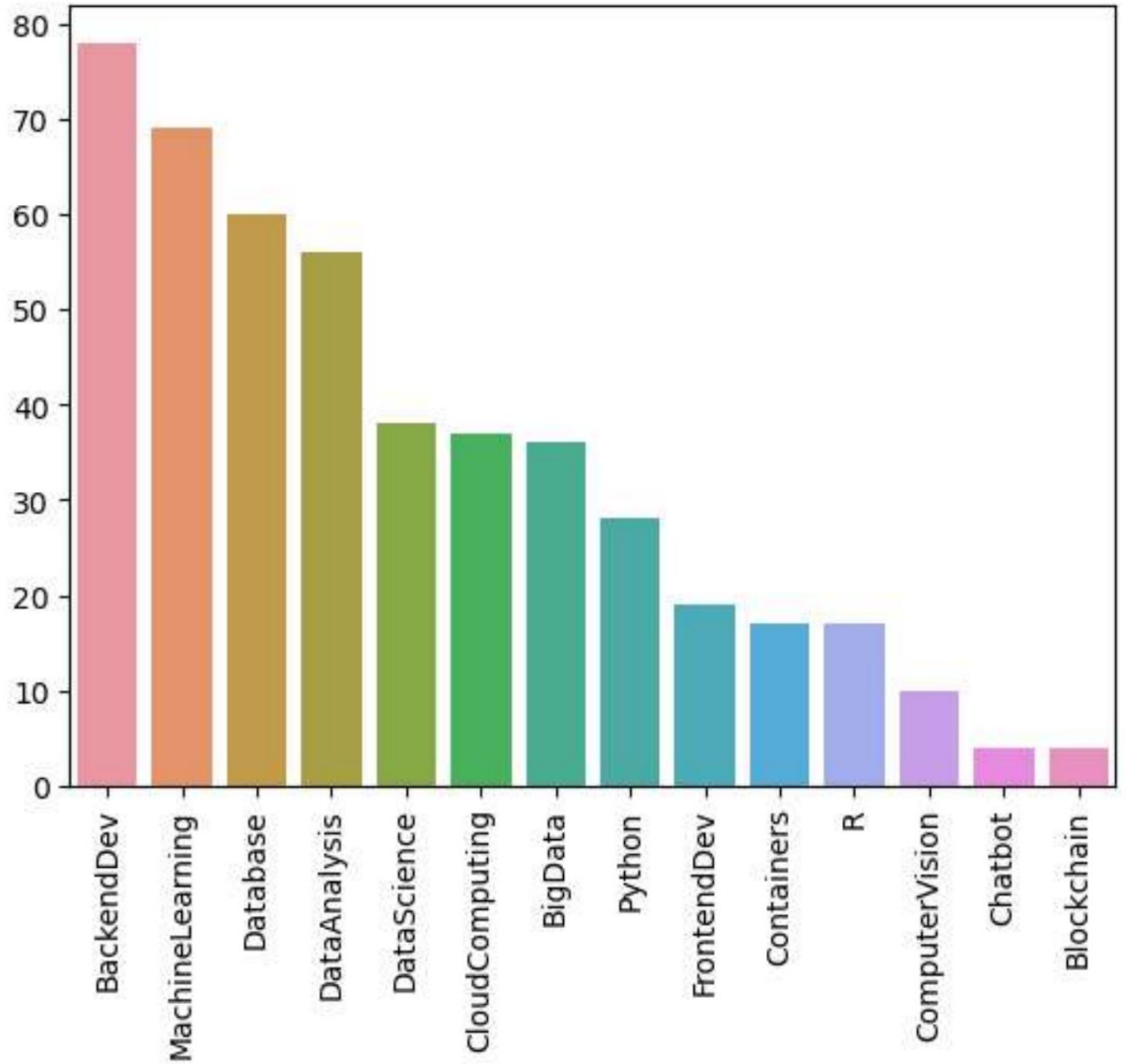
Steps:

1. Statistical Overview
2. Keyword Identification using WordCloud
3. Find Popular Course Genres
4. Summary Statistics and Visualizations for Enrollment Data

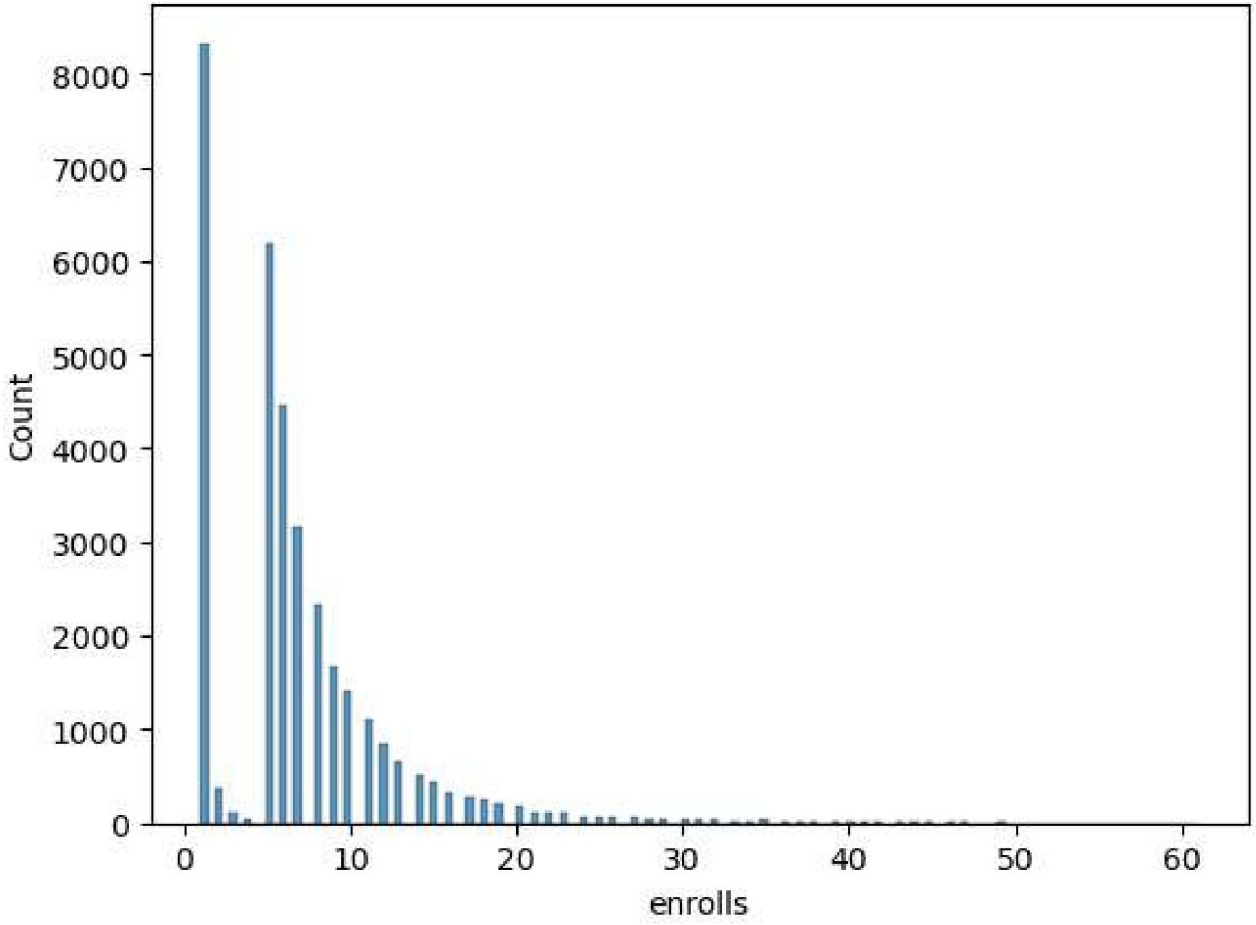
Columns/Features of the Data:

```
COURSE_ID          object
TITLE              object
Database            int64
Python              int64
CloudComputing      int64
DataAnalysis        int64
Containers          int64
MachineLearning     int64
ComputerVision      int64
DataScience         int64
BigData             int64
Chatbot             int64
R                   int64
BackendDev          int64
FrontendDev         int64
Blockchain          int64
dtype: object
```

