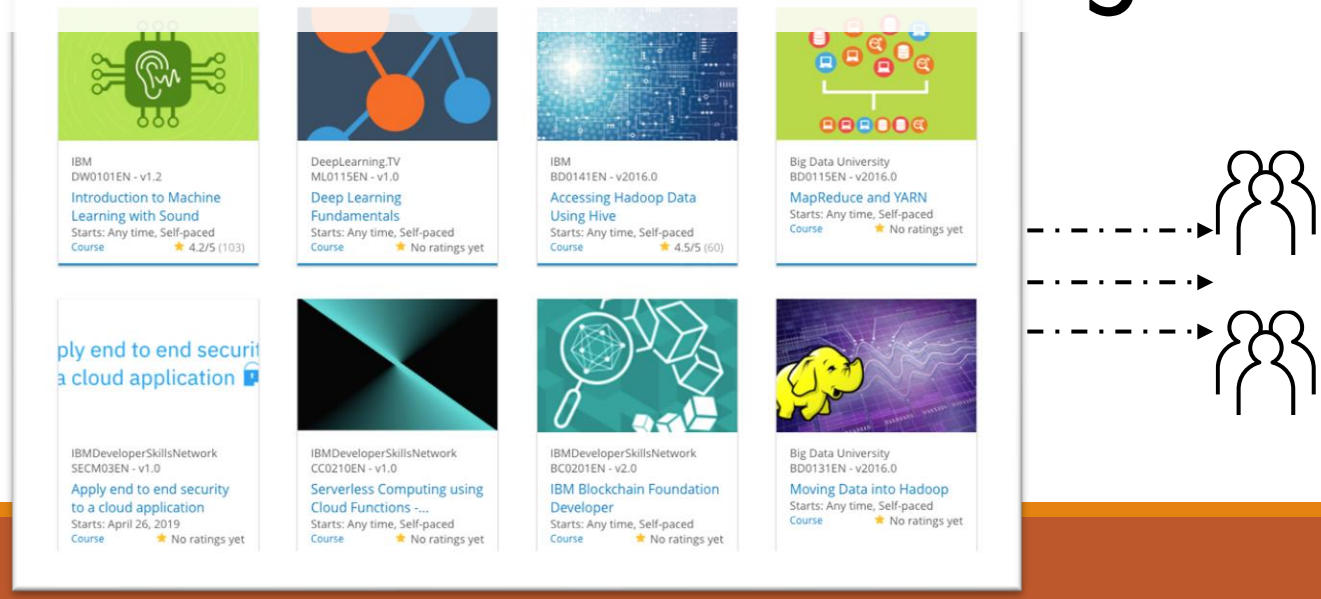
# Build a Personalized Online Course Recommender System with Machine Learning

MUHAMMAD AMIN GHIAS

12TH JULY 2023



# 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

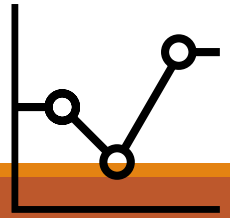
- Project background and context

Online courses are becoming popular these days and fining a suitable online course is a tedious task. In this project we have data of online courses its title genres and the users which have taken these courses. We would analyze this data and help in selection of courses for users based on their preferences and past history

