Final Report -Recommendation system

Machine Learning Capstone - IBM Professional Certificate
By / Adham Ashraf Salah
Date: 16/9/2024

Executive Summary

The main goal of this project is to improve learners' learning experience via helping them quickly find new interested courses and better paving their learning paths. Meanwhile, with more learners interacting with more courses via your recommender systems, your company's revenue may also be increased.

This project is currently at the Proof of Concept (PoC) phase so the main focus at this moment is to explore and compare various machine learning models and find one with the best performance in off-line evaluations.

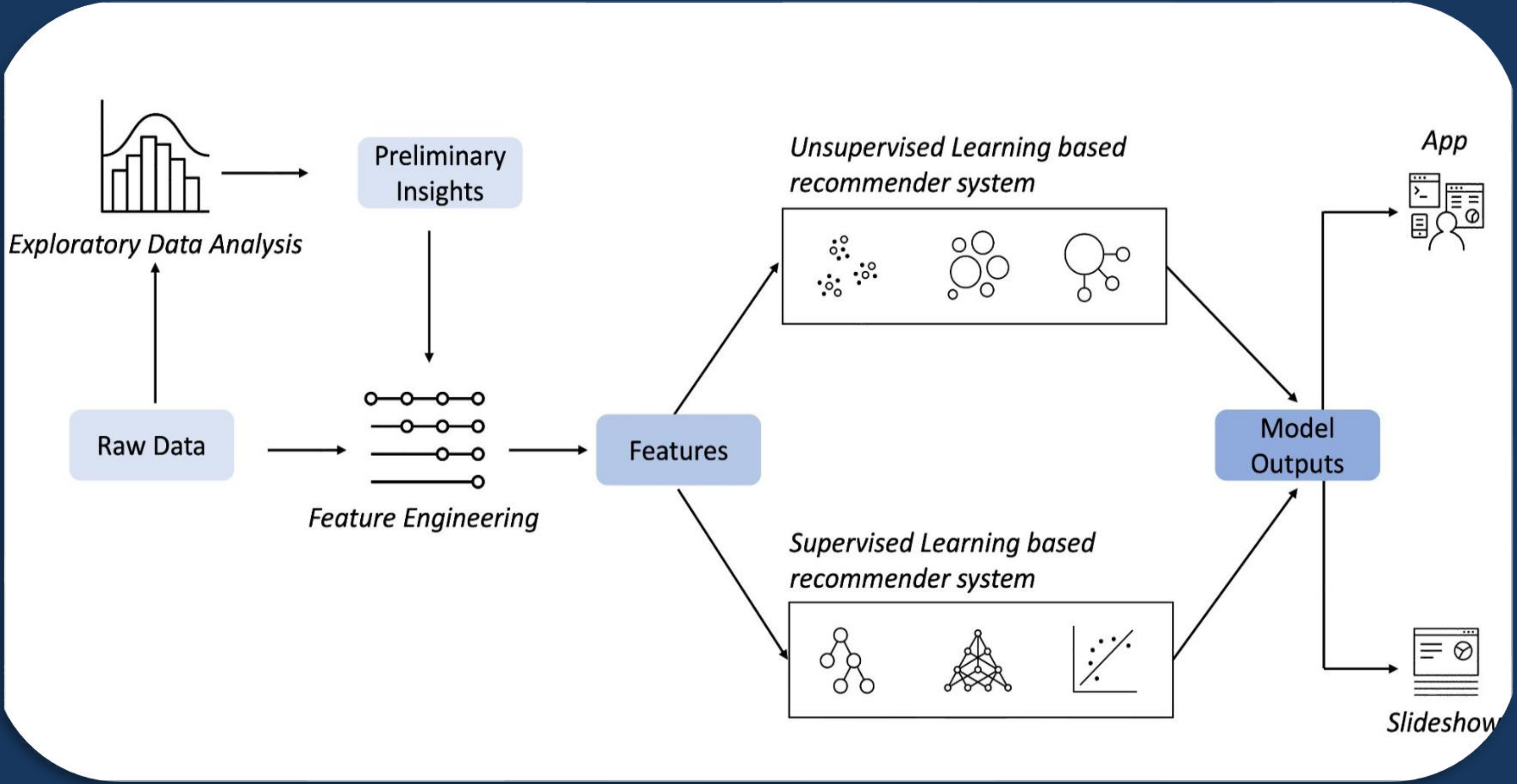
Table Of Contents (TOC):

1. Introduction (Capstone Overview)
2. Exploratory Data Analysis and Feature Engineering
3. Content-based Recommendation System using unsupervised Learning
4. Collaborative filtering-based Recommendation System using supervised learning
5. Conclusion
6. Future Improvements

Machine learning pipeline

1. Collecting and understanding data
2. Performing exploratory data analysis on online course enrollments datasets
3. Extracting Bag of Words (BoW) features from course textual content
4. Calculating course similarity using BoW features
5. Building content-based recommender systems using various unsupervised learning algorithms, such as: Distance/Similarity measurements, K-means, Principal Component Analysis (PCA), etc.
6. Building collaborative-filtering recommender systems using various supervised learning algorithms
   K Nearest Neighbors, Non-negative Matrix Factorization (NMF), Neural Networks, Linear Regression, Logistic Regression, RandomForest, etc.
7. Deploying and demonstrate modeling via a web app built with `streamlit`. `Streamlit` is an open-source app framework for Machine Learning and Data Science to quickly demonstration.
8. Reporting in paper.