Enrollment counts per genre



Course enrollment distribution



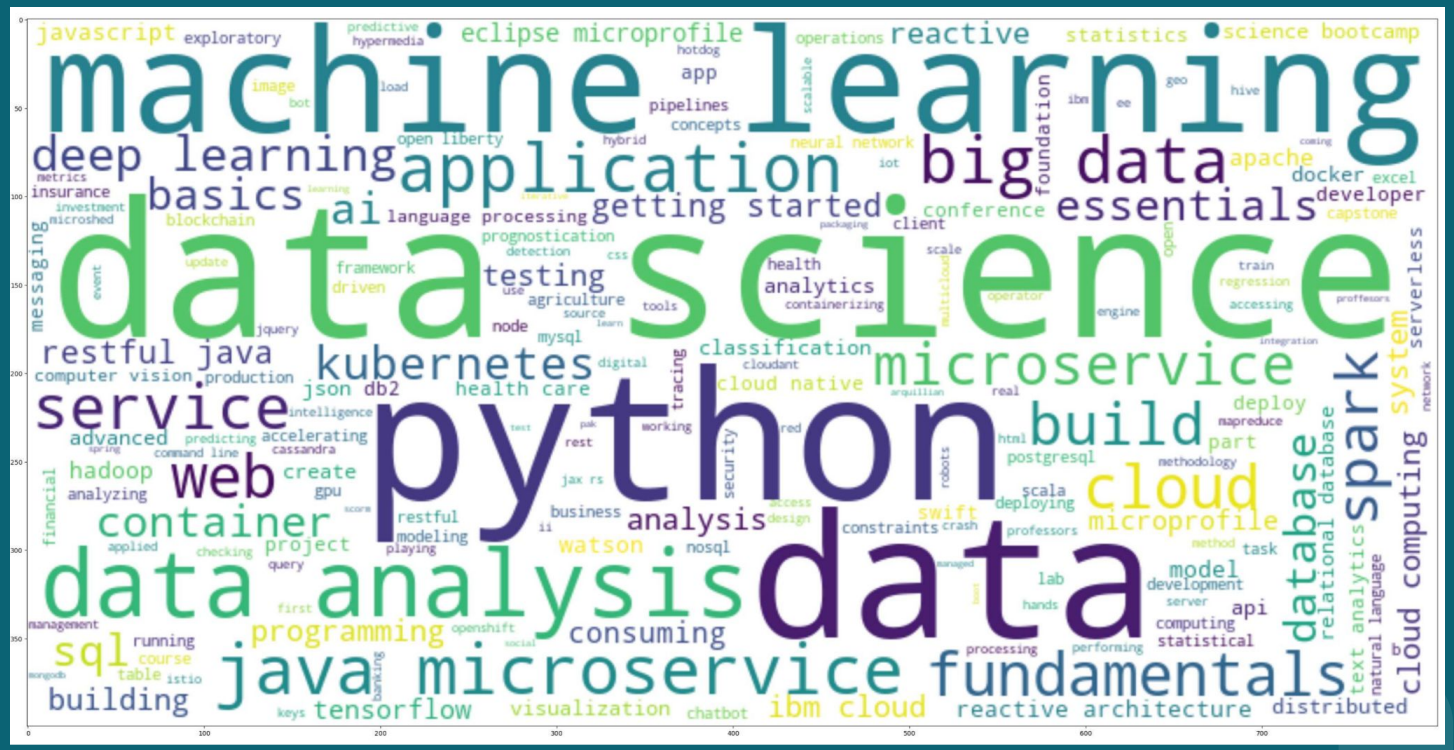
20 most popular courses

	TITLE	Enrolls
0	python for data science	14936
1	introduction to data science	14477
2	big data 101	13291
3	hadoop 101	10599
4	data analysis with python	8303
5	data science methodology	7719
6	machine learning with python	7644
7	spark fundamentals i	7551
8	data science hands on with open source tools	7199
9	blockchain essentials	6719
10	data visualization with python	6709
11	deep learning 101	6323
12	build your own chatbot	5512
13	r for data science	5237
14	statistics 101	5015
15	introduction to cloud	4983
16	docker essentials a developer introduction	4480
17	sql and relational databases 101	3697
18	mapreduce and yarn	3670
19	data privacy fundamentals	3624

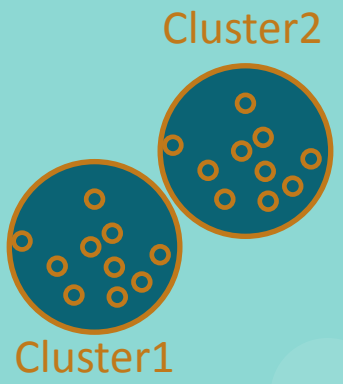
Identify keywords in course titles using a WordCloud

Top 10 omnipresences following by size:

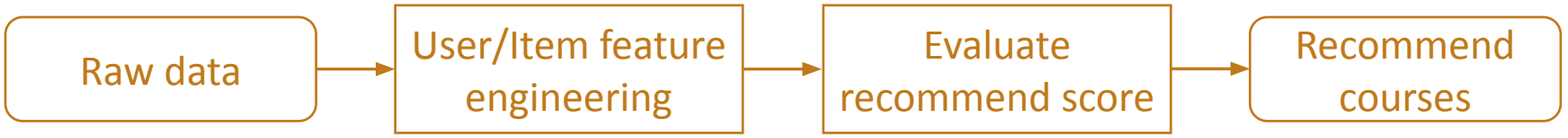
1. data
2. data science
3. python
4. machine learning
5. data analysis
6. application
7. big data
8. java
9. microse



Content-based Recommender System using Unsupervised Learning



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



Course genres dataframe:
`course_id, title, [genre_x, genre_y,...]`

User dataframe:
`user_id, [genre_interest_x, genre_interest_y,...]`

Analyze data
Drop the outliers
Normalize data

Extract features from users and courses (such as genres or BoW features)

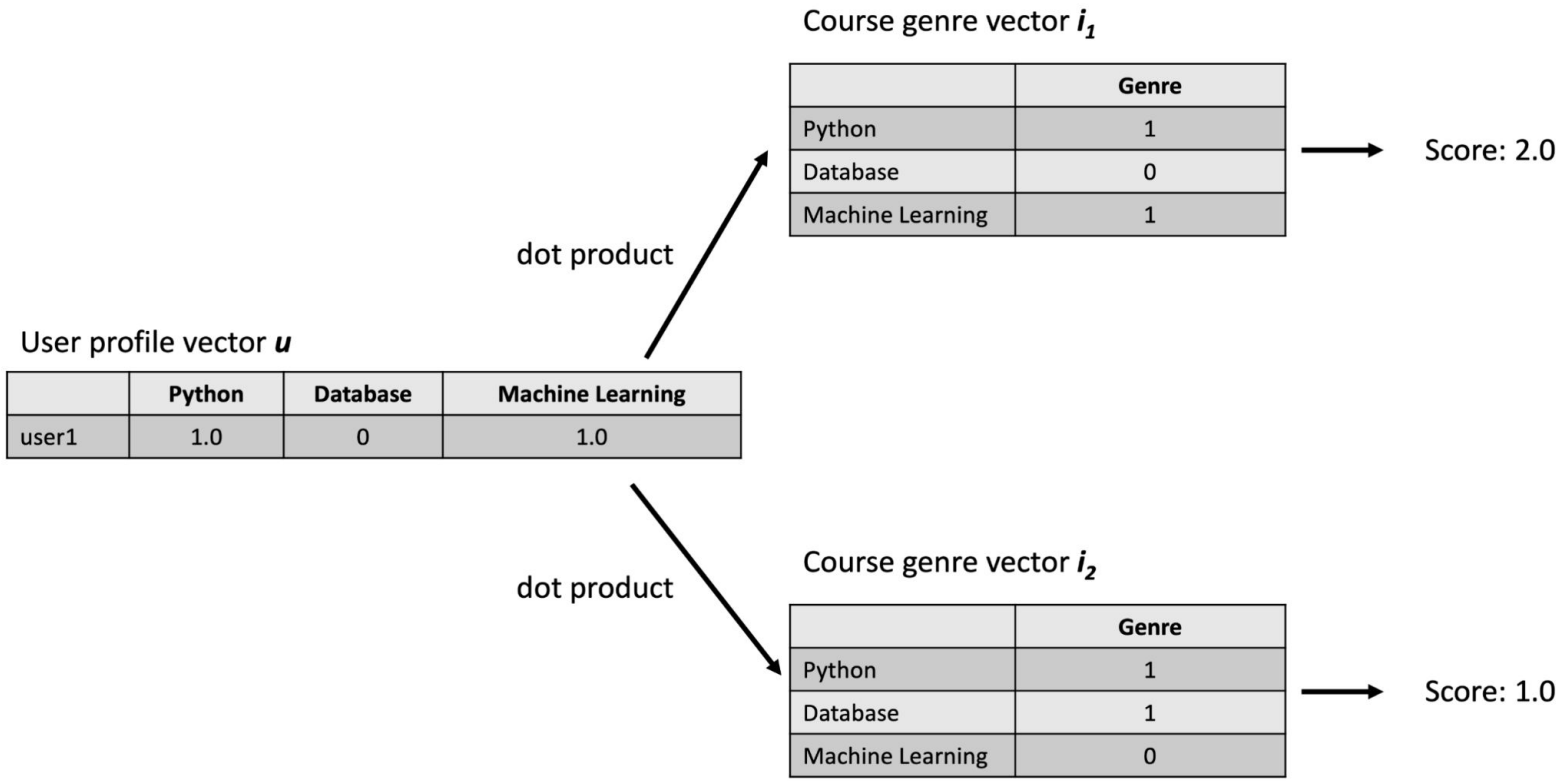
Use the dot product between the user profile feature vectors and course genre feature vectors obtain the recommend scores / similarities

