Introduction

- This project implements and deploys several AI course Recommender Systems using Streamlit. The final codebase and the deployment can be seen here:

  https://github.com/mxagar/course_recommender_streamlit

  https://ai-course-recommender-demo.herokuapp.com/

- The application is the final/capstone project of the IBM Machine Learning Professional Certificate offered by IBM & Coursera.
- The most important dataset of the project is a ratings dataframe in which we have 233,306 entries with a **user id**, **course id** and an associated **rating** (2: audited, 3: finished). In addition, to that, we also have another dataset with 307 course entries, each with course descriptions and one-hot encoded genre values (14 genres/topics). From those, we can
  - build user profiles,
  - compute user and course similarities,
  - infer latent user and course features,
  - build recommender models, both content-based and with collaborative filtering.

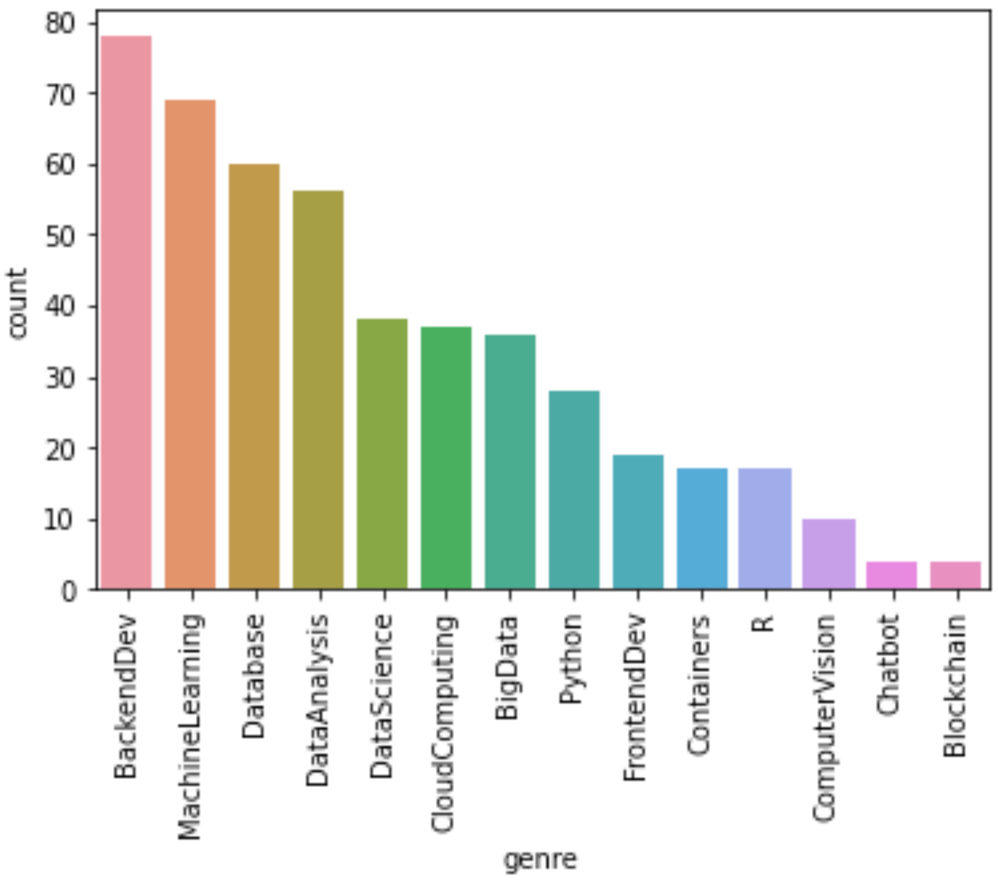
Introduction, contd.

- All in all, the following models are created:
  - Content-based: user and course features (i.e., genres/topics) are known:
    - Course Similarity: course descriptors formed by bags-of-words of course descriptions.
    - User Profile: user descriptors formed by the genre preferences derived from course ratings.
    - Clustering: K-means applied to user profile vectors to discover most common courses per cluster.
    - Clustering with PCA: equivalent to the previous, but with dimensionality reduction.
  - Collaborative Filtering: user and course features (i.e., genres/topics) are not known, or they are inferred:
    - KNN: course similarities based on the ratings provided by all users.
    - NMF: ratings table factorization to discover latent course and user features.
    - Neural Network (NN): user and course pairs mapped to ratings with intermediate embeddings.
    - Regression with Embedding Features: embeddings from NN used to regress ratings.
    - Classification with Embedding Features: embeddings from NN used to infer rating classes.