- Problem states and hypotheses

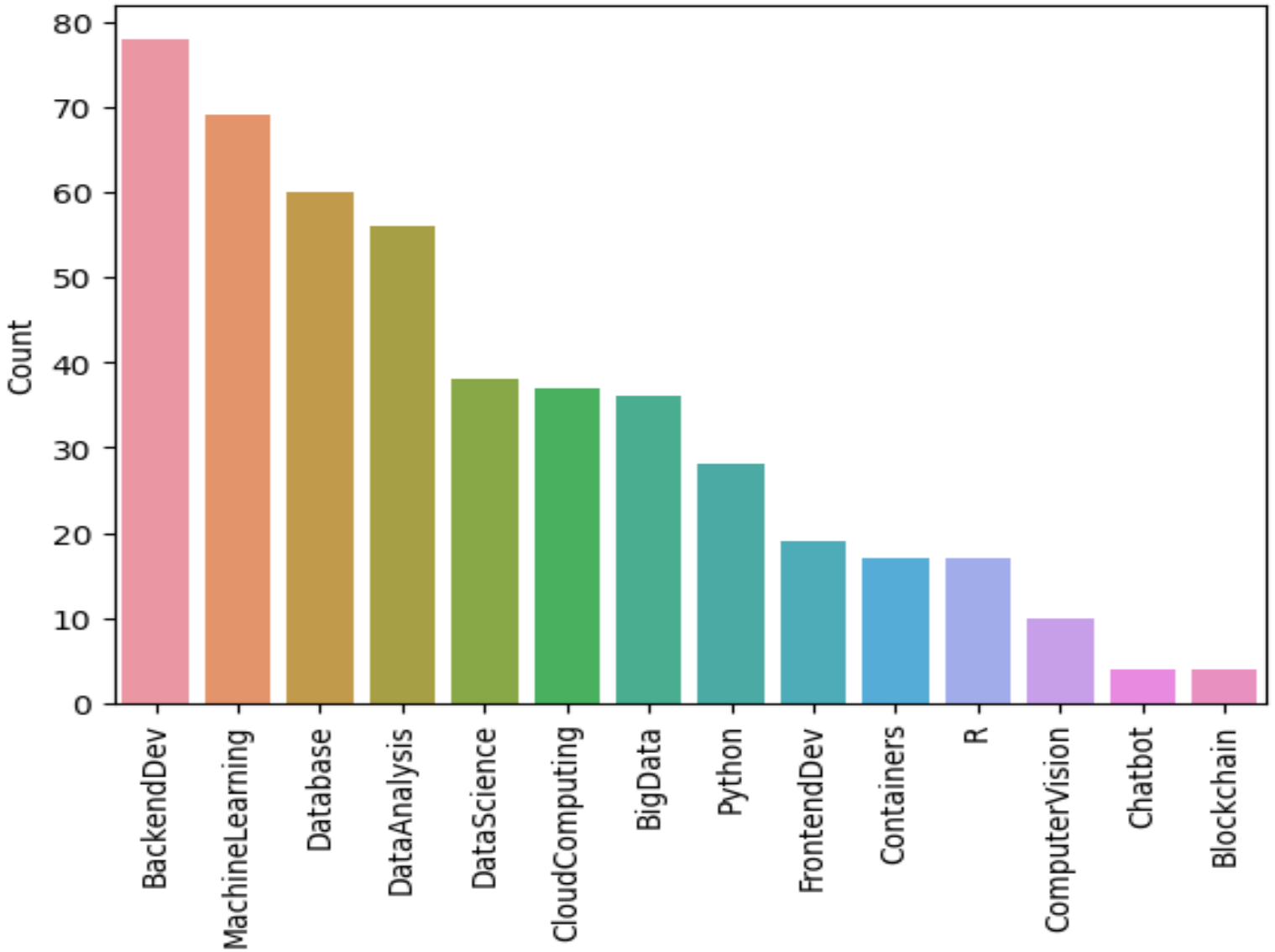
There are so many similar online courses that it gets difficult to select from many courses which is suitable for a user. To overcome this we will build a recommender system to help in course selection