Make predictions by setting a threshold on recommend scores

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

Evaluate recommend score

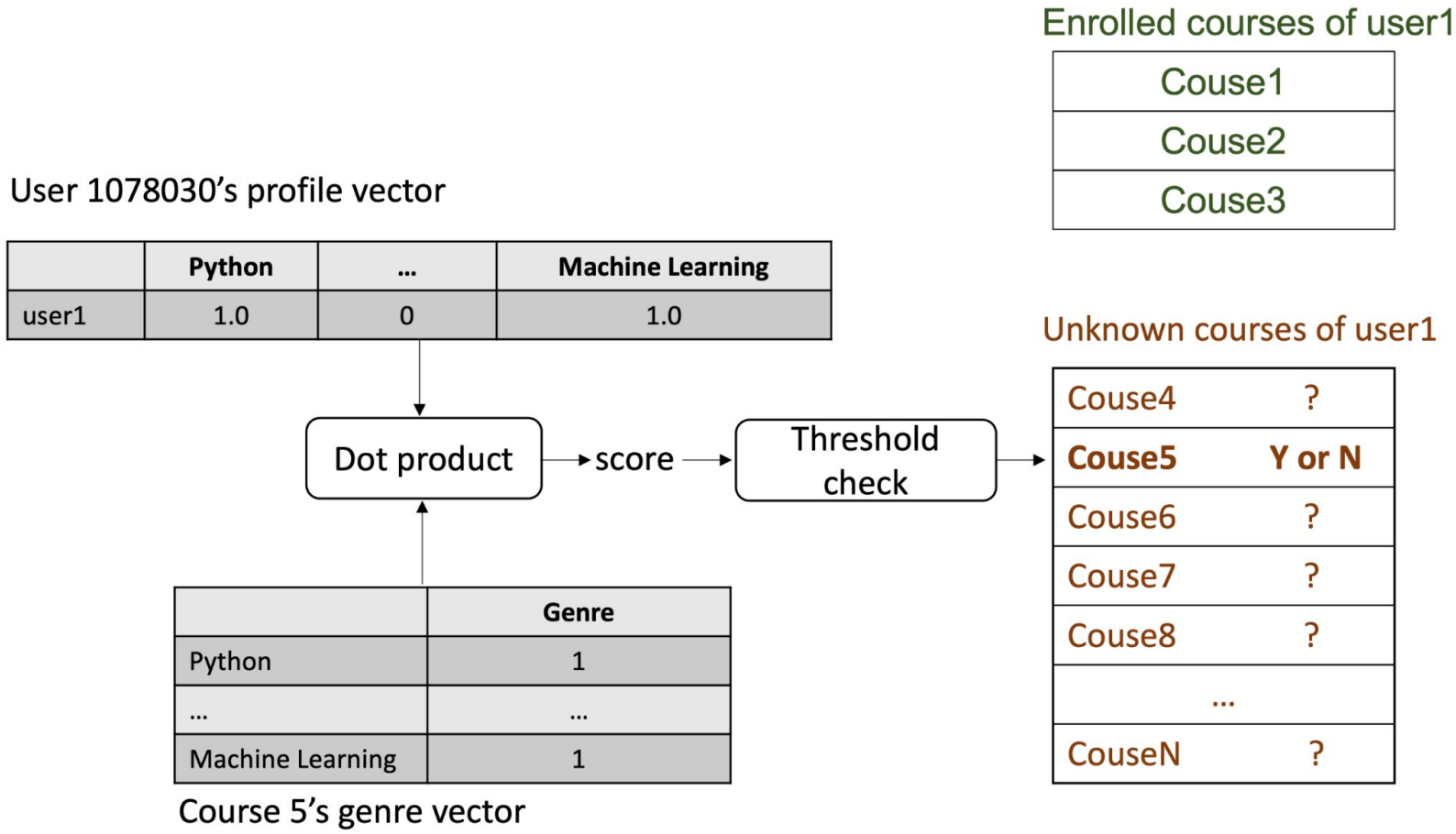
Use the dot product between the user profile feature vectors and course genre feature vectors obtain the similarities



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

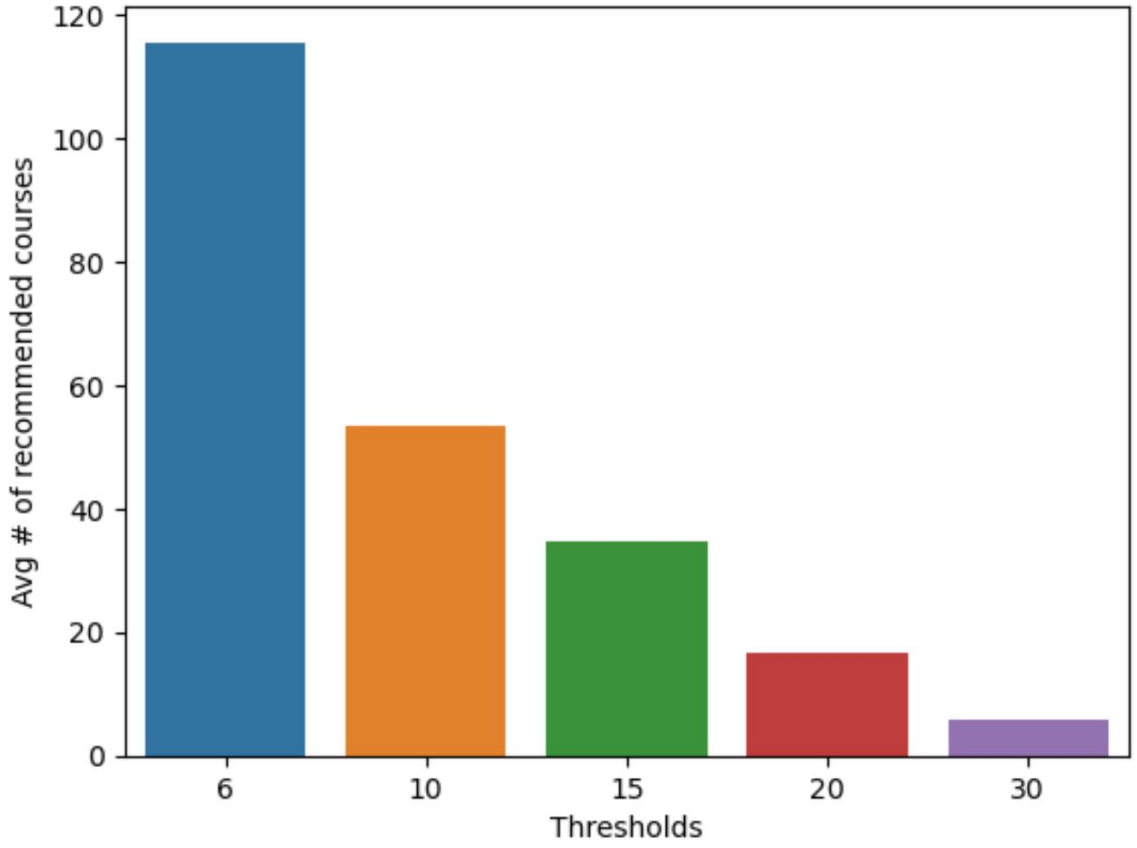
Recommend courses

Make predictions by setting a threshold on recommend scores



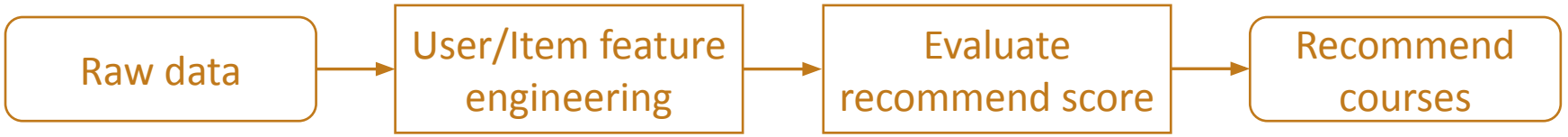
Evaluation results of user profile-based recommender system

In the lower left chart, we varied the threshold from 6 to 30. To ensure approximately 10 recommended courses per user, we set the threshold to 25 in order to calculate the top-10 commonly recommended courses in the lower right.