Recommender Systems Machine learning pipeline


Exploratory Data Analysis
Preliminary Insights
Raw Data
Feature Engineering
Features
Unsupervised Learning based recommender system
Supervised Learning based recommender system
Model Outputs
App
Slideshow

Exploratory Data Analysis and Feature Engineering

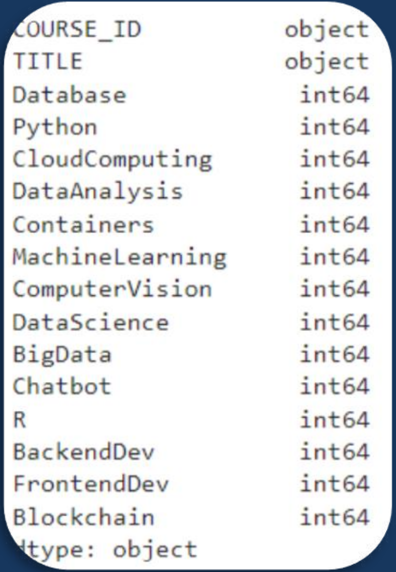
In this exploratory data analysis, the utilization of WordCloud visualization unveils prevalent keywords within a dataset of online course titles. By aggregating and filtering course titles, dominant themes such as Python, data science, machine learning, big data, AI, TensorFlow, containers, and cloud computing are identified. This analysis provides valuable insights into the trending IT skills and subject areas covered by the online courses, guiding learners and educators in understanding the current landscape of digital learning opportunities.