Exploratory Data Analysis

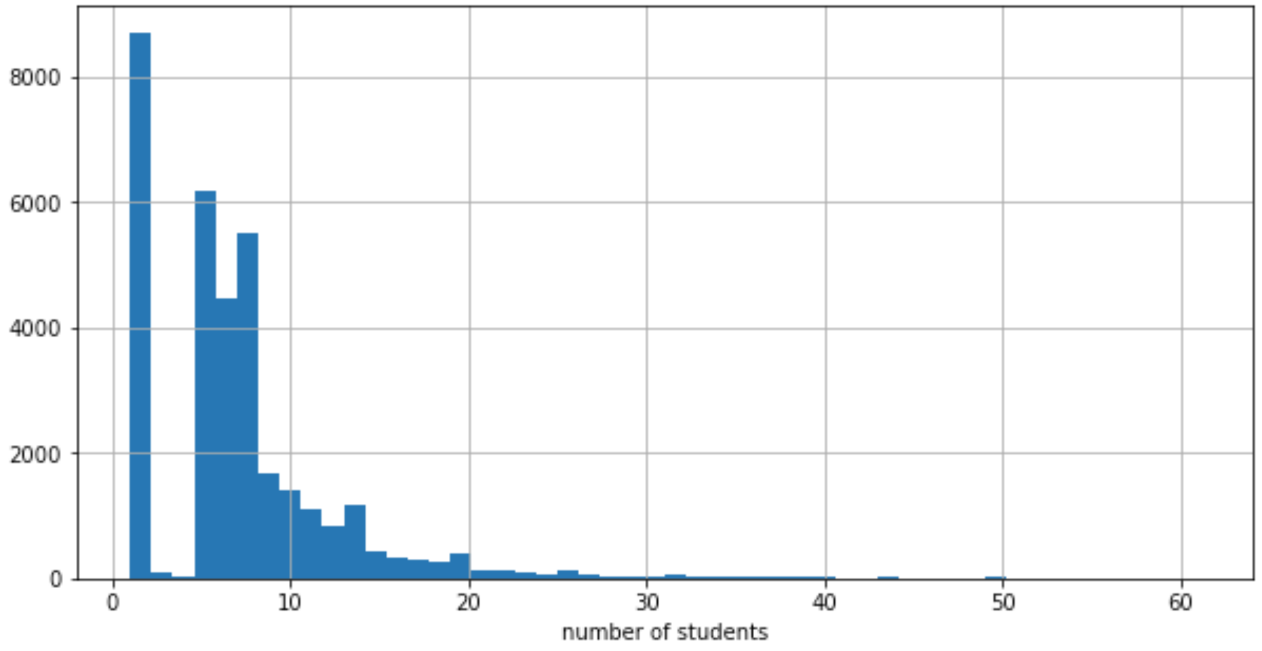
Course counts per genre

- There are 307 courses in total
- 14 genres or topics have been manually defined
- Each course can deliver contents related to several topics
- In the figure, the distribution of courses that provide contents on each topic
- 5 most common genres: backend development, machine learning, databases, data analysis and data science.



Course enrollment distribution

- Each student can attend several courses and each course can have many students.
- The course enrollment distribution seems to be a power law: most of the courses have one student enrolled, the number of courses with many students decreases rapidly with the number of students in them.
- There is a gap for the courses with 2-5 students: there are barely no such courses.