	COURSE_ID	recommended_count
59	TA0106EN	608
38	GPXX0IBEN	548
234	excourse21	547
235	excourse22	547
1	ML0122EN	544
186	GPXX0TY1EN	533
217	excourse04	533
219	excourse06	533
244	excourse31	524
285	excourse72	516

Flowchart of content-based recommender system using course similarity



Course genres dataframe:
`course_id, title, [genre_x, genre_y,...]`

User dataframe:
`user_id, course_id`

Analyze data
Drop the outliers
Normalize data

Extract features from courses (such as genres or BoW features)

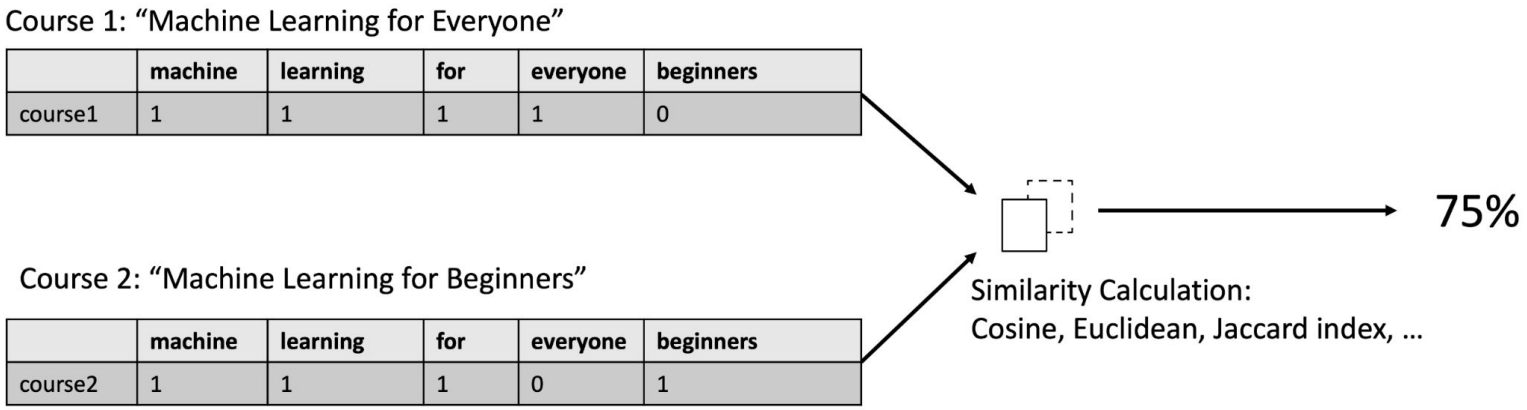
Find courses which are similar enough to your enrolled courses.

Make predictions by setting a threshold on recommend scores

Flowchart of content-based recommender system using course similarity

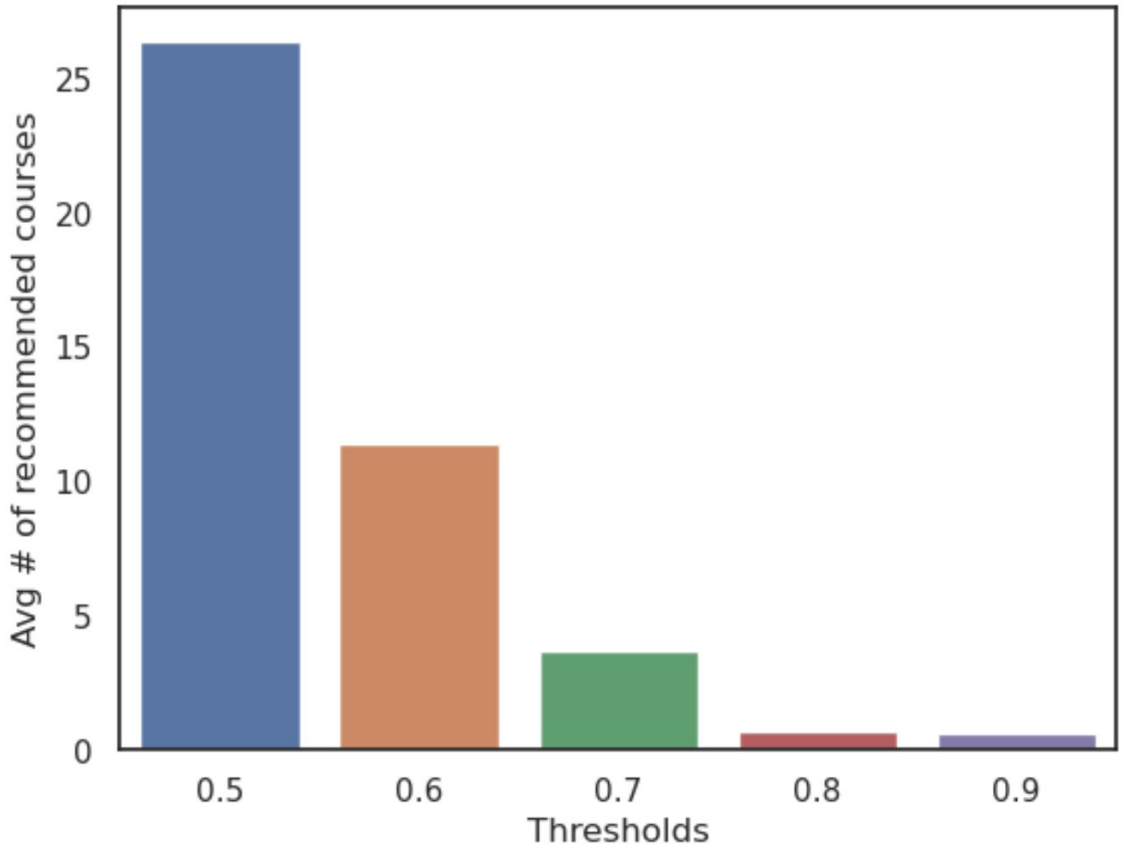
Evaluate recommend score

Find courses which are similar enough to your enrolled courses.
In our example, we use cosine approach as previous one.



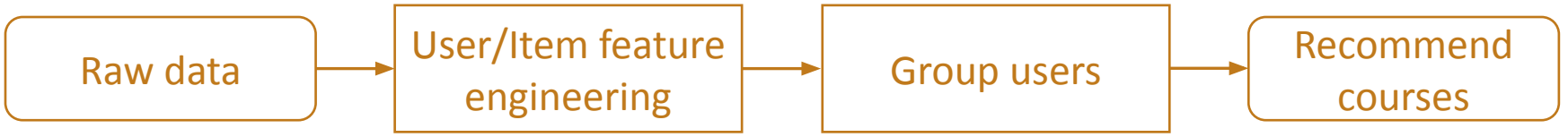
Evaluation results of course similarity based recommender system

In the lower left chart, we varied the threshold from 0.5 to 0.9. To ensure approximately 10 recommended courses per user, we set the threshold to 0.6 in order to calculate the top-10 commonly recommended courses in the lower right.