# Exploratory Data Analysis



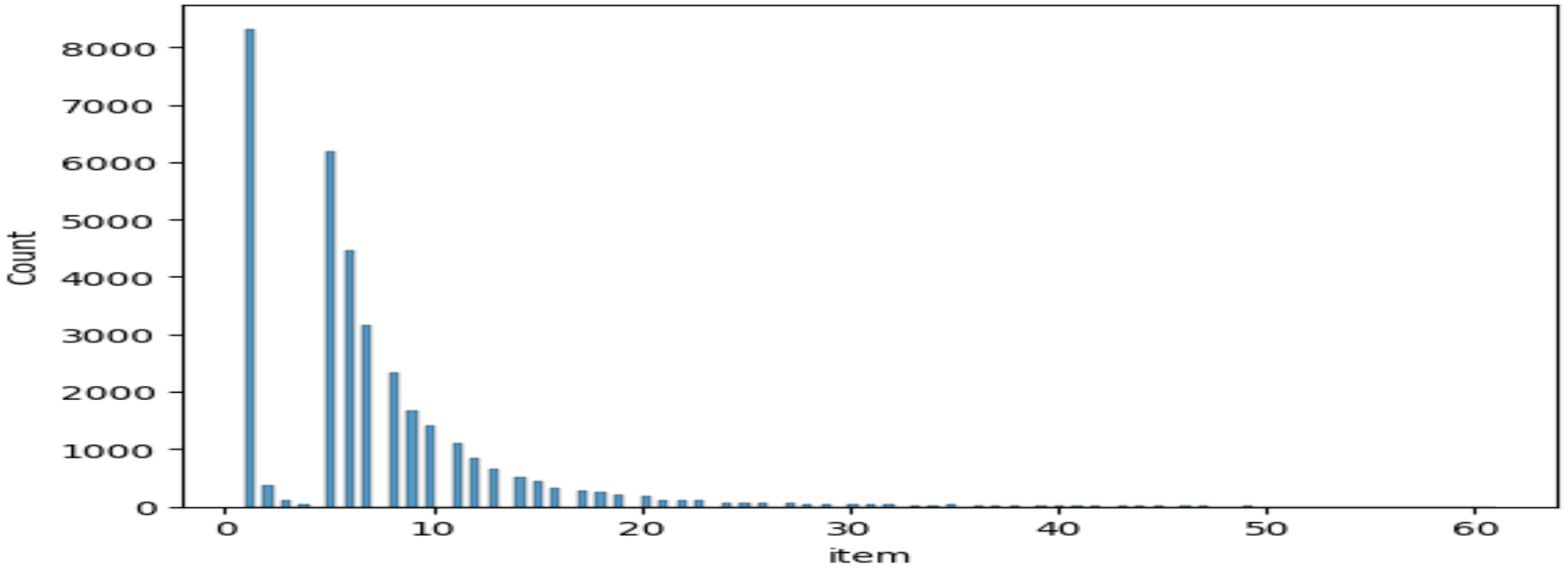
# Course counts

From our data we can see courses belong to many genre. The chart shows that BackEndDev genre has most counts of courses available that is more than 75 courses have genre of BackEndDev.



# Course enrollment distribution

-The histogram plot show the count of users that have taken (1 to 30 courses). Below plot shows that most users (more than 8000) have taken 1 course. The number of users taking multiple courses decreases as courses(items) are increased.



# 20 most popular courses

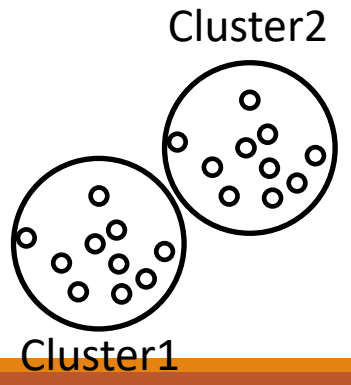
- List shows the top 20 most popular courses with python for data science and introduction to data science being most popular

| | TITLE | Ratings |
|---|---|---|
| 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 |

# Word cloud of course titles

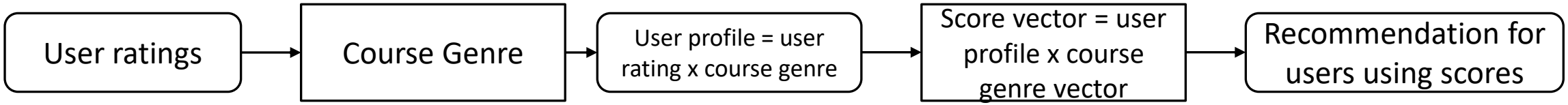
- The word cloud shows the most common words from the titles, this shows that word data, python, machine learning have been used mostly in the titles names.