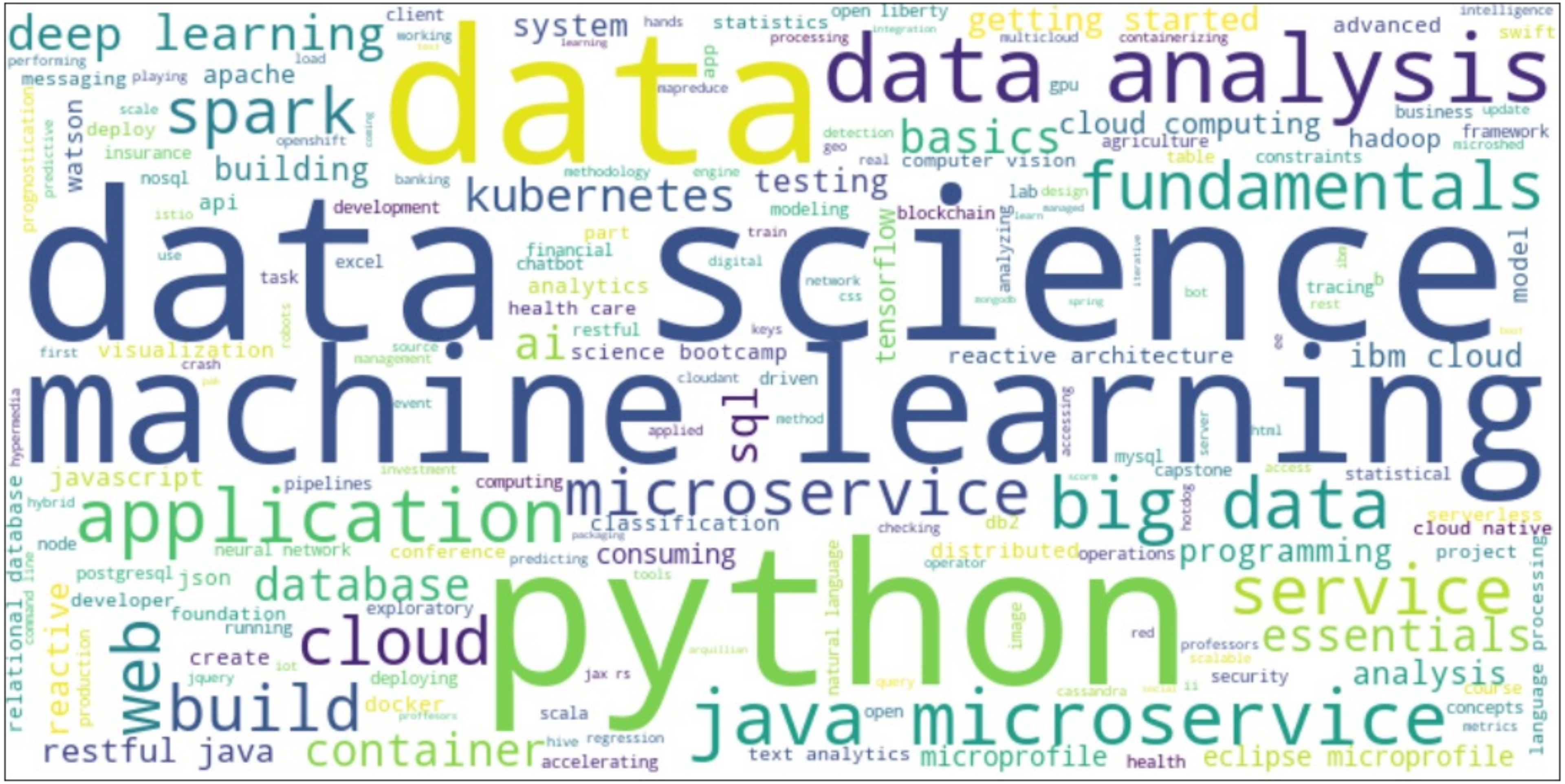
Twenty most popular courses

- The figure shows the 20 most popular courses, by the number of enrollments.
- The courses are in line with the most popular topics shown before.

	course	enrollments	TITLE
0	PY0101EN	14936	python for data science
1	DS0101EN	14477	introduction to data science
2	BD0101EN	13291	big data 101
3	BD0111EN	10599	hadoop 101
4	DA0101EN	8303	data analysis with python
5	DS0103EN	7719	data science methodology
6	ML0101ENv3	7644	machine learning with python
7	BD0211EN	7551	spark fundamentals i
8	DS0105EN	7199	data science hands on with open source tools
9	BC0101EN	6719	blockchain essentials
10	DV0101EN	6709	data visualization with python
11	ML0115EN	6323	deep learning 101
12	CB0103EN	5512	build your own chatbot
13	RP0101EN	5237	r for data science
14	ST0101EN	5015	statistics 101
15	CC0101EN	4983	introduction to cloud
16	CO0101EN	4480	docker essentials a developer introduction
17	DB0101EN	3697	sql and relational databases 101
18	BD0115EN	3670	mapreduce and yarn
19	DS0301EN	3624	data privacy fundamentals

Word cloud of course titles

The word cloud visually shows the importance (i.e., number of occurrences) of specific keywords in the titles of the courses.