	COURSE_ID	recommended_count
153	CB0101EN	181
212	ML0120ENv3	156
110	ML0120EN	110
57	ML0122ENv1	105
202	ML0120ENv2	60
156	CB0103EN	7
0	ML0201EN	0
1	ML0122EN	0
2	GPXX0ZG0EN	0
3	RP0105EN	0

Flowchart of clustering-based recommender system



Course genres dataframe:
`course_id, title, [genre_x, genre_y,...]`

User dataframe:
`user_id, [genre_interest_x, genre_interest_y,...]`

Extract features from users such as enrollments or genres

Apply PCA on user profile feature vectors to reduce dimensions

Divide users into several groups based on their enrollments or genres

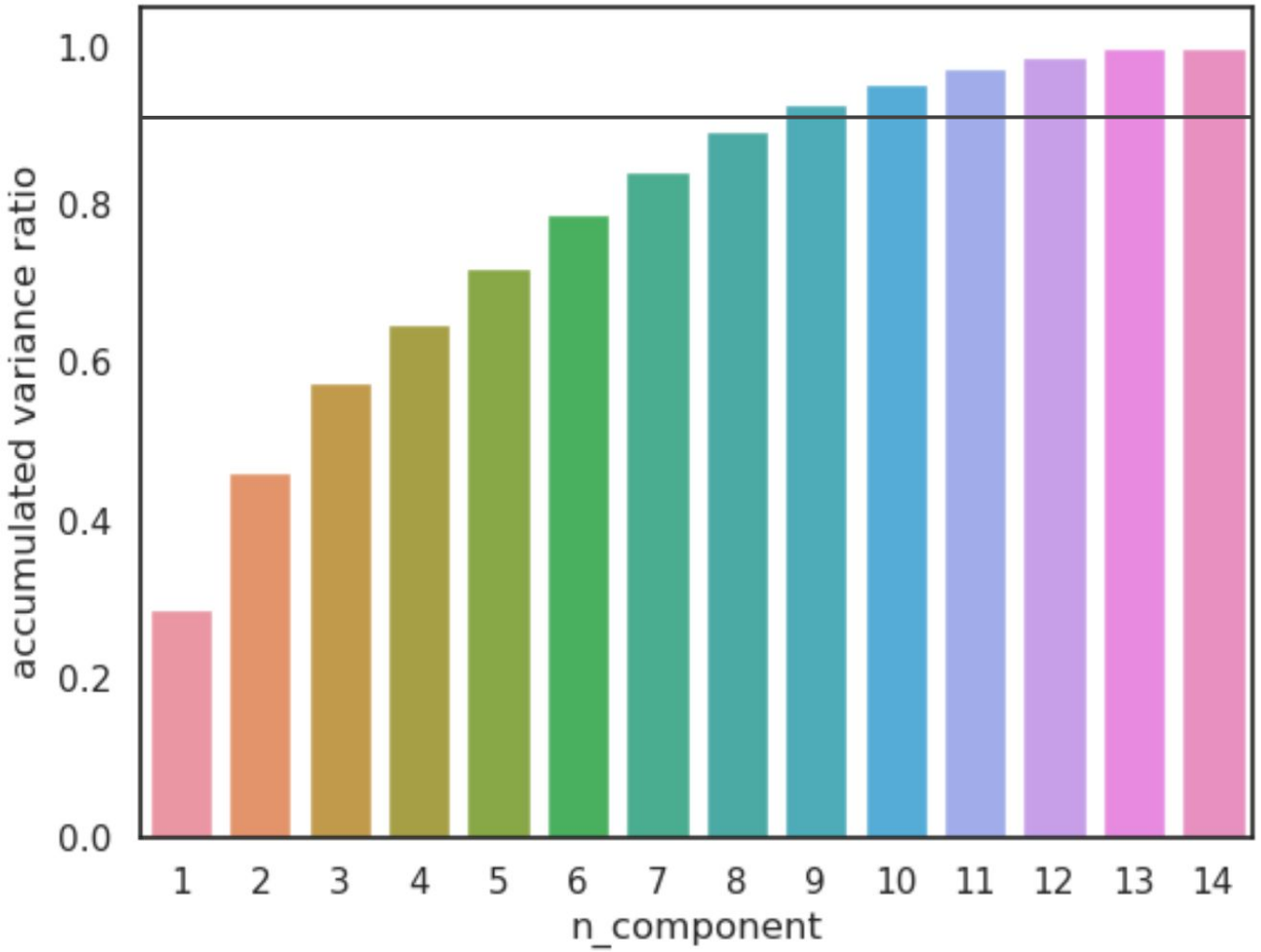
Recommend popular courses according to the preferences of other users within the same group.

Flowchart of clustering-based recommender system

User/Item feature engineering

Extract features from users such as enrollments or genres

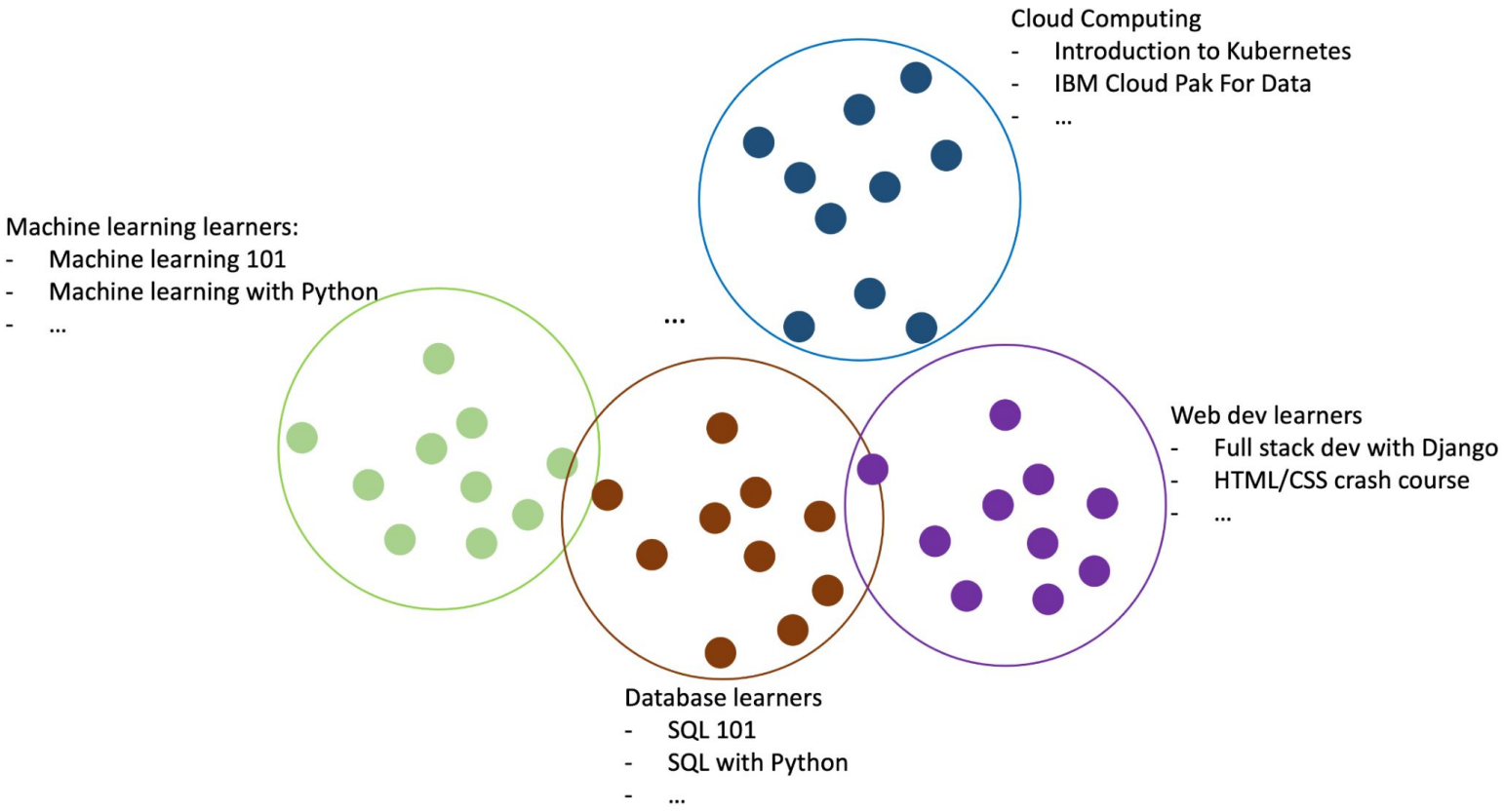
Apply PCA on user profile feature vectors to reduce dimensions