# Content-based Recommender System using Unsupervised Learning



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

- We had 2 data one of courses with titles and genres, other of courses with user rating. Following steps done:
  - Make user rating vector with ratings dataframe
  - Make course genre vector
  - Multiply user rating with course genre to find user profile
  - Now multiply this user profile with genre vector of new courses to get a score vector
  - Score vector will be used to recommend courses



# Results

| | USER | COURSE_ID | SCORE |
|---|---|---|---|
| 0 | 37465 | RP0105EN | [27.0] |
| 1 | 37465 | GPXX06RFEN | [12.0] |
| 2 | 37465 | CC0271EN | [15.0] |
| 3 | 37465 | BD0145EN | [24.0] |
| 4 | 37465 | DE0205EN | [15.0] |
| ... | ... | ... | ... |
| 53406 | 2087663 | excourse88 | [15.0] |
| 53407 | 2087663 | excourse89 | [15.0] |
| 53408 | 2087663 | excourse90 | [15.0] |
| 53409 | 2087663 | excourse92 | [15.0] |
| 53410 | 2087663 | excourse93 | [15.0] |

53411 rows × 3 columns

# Evaluation results of user profile-based recommender system

## New courses recommended on average

| | userid | no_of_new_cr_recommended |
|---|---|---|
| 0 | 37465 | 67 |
| 1 | 85625 | 131 |
| 2 | 108541 | 67 |
| 3 | 109915 | 2 |
| 4 | 149690 | 72 |
| .. | ... | ... |
| 859 | 2056952 | 68 |
| 860 | 2061096 | 77 |
| 861 | 2074313 | 66 |
| 862 | 2074462 | 3 |
| 863 | 2087663 | 204 |

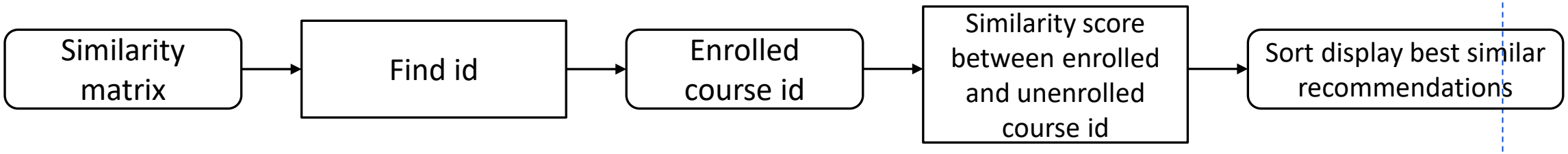
[864 rows x 2 columns]

## Top 10 common recommended courses

```
[('TA0106EN', 608),
 ('GPXX0IBEN', 548),
 ('excourse21', 547),
 ('excourse22', 547),
 ('ML0122EN', 544),
 ('GPXX0TY1EN', 533),
 ('excourse04', 533),
 ('excourse06', 533),
 ('excourse31', 524),
 ('excourse72', 516)]
```

# Flowchart of content-based recommender system using course similarity

- We will use the similarity matrix then do following steps:
  - course id to index and index to id mappings
  - Get enrolled course id
  - Find similarity of enrolled course id with unenrolled to get similar score
  - Sort score and display best recommendations for the enrolled course



# Evaluation results of course similarity based recommender system

```python
import numpy as np

# Assuming you have the DataFrame 'res_df' with 'USER' and 'COURSE_ID' columns

# Explode the nested lists in 'COURSE_ID' column
res_df = res_df.explode('COURSE_ID')

# Group by 'USER' and count the number of unique courses recommended for each user
user_course_counts = res_df.groupby('USER')['COURSE_ID'].nunique().reset_index()

# Calculate the average number of new/unseen courses recommended per user
average_new_courses = user_course_counts['COURSE_ID'].mean()

# Print the average number of new/unseen courses recommended per user
print("Average number of new/unseen courses recommended per user:", average_new_courses)
```

✓ 0.0s

```
Average number of new/unseen courses recommended per user: 1.8210332103321034
```

```python
print(top_10_courses)
```

✓ 19.0s

```
COURSE_ID
excourse62    257
excourse22    257
WA0103EN      101
TA0105         41
DS0110EN       38
excourse47     24
excourse46     24
excourse65     23
excourse63     23
TMP0101EN      17
Name: count, dtype: int64
```

# Flowchart of clustering-based recommender system

- Get data do following steps:
  - Get user profile data
  - Scale data
  - PCA
  - Do k-means clustering
  - For each cluster find best recommended courses.