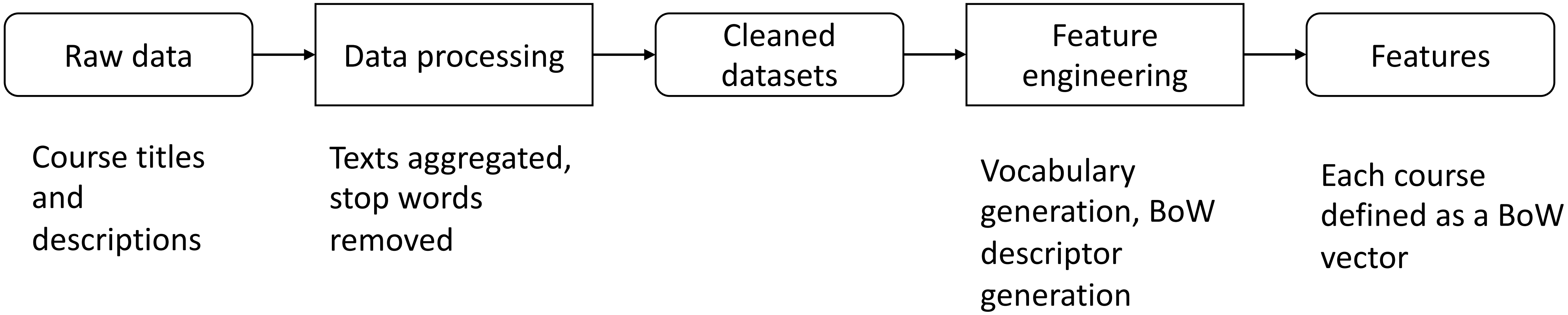


Content-based Recommender System

When user and course features (i.e., genres/topics) are known

Flowchart of content-based recommender system using course similarity

- Course similarities are built from course text descriptions using Bags-of-Words (BoW). A similarity value is the projection of a course descriptor vector in the form of a BoW on another, i.e., the cosine similarity between both. Given the selected courses, the set of courses with the highest similarity value are found.
- Flow chart of the data processing:



Evaluation results of course similarity based recommender system

Key evaluation questions:

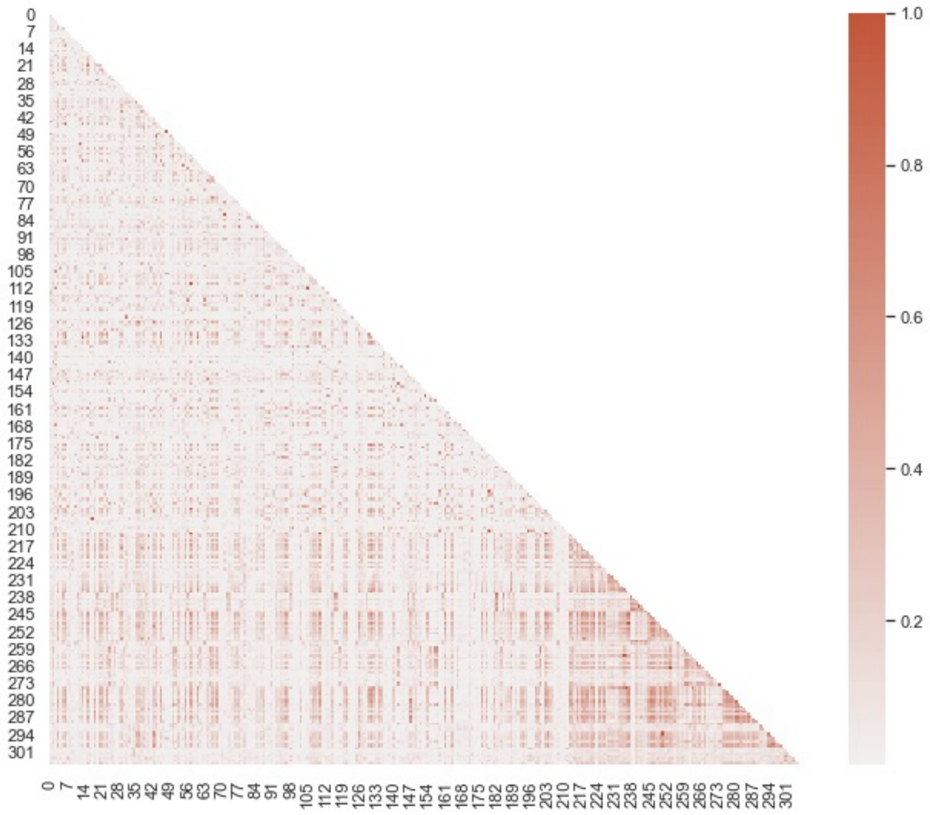
```
# On average, how many new courses have been recommended per test user?
res_df.groupby('USER')['COURSE_ID'].count().mean()
```

```
11.573753814852493
```

```
# What are the most frequently recommended courses?
# Return the top-10 commonly recommended courses across all test users.
res_df.groupby('COURSE_ID').count()['USER'].sort_values(ascending=False)[:10]
```