Flowchart of clustering-based recommender system

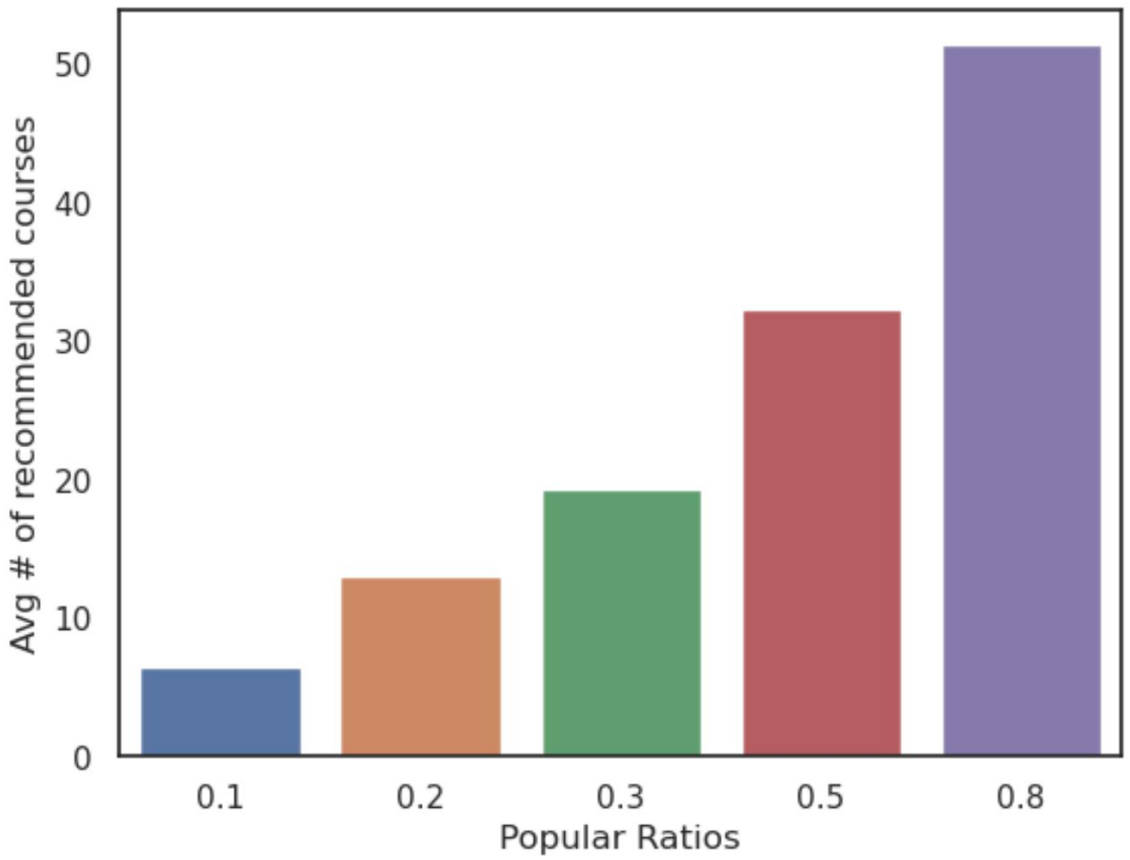
Group users

Divide users into several groups based on their enrollments or genres



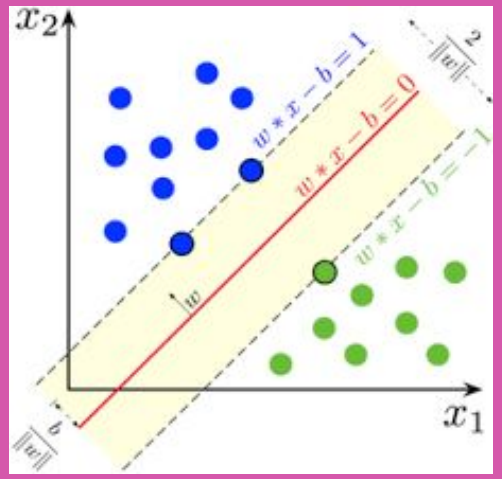
Evaluation results of clustering-based recommender system

In the lower left chart, we vary the **popular ratio range from 0.1 to 0.8**, **maintaining the PCA features at 9 and n_clusters at 30**. To attain approximately 10 recommended courses per user, **we set the popular ratio to 0.2**. The popular ratio is defined as the ratio of all enrollments within the cluster to which the user belongs.

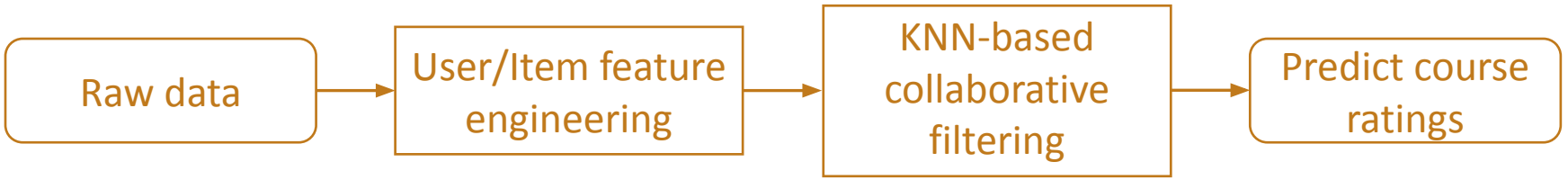


	COURSE_ID	recommended_count
0	DS0101EN	9372
1	PY0101EN	9255
2	BD0101EN	8882
3	BD0111EN	8042
4	ML0115EN	7553
5	DS0103EN	7293
6	ML0101ENv3	7235
7	DS0105EN	6951
8	DA0101EN	6853
9	BD0211EN	6782

Collaborative-filtering Recommender System using Supervised Learning



Flowchart of KNN based recommender system



Course genres dataframe:
`course_id`,
`title`,
`course_rating`

User dataframe:
`user_id`,
`enrollment`,
`course_ratings`

Extract features from users such as enrollments or genres

Build User-item interaction matrix

Perform KNN-based collaborative filtering on the user-item interaction matrix

Compute prediction ratings using the trained KNN model.

Flowchart of clustering-based recommender system

User/Item feature engineering

Extract features from users such as enrollments or genres

Build User-item interaction matrix

User-Item interaction matrix

	Machine Learning With Python	Machine Learning 101	Machine Learning Capstone	SQL with Python	Python 101
...	...	...	...	...	
user2	3.0	3.0	3.0	3.0	3.0
user3	2.0	3.0	3.0	2.0	
user4	3.0	3.0	2.0	2.0	3.0
user5	2.0	3.0	3.0		
user6	3.0	3.0	?		3.0
...	...	...	...	...	

Similar users

Predict the rating of user *user6* to item *Machine Learning Capstone*

Flowchart of KNN based recommender system

Predict course ratings

Compute prediction ratings using the trained KNN model.

The predicted rating of user u to item i, $\hat{r}_{ui}$ is given by:

User-based collaborative filtering:

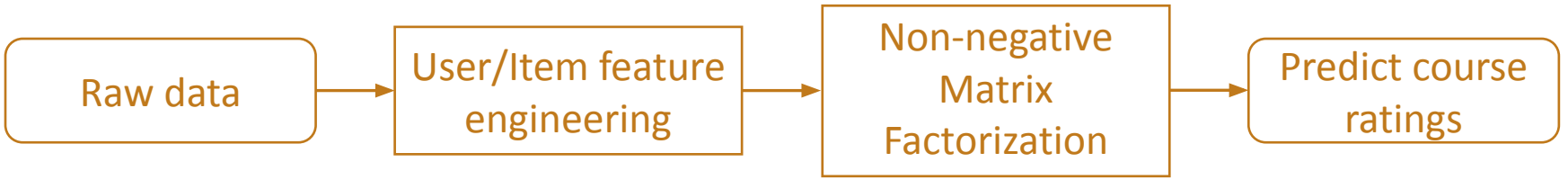
$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} \text{similarity}(u, v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} \text{similarity}(u, v)}$$

Item-based collaborative filtering:

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} \text{similarity}(i, j) \cdot r_{uj}}{\sum_{j \in N_u^k(i)} \text{similarity}(i, j)}$$

Here $N_i^k(u)$ notates the nearest k neighbors of u.

Flowchart of NMF based recommender system



Course genres dataframe:
`course_id`,
`title`,
`course_rating`

User dataframe:
`user_id`,
`enrollment`,
`course_ratings`

Choose the Number of Components (Rank)

Initialize Factors W and H

Derive the latent user and item features through Non-negative Matrix Factorization (NMF)

Dot-product the latent features of users and items to get an estimated rating

Flowchart of NMF based recommender system

Non-negative Matrix Factorization

Extract the latent user and item features

User-item interaction matrix: **A** 10000 x 100

	item1	...	item100
user1	...	...	
user2	3.0	3.0	3.0
user3	2.0	2.0	-
user4	3.0	2.0	3.0
user5	2.0	-	-
user6	3.0	-	3.0
...	...	...	

≈

User matrix: **U** 10000 x 16

	feature1	...	feature16
user1	...	...	...
user2	...	...	...
user3	...	...	...
user4	...	...	...
...	...	...	...
...	...	...	...
user6	...	...	...