# Results

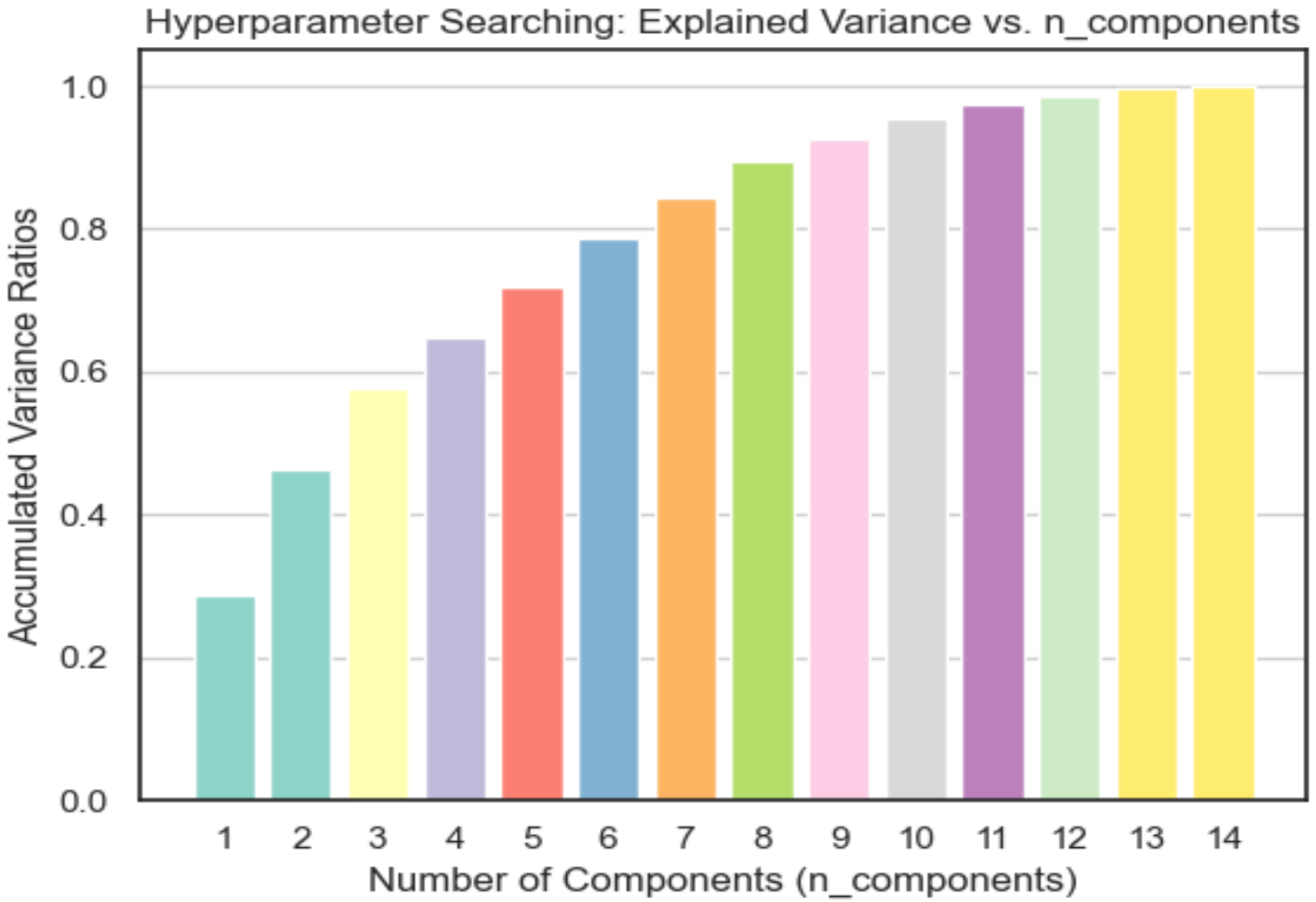
Clustering results

| | user | cluster |
|---|---|---|
| 0 | 2 | 4 |
| 1 | 4 | 4 |
| 2 | 5 | 4 |
| 3 | 7 | 8 |
| 4 | 8 | 8 |
| ... | ... | ... |
| 33896 | 2102054 | 1 |
| 33897 | 2102356 | 8 |
| 33898 | 2102680 | 3 |
| 33899 | 2102983 | 8 |
| 33900 | 2103039 | 8 |