Exploratory Data Analysis

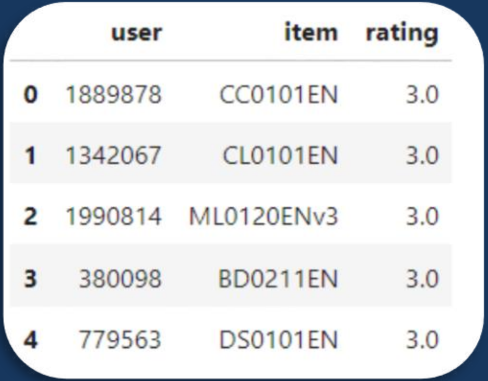
Describe statistics of data sets

There are two data sets one for courses and one for ratings

There are 307 courses total



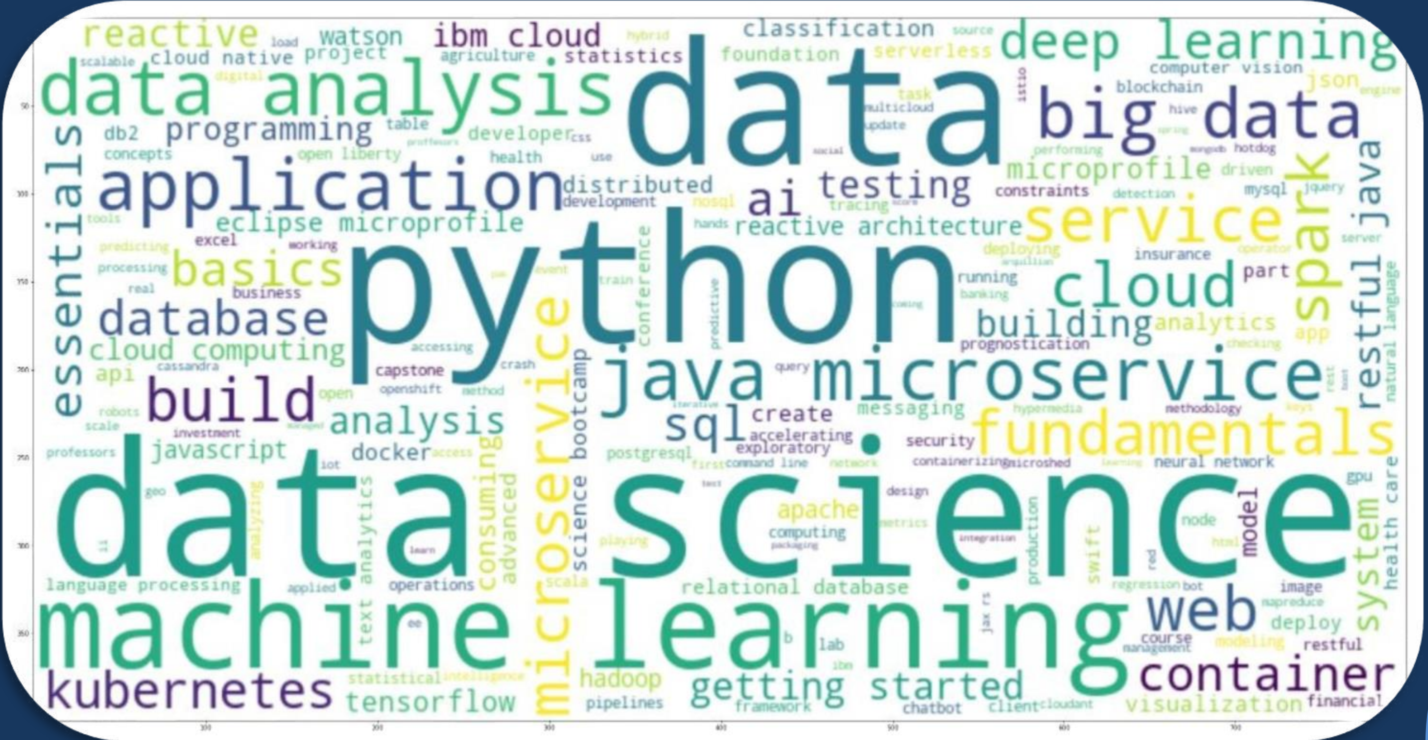
```
COURSE_ID           object
TITLE               object
Database             int64
Python               int64
CloudComputing       int64
DataAnalysis         int64
Containers           int64
MachineLearning      int64
ComputerVision       int64
DataScience          int64
BigData              int64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



	user	item	rating
0	1889878	CC0101EN	3.0
1	1342067	CL0101EN	3.0
2	1990814	ML0120ENv3	3.0
3	380098	BD0211EN	3.0
4	779563	DS0101EN	3.0

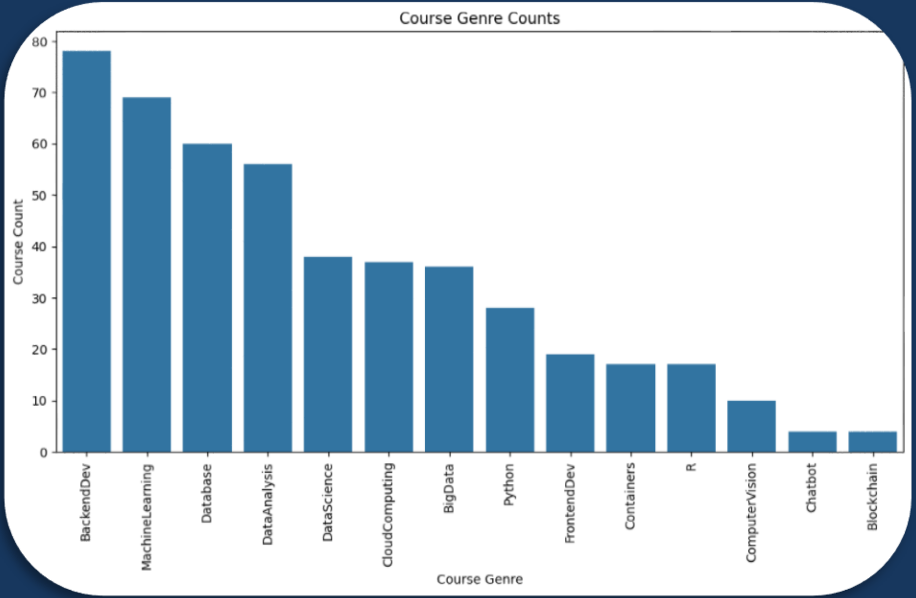
Plot a Word Cloud from Course Titles

As we can see from the word cloud ,there are many popular IT-related keywords such as python, data science, machine learning, big data ,AI , tensorflow, container, cloud, etc .



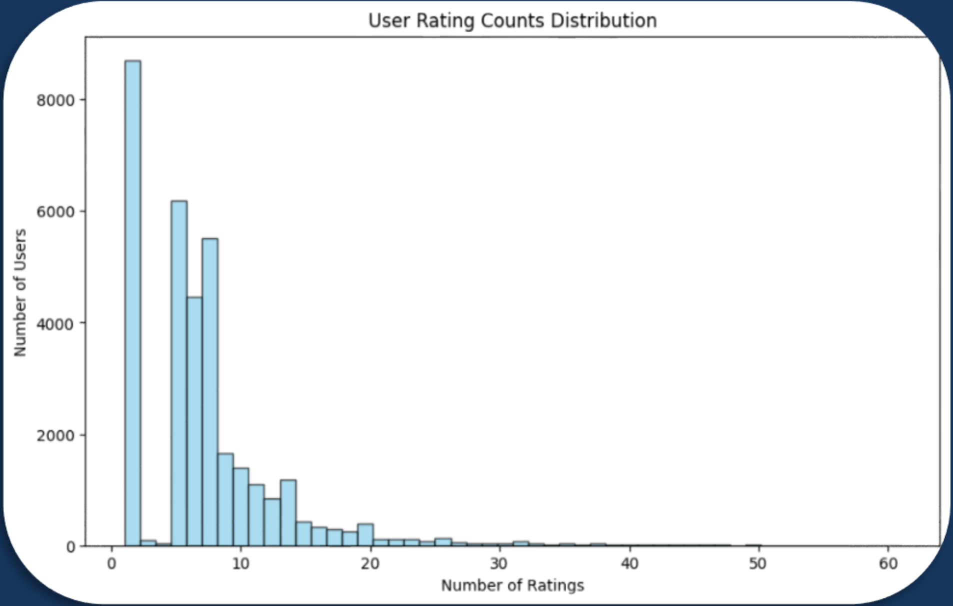
Determine popular course genres

The analysis revealed that Backend Development, Machine Learning, and Database courses are among the most prevalent, while Blockchain and Chatbot courses are less common. This exploration provides valuable insights for learners and educators seeking to understand current trends in online education.



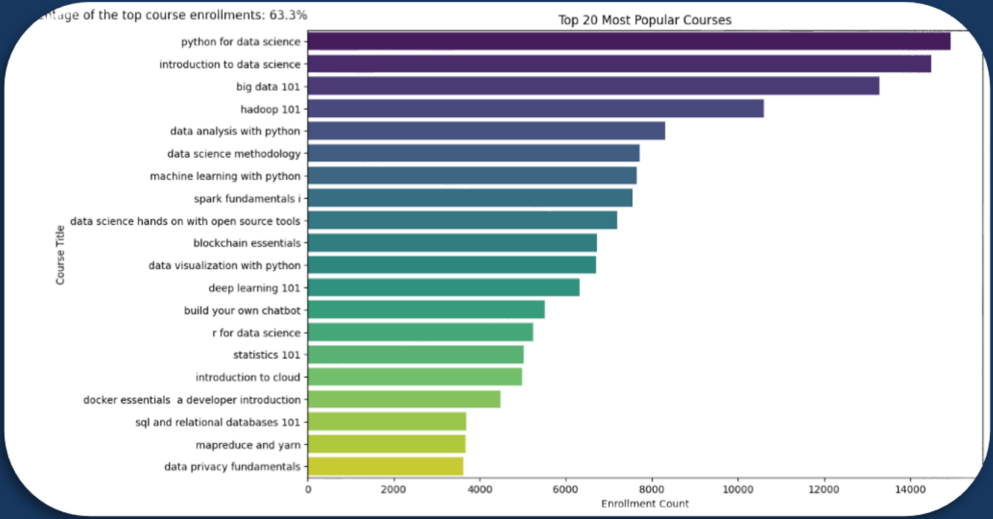
Understanding User engagement and interaction

By aggregating the rating counts for each user, it was found that the dataset comprises 233,306 enrollment records from 5,000 unique users. The histogram distribution of user rating counts illustrates varying levels of engagement, with the majority of users giving relatively few ratings while a smaller proportion giving a larger number of ratings.