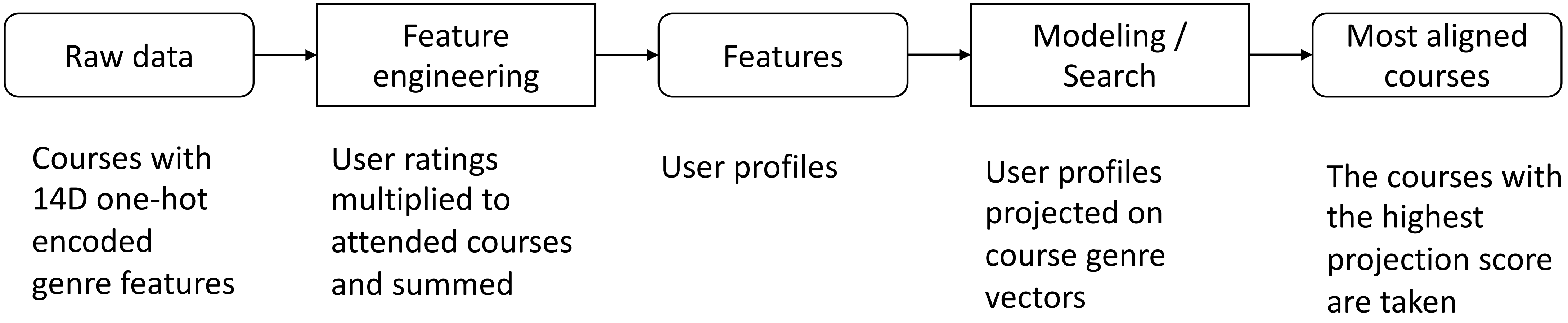
```
COURSE_ID
excourse62    579
excourse22    579
DS0110EN      562
excourse65    555
excourse63    555
excourse72    551
excourse68    550
excourse74    539
excourse67    539
BD0145EN      506
Name: USER, dtype: int64
```

Course similarity heatmap (1: very similar)



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

- Courses have a genre descriptor vector which encodes all the topics covered by them. User profiles can be built by summing the user course descriptors scaled by the ratings given by the user. Then, for a target user profile, the unselected courses that are most aligned with it can be found using the cosine similarity (i.e., dot product) between the profile and the courses. Finally, the courses with the highest scores are provided.
- Flow chart of the data processing and modelling:



Evaluation results of user profile-based recommender system

Key evaluation questions:

```
# On average, how many new courses have been recommended per test user?
res_df.groupby('USER')['COURSE_ID'].count().mean()
```

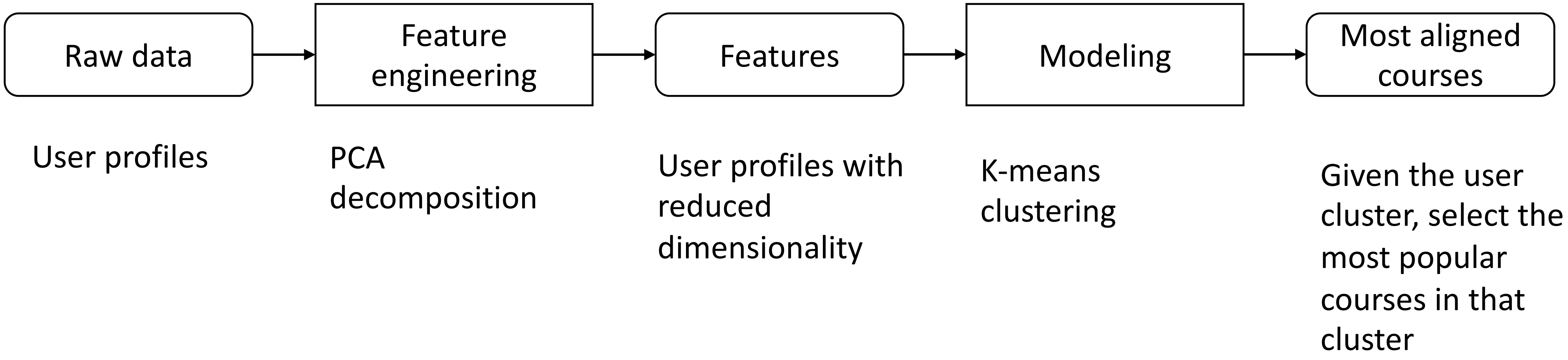
```
61.81828703703704
```

```
# What are the most frequently recommended courses?
# Return the top-10 commonly recommended courses across all test users.
res_df.groupby('COURSE_ID').count()['USER'].sort_values(ascending=False)[:10]
```

```
COURSE_ID
TA0106EN       608
GPXX0IBEN      548
excourse22     547
excourse21     547
ML0122EN       544
excourse06     533
excourse04     533
GPXX0TY1EN     533
excourse31     524
excourse73     516
Name: USER, dtype: int64
```