33901 rows × 2 columns

# Evaluation results of clustering-based recommender system

PCA hyper parameters searching



# Evaluation results of clustering-based recommender system

```python
total_recommendations = 0
num_users = len(recommendations)

for user_id, courses in recommendations.items():
    total_recommendations += len(courses)

average_recommendations = total_recommendations / num_users

print("Average number of new/unseen courses recommended per user:", average_recommendations)
```

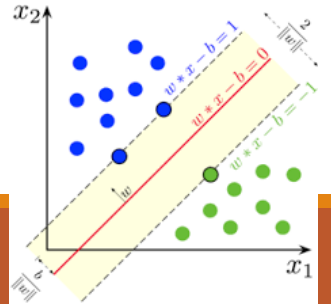
✓ 0.0s

```
Average number of new/unseen courses recommended per user: 50.0
```

✓ 0.0s

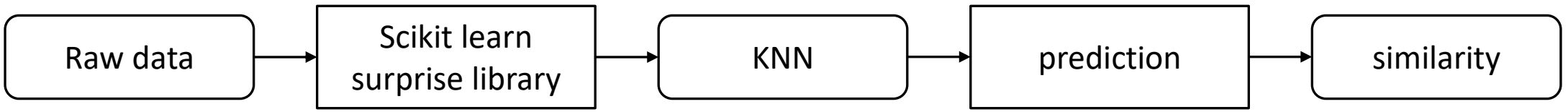
```
Top-10 commonly recommended courses:
LB0101ENv1 : 1000
RP0103 : 1000
CC0103EN : 1000
CC0101EN : 1000
CO0201EN : 1000
ML0115EN : 1000
ML0122ENv1 : 1000
ML0109EN : 1000
DE0205EN : 1000
CC0201EN : 1000
```

# Collaborative-filtering Recommender System using Supervised Learning



# Flowchart of KNN based recommender system

- Steps:
  - Data read
  - Use scikit surprise library
  - KNN algorithm, train and make prediction
  - Display similarity



# KNN
# BASIC AND WITH HYPERPARAMETER TUNNING

```
# We'll use the famous KNNBasic algorithm.
algo = KNNBasic()

# Train the algorithm on the trainset, and predict ratings for the testset
algo.fit(trainset)
predictions = algo.test(testset)

# Then compute RMSE
accuracy.rmse(predictions)
```

✓ 5.5s

```
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9871

0.9870777947206333
```

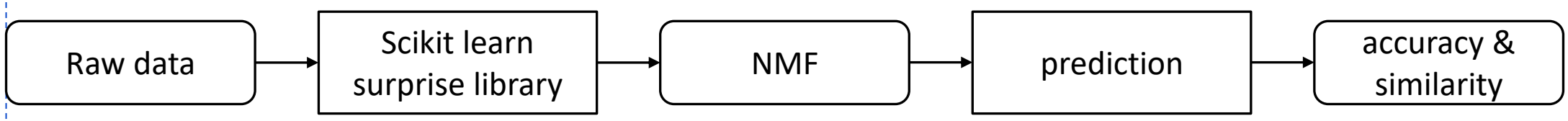
```
algo = KNNBasic()

# Train the algorithm on the trainset, and predict
algo.fit(trainset)
predictions = algo.test(testset)

# Then compute RMSE
accuracy.rmse(predictions)
```