Identifying the 20 most popular courses

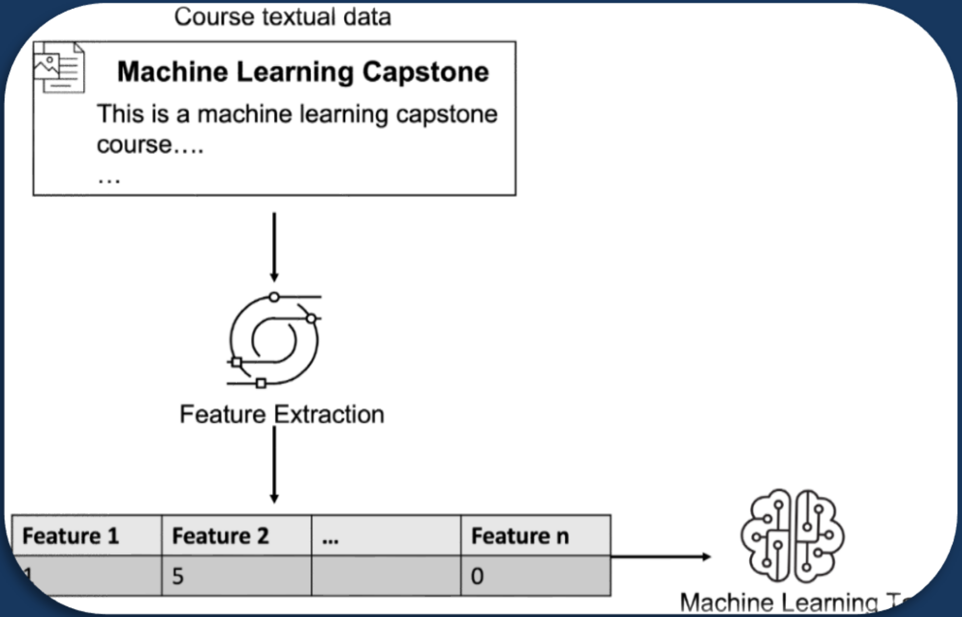
In this task, the objective was to determine the top 20 most popular courses based on enrolment counts. Using the enrolment data, the courses were aggregated, and sorted, and the top 20 courses were identified. The resulting list showcases the titles and enrollment counts of these highly sought-after courses, providing insights into the preferences of learners in the online learning platform.



Feature Engineering

Machine learning algorithms cannot work on an item directly so we first need to extract features and represent the items mathematically, i.e., with a feature vector.

Many items are often described by text so they are associated with textual data, such as the titles and descriptions of a movie or course. Since machine learning algorithms can not process textual data directly, we need to transform the raw text into numeric feature vectors.



Machine Learning Capstone

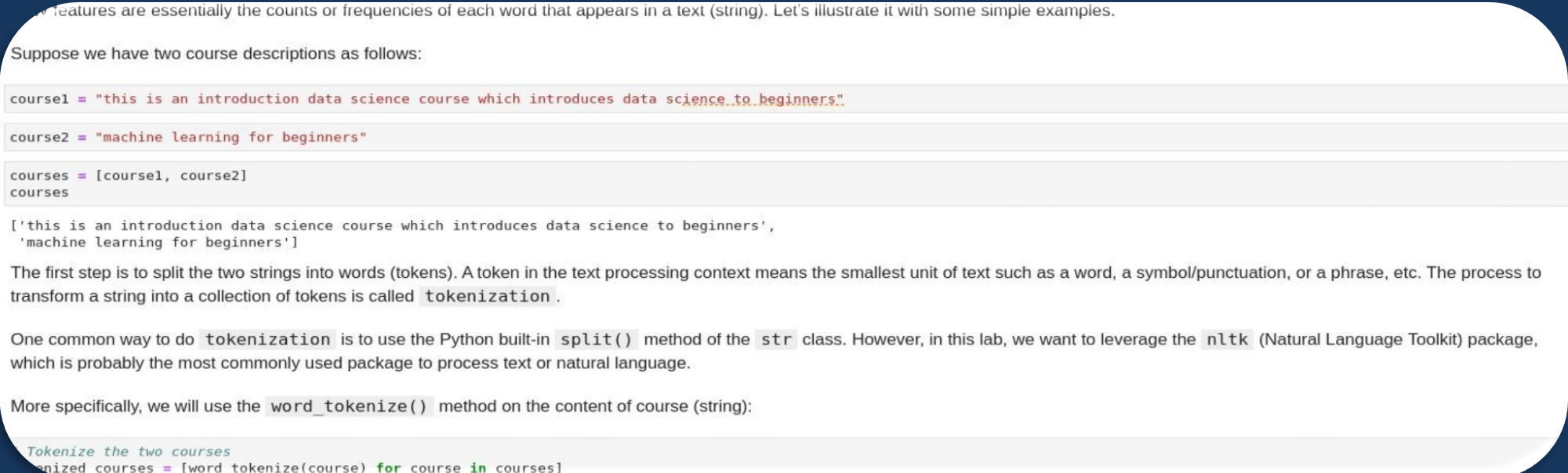
This is a machine learning capstone course….

…

Feature 1	Feature 2	...	Feature n
1	5		0

Feature Engineering -BoW

- A document may contain tens of thousands of words which makes the dimension of the BoW feature vector huge.
- To reduce the dimensionality, one common way is to filter the relatively meaningless tokens such as stop words or sometimes add position and adjective words.



features are essentially the counts or frequencies of each word that appears in a text (string). Let's illustrate it with some simple examples.

Suppose we have two course descriptions as follows:

```
course1 = "this is an introduction data science course which introduces data science to beginners"
```

```
course2 = "machine learning for beginners"
```

```
courses = [course1, course2]
courses
```

```
['this is an introduction data science course which introduces data science to beginners',
 'machine learning for beginners']
```

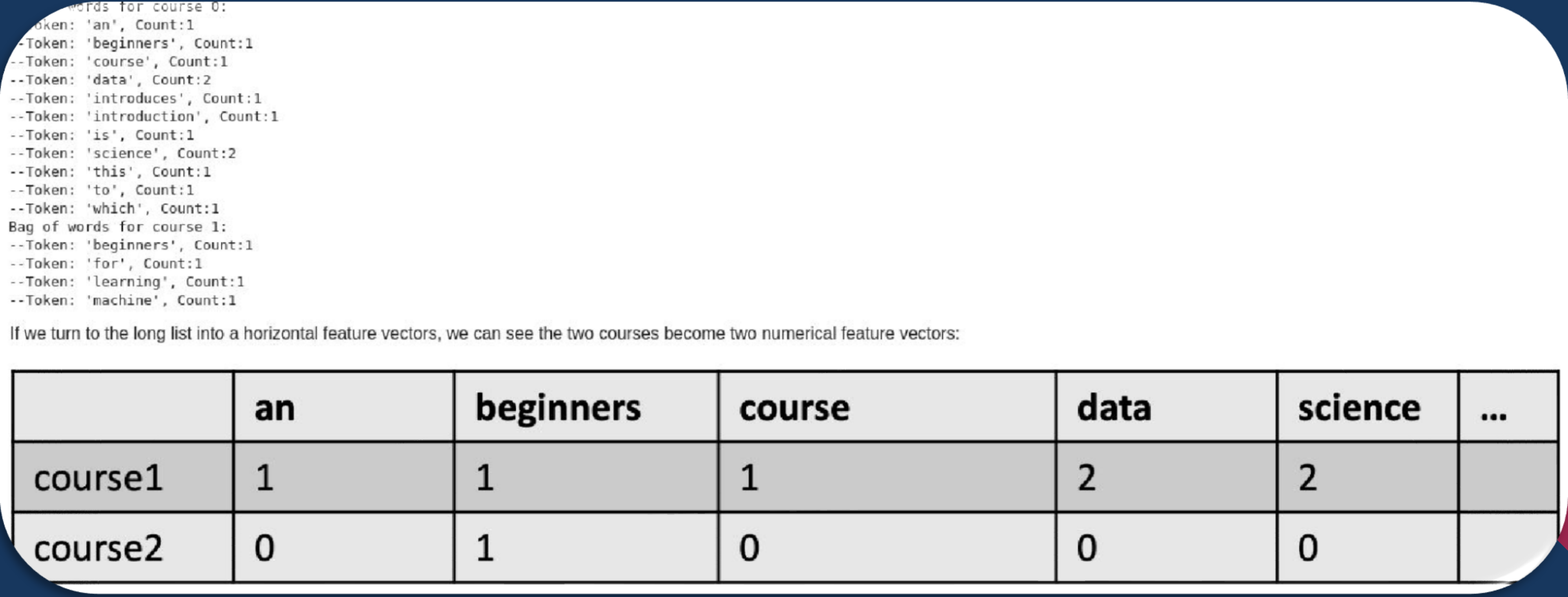
The first step is to split the two strings into words (tokens). A token in the text processing context means the smallest unit of text such as a word, a symbol/punctuation, or a phrase, etc. The process to transform a string into a collection of tokens is called `tokenization`.

One common way to do `tokenization` is to use the Python built-in `split()` method of the `str` class. However, in this lab, we want to leverage the `nltk` (Natural Language Toolkit) package, which is probably the most commonly used package to process text or natural language.

More specifically, we will use the `word_tokenize()` method on the content of course (string):