Flowchart of clustering-based recommender system

- Courses have a genre descriptor vector which encodes all the topics covered by them. User profiles can be built by summing the user course descriptors scaled by the ratings given by the users. Then, those users can be clustered according to their profile. This approach provides with the courses most popular within the user cluster.
- Additionally, user profile descriptors can be transformed to their principal components, taking only a subset of them, enough to cover a percentage of the total variance, selected by the user.
- Flow chart of the data processing and modelling:



Evaluation results of clustering-based recommender system

Key evaluation questions:

```
# On average, how many new courses have been recommended per test user?
recommended_courses.groupby('user')['course'].count().mean()
```

```
2.9663975492229633
```

```
# What are the most frequently recommended courses?
# Return the top-10 commonly recommended courses across all test users.
recommended_courses.groupby('course').count()['user'].sort_values(ascending=False)[:10]
```

```
course
PY0101EN      18517
DS0101EN      15561
BD0101EN      14381
ML0115EN       5249
DV0101EN       5223
ML0101ENv3     5213
DA0101EN       5195
BD0115EN       4079
BD0211EN       4041
BD0111EN       4019
Name: user, dtype: int64
```

Collaborative-filtering Recommender System

When user and course features (i.e., genres/topics) are not known, or they are inferred

Flowchart of KNN based recommender system

- Given the ratings dataframe, course columns are treated as course descriptors, i.e., each course is defined by all the ratings provided by the users. With that, a course similarity matrix is built using the cosine similarity. Then, for the set of selected courses, the most similar ones are suggested.
- Flow chart of the data processing and modelling:

(Dense) User ratings

Pivot ratings to create a sparse ratings matrix

Course descriptors consisting of all user ratings, i.e., course ratings columns

Compute similaty matrix between sparse course vectors

Select the most similar courses given a set of selected courses

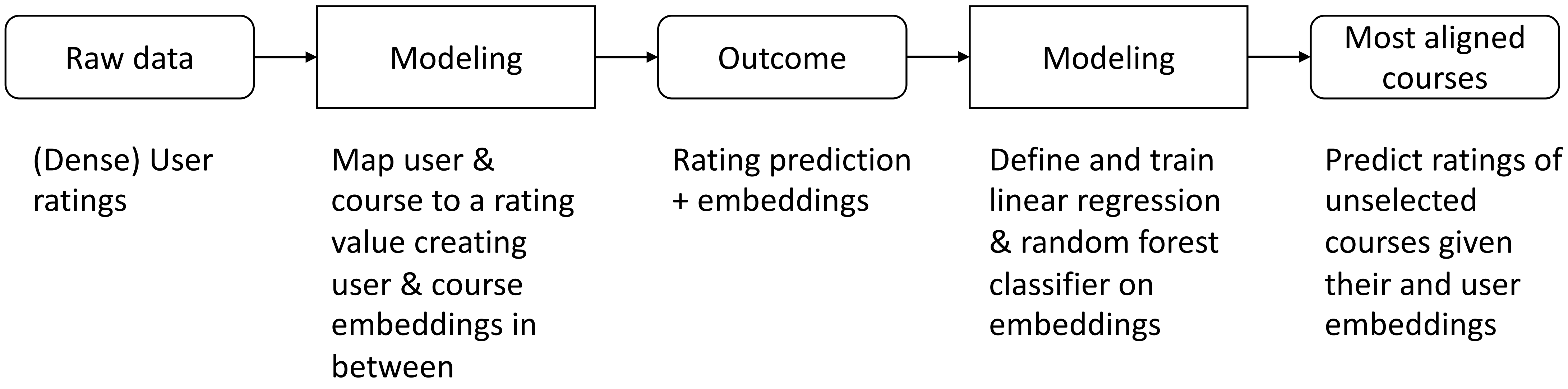
Flowchart of NMF based recommender system

- Non-Negative Matrix Factorization is performed: given the ratings dataset which contains the rating of each user for each course (sparse notation), the matrix is factorized as the multiplication of two lower rank matrices. That lower rank is the size of a latent space which represents discovered inherent features (e.g., genres). With the factorization, the ratings of unselected courses are predicted by multiplying the lower rank matrices, which yields the approximate but complete user-course rating table.
- Flow chart of the data processing and modelling:

(Dense) User ratings	Pivot ratings to create a sparse ratings matrix	Sparse user ratings	Apply NMF decomposition	Multiply the factorized matrices; for each unselected course, consider it if the predicted rating is high

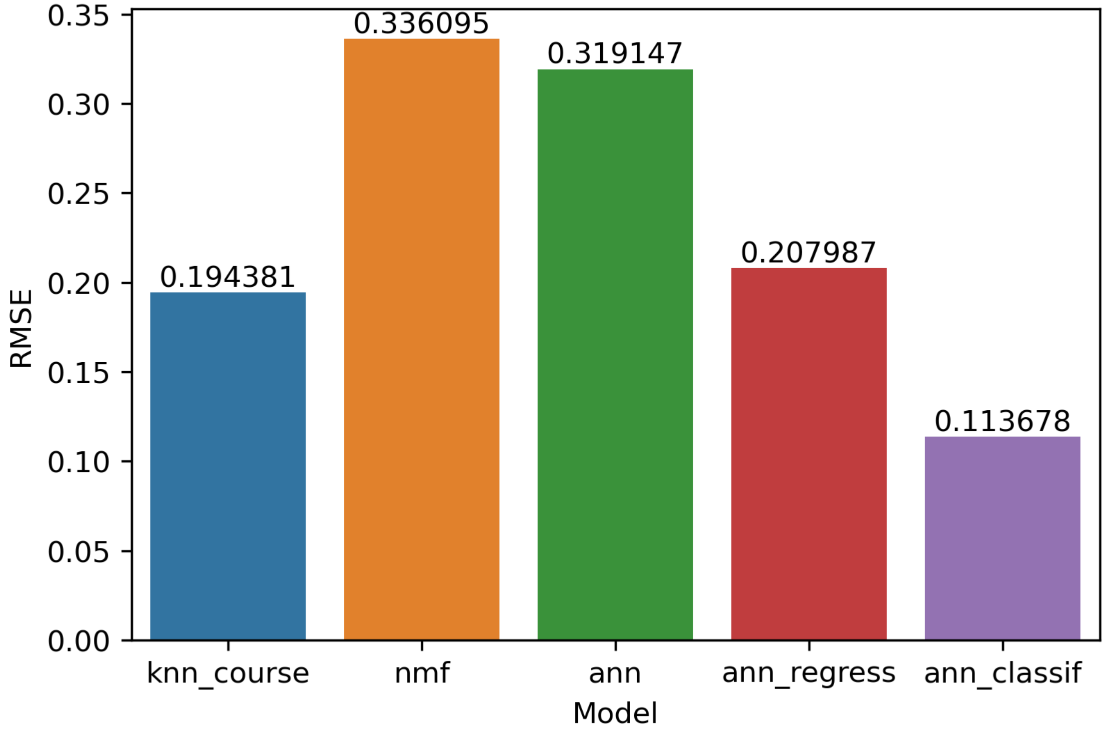
Flowchart of Neural Network Embedding based recommender system

- An Artificial Neural Network (ANN) which maps users and courses to ratings is defined and trained. If the user is in the training set, the ratings for unselected courses can be predicted. However, the most interesting part of this approach consists in extracting the user and course embeddings from the ANN for later use. An embedding vector is a continuous N-dimensional representation of a discrete object (e.g., a user).
- The user and item embeddings extracted from the ANN are used to build a linear regression model and a random forest classifier which predict the rating given the embedding of a user and a course.
- Flow chart of the data processing and modelling:

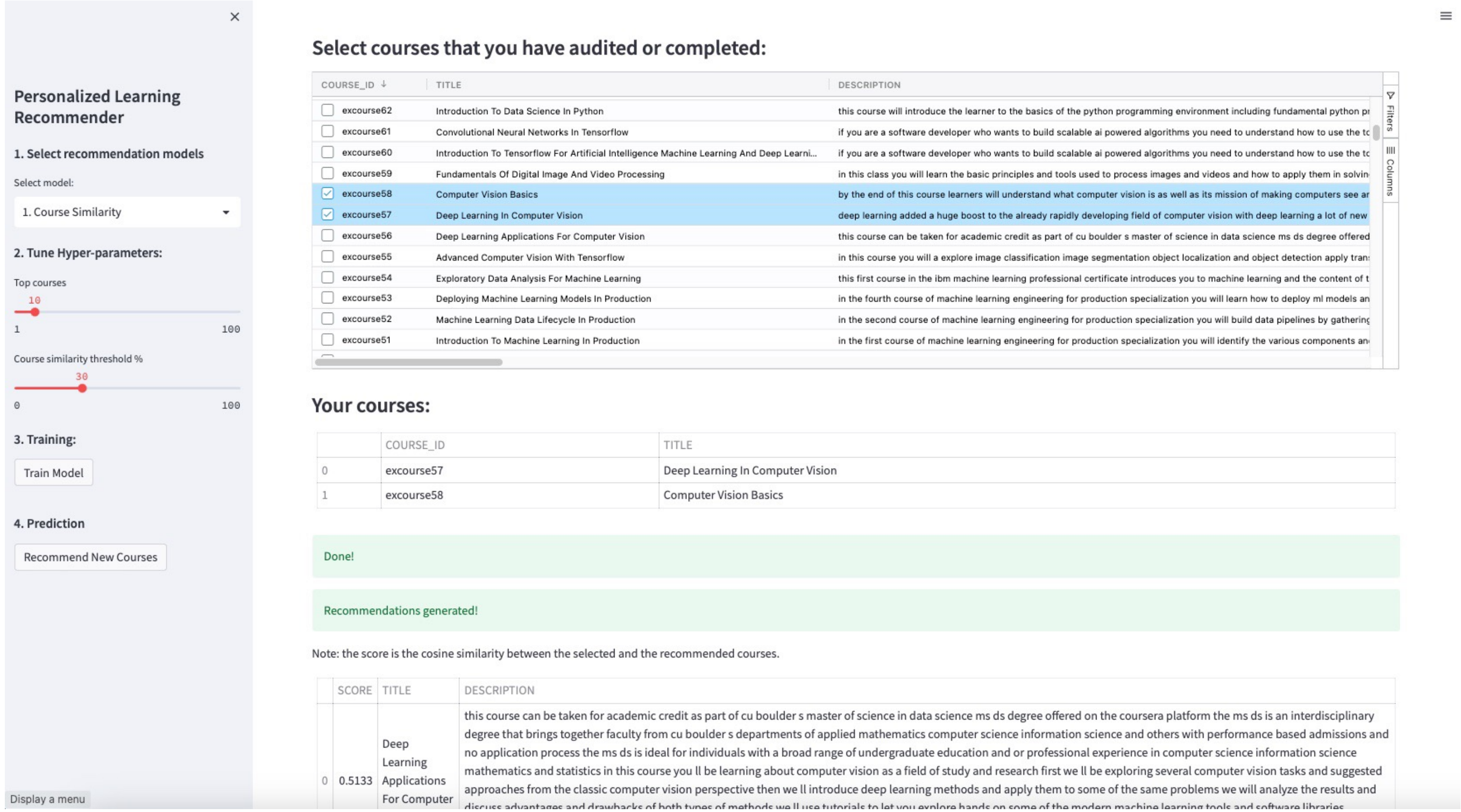


Comparing the performance of collaborative-filtering models

- Root mean square error for the test split is shown in the figure
- The most known method NMF yields the worst result
- The random forest classifier with ANN embeddings yields the best result, followed by the KNN model created with the Surprise library, based on course neighboring (not users)



The course recommender system app with Streamlit deployed on Heroku



https://ai-course-recommender-demo.herokuapp.com

https://github.com/mxagar/course_recommender_streamlit

Summary and conclusions

- Eight recommender systems have been created and deployed; these can be classified in two groups:
  - Content-based: when user and course features (i.e., genres/topics) are known
  - Collaborative Filtering: when user and course features (i.e., genres/topics) are not known, or they are inferred
- **Content-based** systems work efficiently and provide similar results, but they require (manual) genre characterization for users and courses/items
- **Collaborative Filtering** systems are based on the assumption that there is a relationship between users and items, so that we can discover the latent features that reveal user preferences
- The collaborative system which seems to best predict the ratings is the random forest classifier that maps user and course embeddings (from the ANN) to rating classes (2 or 3). However:
  - The training time is the longest
  - The new users for whom we want to predict (and their example ratings) must be trained with the system so that they have an embedding representation

Appendix

- Github repository: https://github.com/mxagar/course_recommender_streamlit
  - Notebooks: https://github.com/mxagar/course_recommender_streamlit/blob/main/notebooks
  - App:
    - https://github.com/mxagar/course_recommender_streamlit/blob/main/recommender_app.py
    - https://github.com/mxagar/course_recommender_streamlit/blob/main/backend.py
- Deployment URL: https://ai-course-recommender-demo.herokuapp.com/