✓ 6m 46.9s

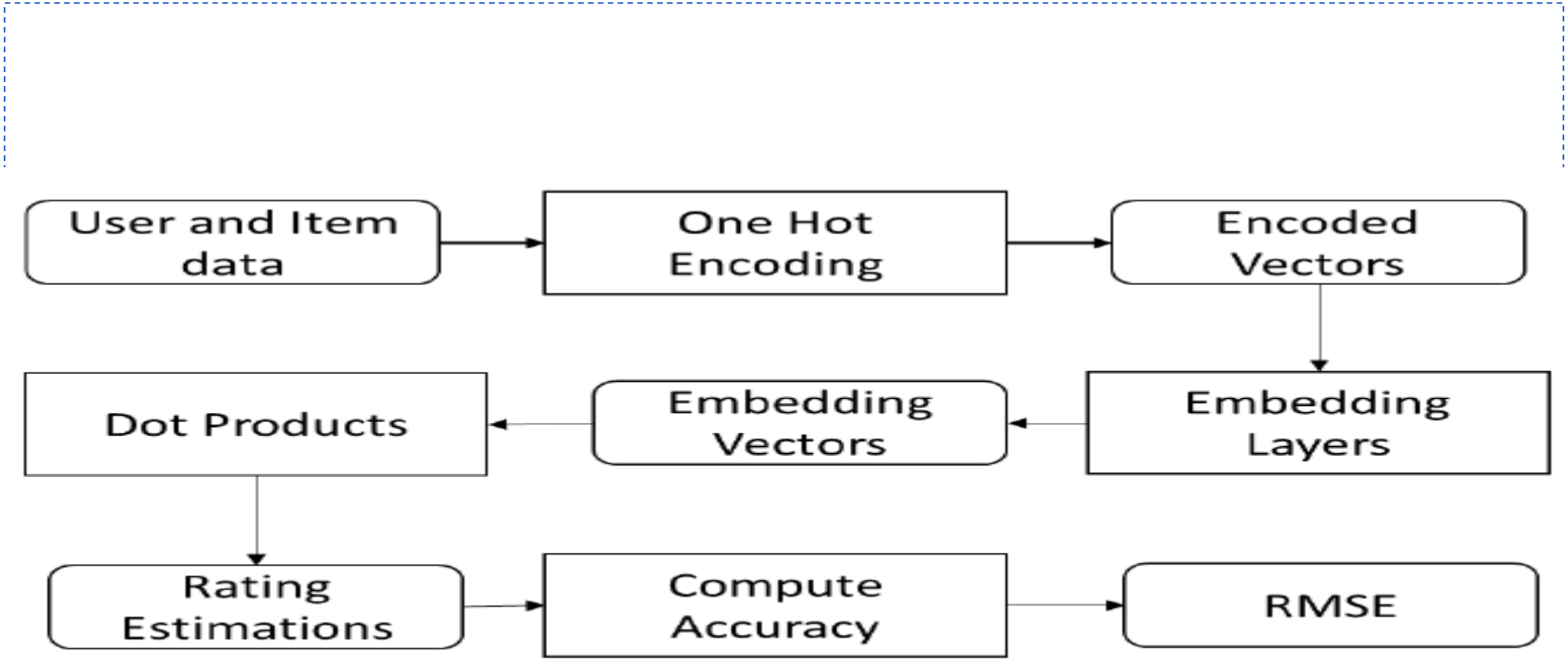
```
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.1955

0.19554180488155282
```