```
 # Tokenize the two courses
enized_courses = [word_tokenize(course) for course in courses]
```

Feature Engineering -BoW



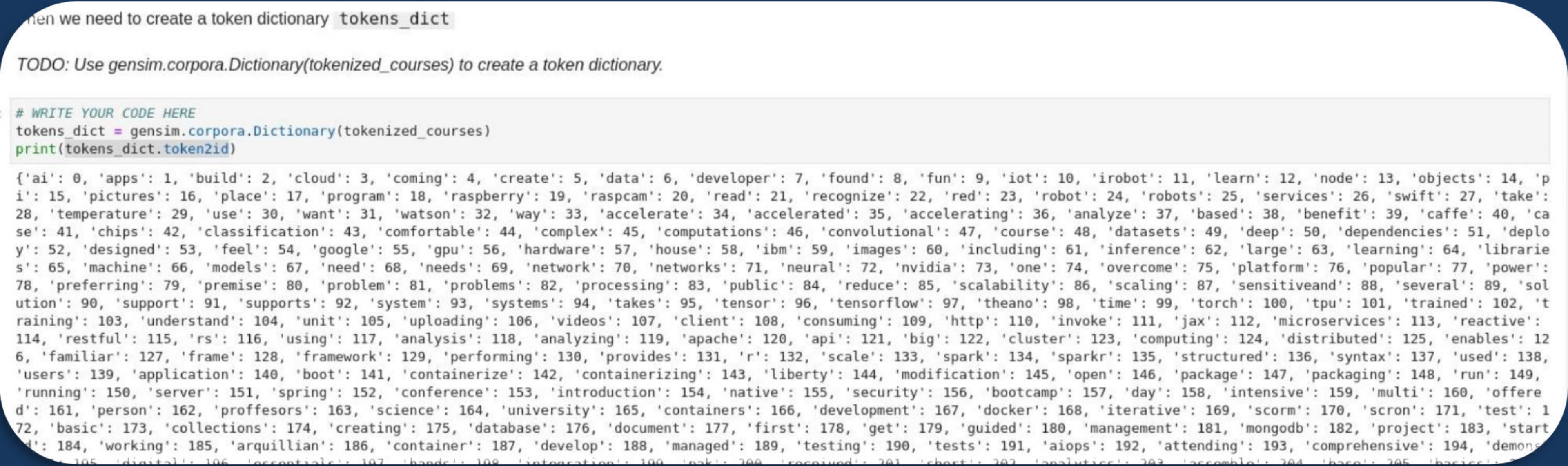
```
words for course 0:
oken: 'an', Count:1
-Token: 'beginners', Count:1
--Token: 'course', Count:1
--Token: 'data', Count:2
--Token: 'introduces', Count:1
--Token: 'introduction', Count:1
--Token: 'is', Count:1
--Token: 'science', Count:2
--Token: 'this', Count:1
--Token: 'to', Count:1
--Token: 'which', Count:1
Bag of words for course 1:
--Token: 'beginners', Count:1
--Token: 'for', Count:1
--Token: 'learning', Count:1
--Token: 'machine', Count:1
```

If we turn to the long list into a horizontal feature vectors, we can see the two courses become two numerical feature vectors:

	an	beginners	course	data	science	...
course1	1	1	1	2	2	
course2	0	1	0	0	0	

Feature Engineering -BoW

Extract BoW features for course textual content and build a dataset



hen we need to create a token dictionary `tokens_dict`

TODO: Use gensim.corpora.Dictionary(tokenized_courses) to create a token dictionary.