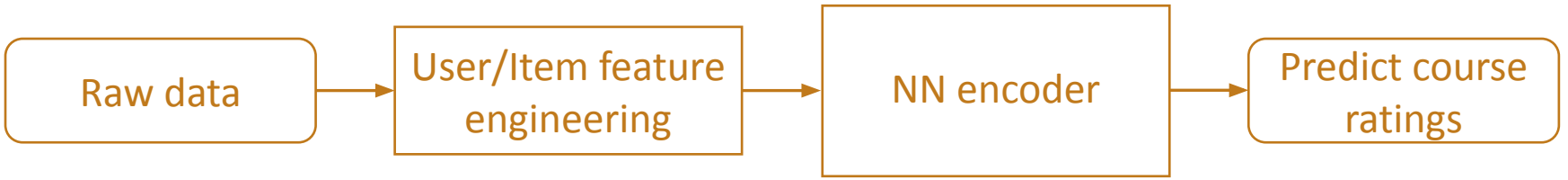
X

Item matrix: **I** 16 x 100

	item1	...	item100
feature1	...	...	...
feature2	...	...	...
...	...	...	...
feature16	...	...	...

$$\sum_{r_{jk} \in train} \left(r_{jk} - \hat{r}_{jk}\right)^2, \text{ where } \hat{r}_{ij} \text{ is the dot product of } u_j^T \text{ and } i_k: \quad \hat{r}_{jk} = u_j^T i_k$$

Flowchart of Neural Network Embedding based recommender system



Course genres dataframe:
`course_id`,
`title`,
`course_rating`

User dataframe:
`user_id`,
`enrollment`,
`course_ratings`

Extract features from users and items such as genres

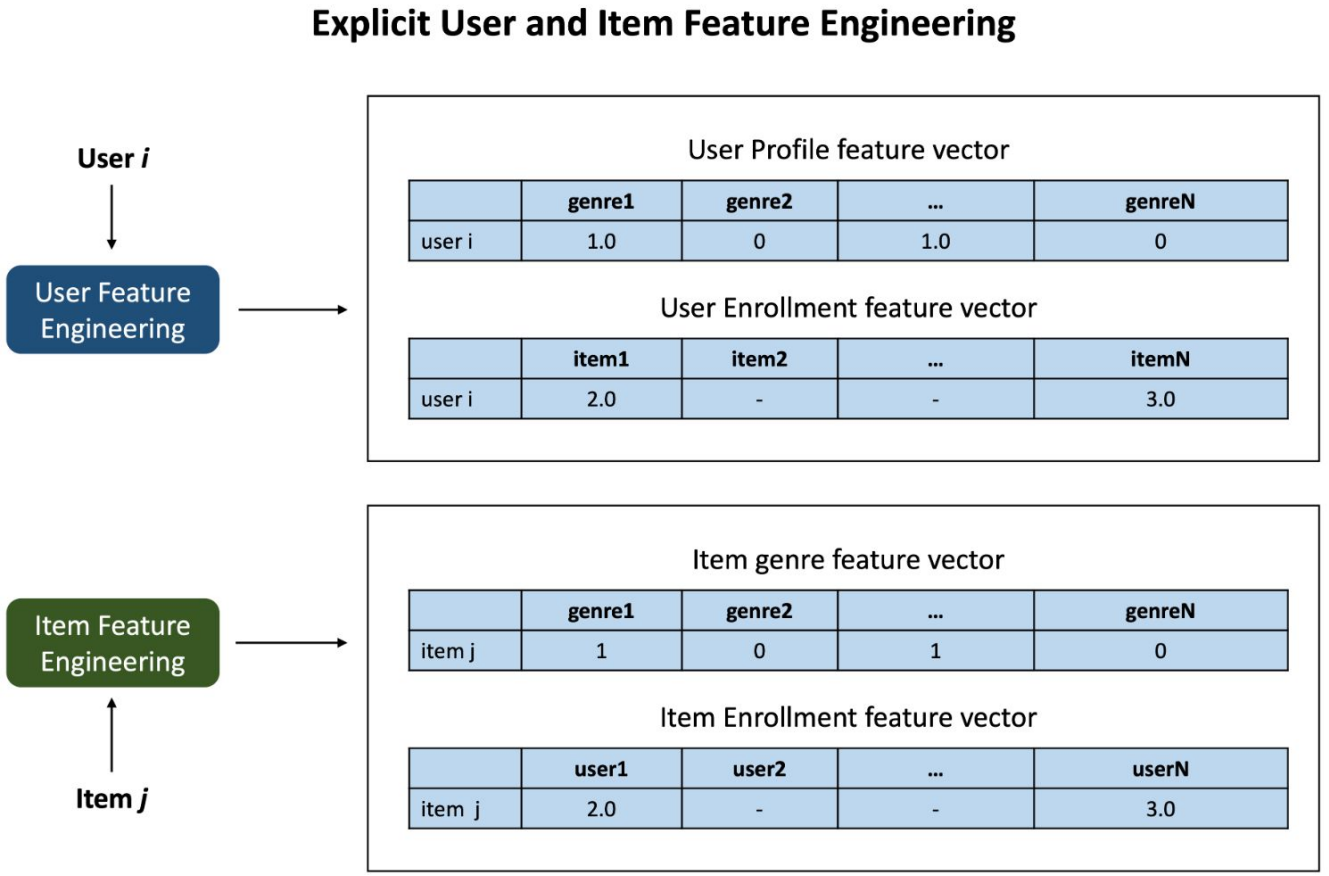
Extract the latent user and item features using Neural Network(NN)

Dot-product the latent features of users and items to get an estimated rating

Flowchart of Neural Network Embedding based recommender system

User/Item Feature engineering

Extract features from users and items such as genres



Flowchart of Neural Network Embedding based recommender system

NN encoder

Extract the latent user and item features using Neural Network(NN)

User-item interaction matrix: **A** 10000 x 100

	item1	...	item100
user1	...	...	
user2	3.0	3.0	3.0
user3	2.0	2.0	-
user4	3.0	2.0	3.0
user5	2.0	-	-
user6	3.0	-	3.0
...	...	...	

≈

User matrix: **U** 10000 x 16

	feature1	...	feature16
user1	...	...	...
user2	...	...	...
user3	...	...	...
user4	...	...	...
...	...	...	...
...	...	...	...
user6	...	...	...

X

Item matrix: **I** 16 x 100

	item1	...	item100
feature1	...	...	...
feature2	...	...	...
...	...	...	...
feature16	...	...	...

$$\sum_{r_{jk} \in train} \left(r_{jk} - \hat{r}_{jk}\right)^2, \text{ where } \hat{r}_{ij} \text{ is the dot product of } u_j^T \text{ and } i_k: \quad \hat{r}_{jk} = u_j^T i_k$$