# Flowchart of NMF based recommender system

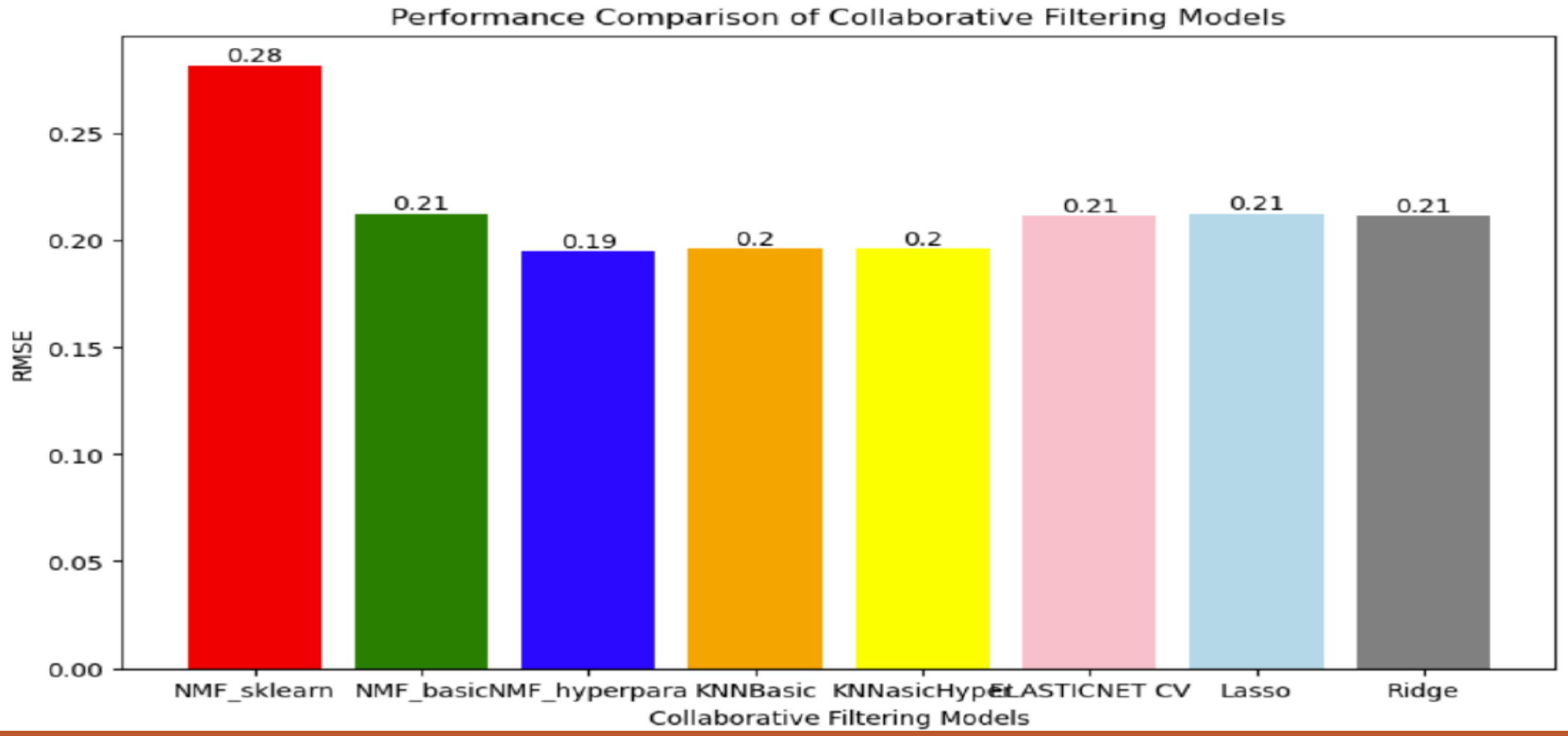
- Steps:
  - Data read
  - Use scikit surprise library
  - NMF algorithm, train and make prediction
  - Display similarity



# Flowchart of Neural Network Embedding based recommender system



# Compare the performance of collaborative-filtering models



# Conclusions

1. Enrollment distributions can provide insights into the popularity and usage patterns of online courses. By aggregating enrollment counts for each user, we can identify the total number of users and understand the distribution of enrollments among them.
2. Word cloud visualizations of course titles can help identify the most frequently occurring words, indicating popular topics or themes in the courses. This provides a quick overview of the key areas of interest covered by the courses and helps understand the general focus of the course offerings.
3. Clustering-based recommender systems group users based on their preferences or characteristics, as represented by user profile vectors. By clustering users with similar preferences, the system can generate personalized recommendations for each user based on the preferences of users in the same cluster. This approach allows for targeted recommendations and can help improve the relevance and effectiveness of the recommendation system.
4. The analysis and techniques applied in this project provide valuable insights into user enrollments, course topics, and the implementation of a clustering-based recommender system. These insights can be used to optimize course offerings, improve user engagement, and enhance the recommendation system to deliver more personalized and relevant recommendations to users.

# Appendix

Github repo: https://github.com/aminghias/ML-Capstone-Coursera-IBM-

Codes : https://github.com/aminghias/ML-Capstone-Coursera-IBM-

Notebooks: https://github.com/aminghias/ML-Capstone-Coursera-IBM-

Thankyou