```python
# WRITE YOUR CODE HERE
tokens_dict = gensim.corpora.Dictionary(tokenized_courses)
print(tokens_dict.token2id)
```

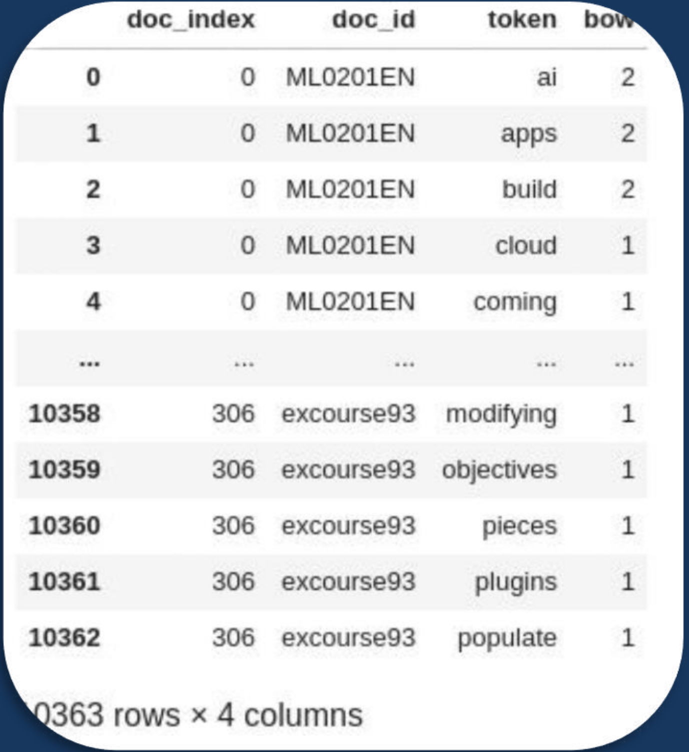
```
{'ai': 0, 'apps': 1, 'build': 2, 'cloud': 3, 'coming': 4, 'create': 5, 'data': 6, 'developer': 7, 'found': 8, 'fun': 9, 'iot': 10, 'irobot': 11, 'learn': 12, 'node': 13, 'objects': 14, 'p
i': 15, 'pictures': 16, 'place': 17, 'program': 18, 'raspberry': 19, 'raspcam': 20, 'read': 21, 'recognize': 22, 'red': 23, 'robot': 24, 'robots': 25, 'services': 26, 'swift': 27, 'take':
28, 'temperature': 29, 'use': 30, 'want': 31, 'watson': 32, 'way': 33, 'accelerate': 34, 'accelerated': 35, 'accelerating': 36, 'analyze': 37, 'based': 38, 'benefit': 39, 'caffe': 40, 'ca
se': 41, 'chips': 42, 'classification': 43, 'comfortable': 44, 'complex': 45, 'computations': 46, 'convolutional': 47, 'course': 48, 'datasets': 49, 'deep': 50, 'dependencies': 51, 'deplo
y': 52, 'designed': 53, 'feel': 54, 'google': 55, 'gpu': 56, 'hardware': 57, 'house': 58, 'ibm': 59, 'images': 60, 'including': 61, 'inference': 62, 'large': 63, 'learning': 64, 'librarie
s': 65, 'machine': 66, 'models': 67, 'need': 68, 'needs': 69, 'network': 70, 'networks': 71, 'neural': 72, 'nvidia': 73, 'one': 74, 'overcome': 75, 'platform': 76, 'popular': 77, 'power':
78, 'preferring': 79, 'premise': 80, 'problem': 81, 'problems': 82, 'processing': 83, 'public': 84, 'reduce': 85, 'scalability': 86, 'scaling': 87, 'sensitiveand': 88, 'several': 89, 'sol
ution': 90, 'support': 91, 'supports': 92, 'system': 93, 'systems': 94, 'takes': 95, 'tensor': 96, 'tensorflow': 97, 'theano': 98, 'time': 99, 'torch': 100, 'tpu': 101, 'trained': 102, 't
raining': 103, 'understand': 104, 'unit': 105, 'uploading': 106, 'videos': 107, 'client': 108, 'consuming': 109, 'http': 110, 'invoke': 111, 'jax': 112, 'microservices': 113, 'reactive':
114, 'restful': 115, 'rs': 116, 'using': 117, 'analysis': 118, 'analyzing': 119, 'apache': 120, 'api': 121, 'big': 122, 'cluster': 123, 'computing': 124, 'distributed': 125, 'enables': 12
6, 'familiar': 127, 'frame': 128, 'framework': 129, 'performing': 130, 'provides': 131, 'r': 132, 'scale': 133, 'spark': 134, 'sparkr': 135, 'structured': 136, 'syntax': 137, 'used': 138,
'users': 139, 'application': 140, 'boot': 141, 'containerize': 142, 'containerizing': 143, 'liberty': 144, 'modification': 145, 'open': 146, 'package': 147, 'packaging': 148, 'run': 149,
'running': 150, 'server': 151, 'spring': 152, 'conference': 153, 'introduction': 154, 'native': 155, 'security': 156, 'bootcamp': 157, 'day': 158, 'intensive': 159, 'multi': 160, 'offere
d': 161, 'person': 162, 'proffesors': 163, 'science': 164, 'university': 165, 'containers': 166, 'development': 167, 'docker': 168, 'iterative': 169, 'scorm': 170, 'scron': 171, 'test': 1
72, 'basic': 173, 'collections': 174, 'creating': 175, 'database': 176, 'document': 177, 'first': 178, 'get': 179, 'guided': 180, 'management': 181, 'mongodb': 182, 'project': 183, 'start
d': 184, 'working': 185, 'arquillian': 186, 'container': 187, 'develop': 188, 'managed': 189, 'testing': 190, 'tests': 191, 'aiops': 192, 'attending': 193, 'comprehensive': 194, 'demons