Flowchart of Neural Network Embedding based recommender system

NN encoder

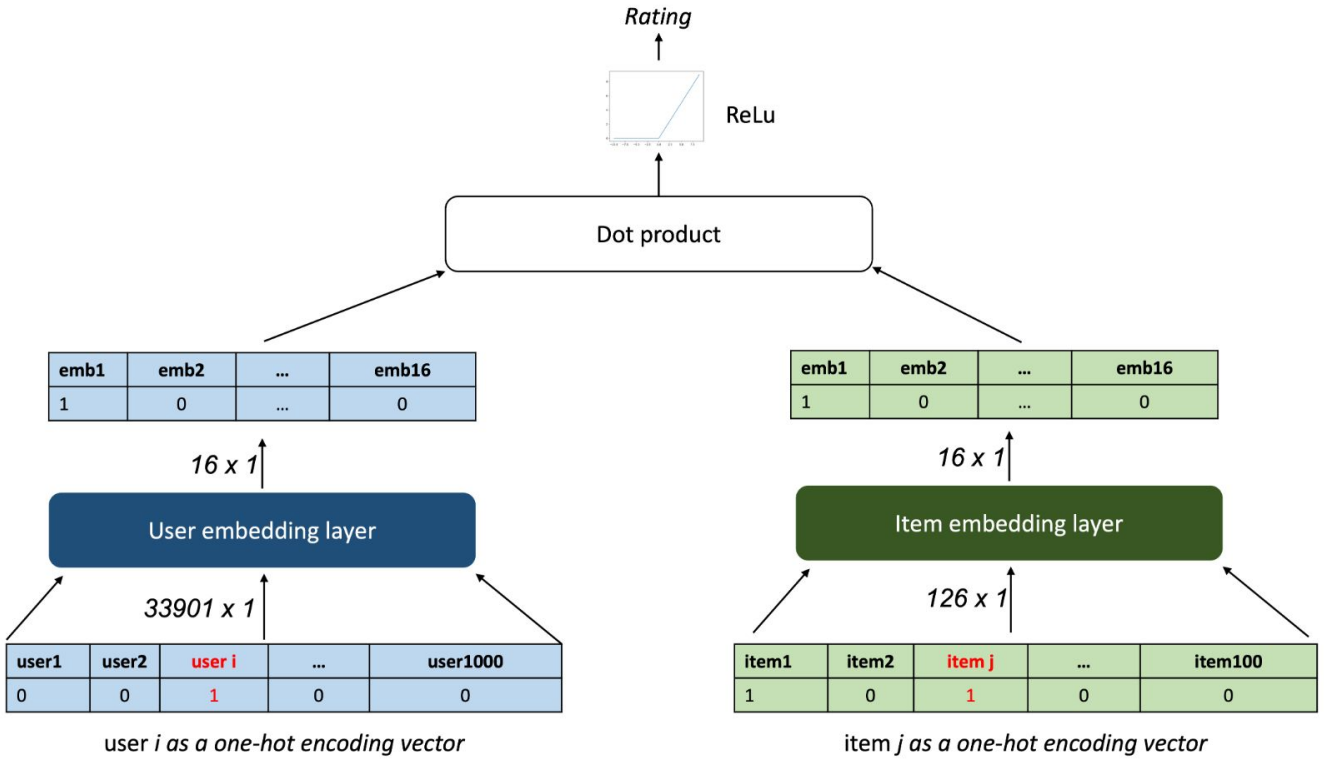
Extract the latent user and item features using Neural Network(NN)

```
Model: "recommender_net"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 user_embedding_layer (Embed  multiple                 542416
 ding)

 user_bias (Embedding)       multiple                  33901

 item_embedding_layer (Embed  multiple                 2016
 ding)

 item_bias (Embedding)       multiple                  126

=================================================================
Total params: 578,459
Trainable params: 578,459
Non-trainable params: 0
_________________________________________________________________
```

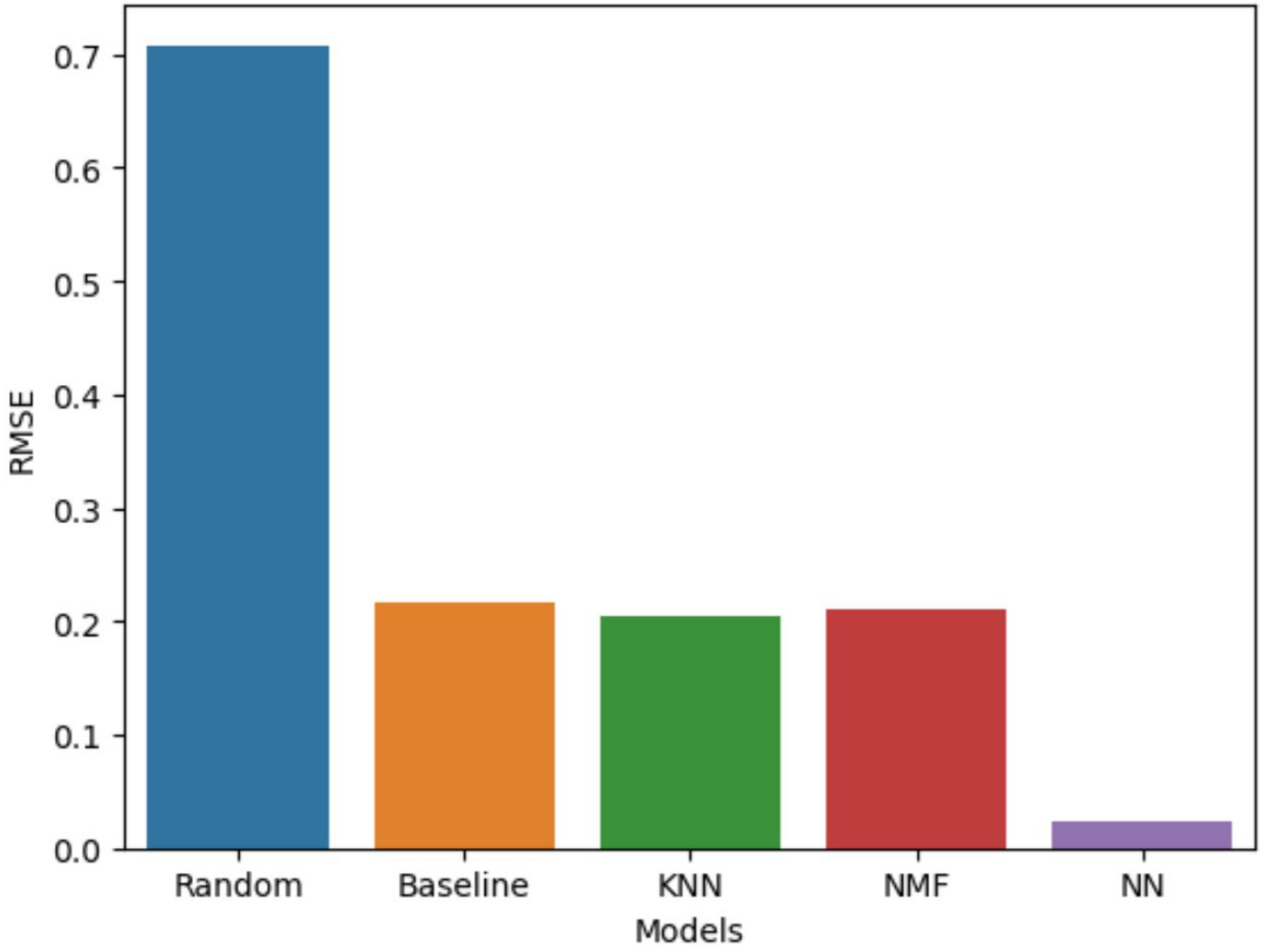
Flowchart of Neural Network Embedding based recommender system

Predict course ratings

Dot-product the latent features of users and items to get an estimated rating



Compare the performance of collaborative-filtering models



Conclusions

- We can recommend courses based on the similarity in genre in their profiles and suggest courses or based on the similarity in the enrollments of the users and suggest courses.
- Grouping users according to the genre in their profiles allows us to recommend courses that are popular within the same cluster.
- Applying PCA to user profile feature vectors can decrease dimensions, consequently reducing computational power requirements.
- Course recommendations can be generated by predicting course ratings through KNN-based collaborative filtering.
- Deriving latent user and item features through Non-negative Matrix Factorization (NMF) or Neural Network (NN) allows us to estimate course ratings and, hence, recommend courses.

Outlook - Possible issues

- **Cluster Interpretability:** Understanding and interpreting the meaning of clusters can be difficult. Without clear domain knowledge, it might be challenging to explain why certain items or users are grouped together.
- **Dynamic Nature of Data:** User preferences and item popularity can change over time. Unsupervised models might struggle to adapt to dynamic shifts in user behavior and preferences.
- **Lack of Personalization:** Traditional clustering methods might not capture individual user preferences well, leading to less personalized recommendations.
- **Evaluation Metrics:** Selecting appropriate evaluation metrics for unsupervised recommendation systems is challenging. Defining what constitutes a "good" clustering can be subjective.

Outlook -Possible solutions:

- **Cluster Interpretability:** Gather user-generated course ratings and construct an interpretable supervised machine learning model, such as a decision tree, to elucidate the characteristics of the identified clusters.
- **Dynamic Nature of Data:** Periodically retrain the model with updated data to adapt to changes in user preferences and item popularity over time.
- **Lack of Personalization:** Combine clustering with user-specific features or collaborative filtering methods to enhance the personalization of recommendations.
- **Evaluation Metrics:** Define and use appropriate evaluation metrics based on the specific goals of the recommendation system. Incorporate user feedback and conduct A/B testing to assess the real-world impact of recommendations. Additionally, assessing supervised learning score prediction accuracy can offer a valuable method for evaluating the system.

Appendix

Data source:
https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-ML321EN-SkillsNetwork/labs/datasets/sim.csv
https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-ML321EN-SkillsNetwork/labs/datasets/user_profile.csv
https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-ML321EN-SkillsNetwork/labs/datasets/course_genre.csv

Courses:
https://www.coursera.org/learn/ibm-unsupervised-machine-learning/home

Jupyter notebooks:
https://github.com/r95222023/IBM-Machine-Learning-Professional-Certificate/tree/main/Machine%20Learning%20Capstone