 195, 'digital': 196, 'essentials': 197, 'hands': 198, 'integration': 199, 'pak': 200, 'received': 201, 'short': 202, 'analytics': 203, 'assemble': 204, 'base': 205, 'basics'
```

Feature Engineering -BoW

Extract BoW features for course textual content and build a dataset

Create a new course_bow dataframe based on the extracted BoW features. he new dataframe needs to include the following columns (you may include other relevant columns as well):

- 'doc_index': the course index starting from 0
- 'doc_id': the actual course id such as ML0201EN
- 'token': the tokens for each course
- 'bow': the bow value for each token



	doc_index	doc_id	token	bow
0	0	ML0201EN	ai	2
1	0	ML0201EN	apps	2
2	0	ML0201EN	build	2
3	0	ML0201EN	cloud	1
4	0	ML0201EN	coming	1
...	...	...	...	...
10358	306	excourse93	modifying	1
10359	306	excourse93	objectives	1
10360	306	excourse93	pieces	1
10361	306	excourse93	plugins	1
10362	306	excourse93	populate	1

0363 rows × 4 columns

Content-based Recommendation System Using Unsupervised Learning

1. Content-based Course Recommender System using User Profile and Course Genres

Raw data EDA | Feature engineering | Get recommend score | Recomendation

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

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

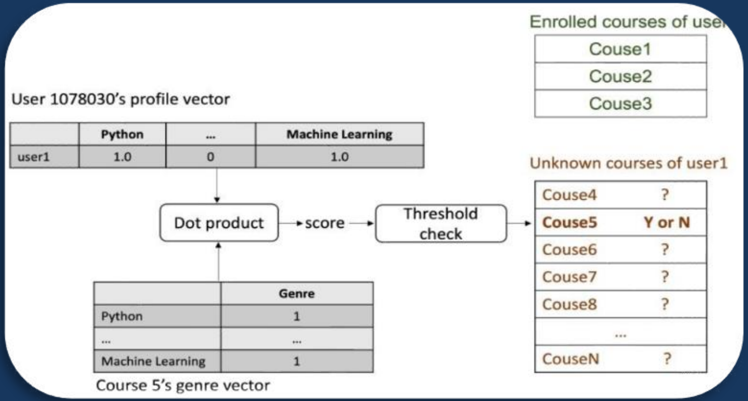
Analysis data

Drop the exceptions

Normalise data

Use dot product between single user vector and specific course to get recommend score

Make prediction by comparing score with determine threshold



	Python	...	Machine Learning
user1	1.0	0	1.0

	Genre
Python	1
...	...
Machine Learning	1

1. Content-based Course Recommender System using User Profile and Course Genres

- Hyper-parameter Settings:

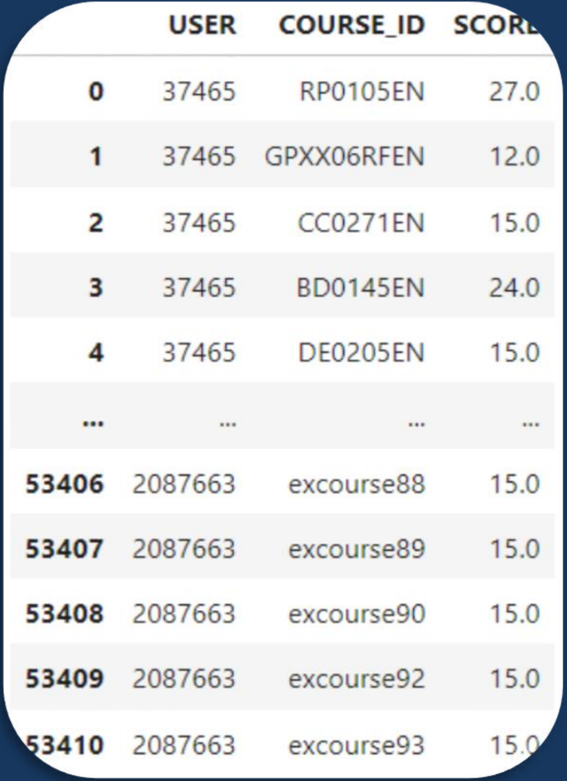
In our discussion, we set a recommendation score threshold of 10.0 to filter out low-scoring recommendations. This threshold determines which courses are considered relevant enough to be recommended to users. Additionally, we may have adjusted other hyperparameters such as feature representation methods or similarity metrics during the implementation of the recommender system.

- Average Number of New Courses Recommended per User:

We calculated the average number of new courses recommended per user in the test user dataset. This metric helps evaluate the coverage and diversity of the recommender system. In our case, the average number was approximately **61.82** courses per user.

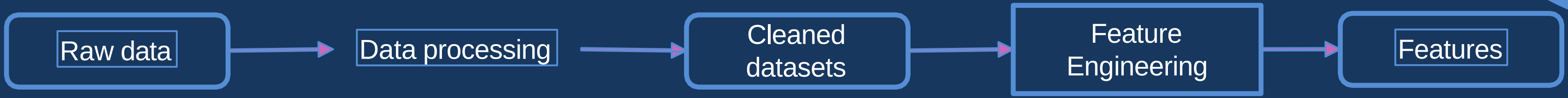
- Top-10 Most Frequently Recommended Courses:

The table represents the top 10 most frequently recommended courses based on the user profile- based recommender system. Each row corresponds to a course, identified by its COURSE_ID, and the number of times that course has been recommended to users, denoted by the RECOMMENDATION_COUNT column. These recommendations are generated by analyzing user profiles and course genre vectors, with courses scoring higher in relevance to a user's interests being recommended more frequently.



	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

2. Content-based Course Recommender System using Course Similarities

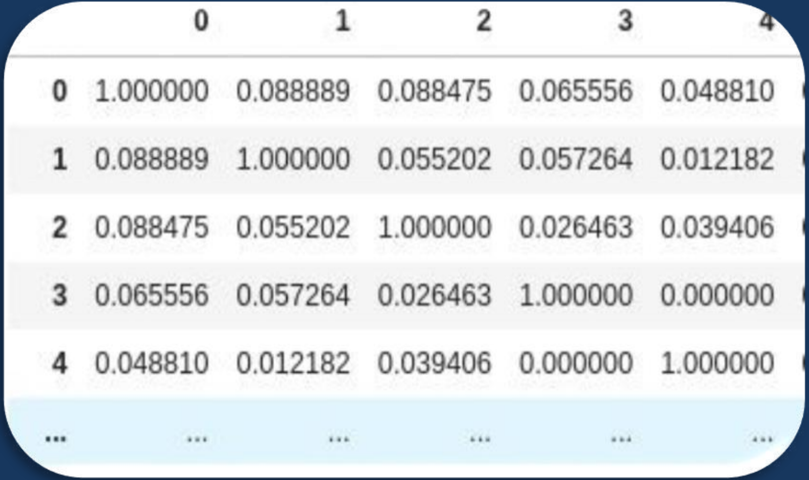


Let's break down the flowchart based on the conversation we had:

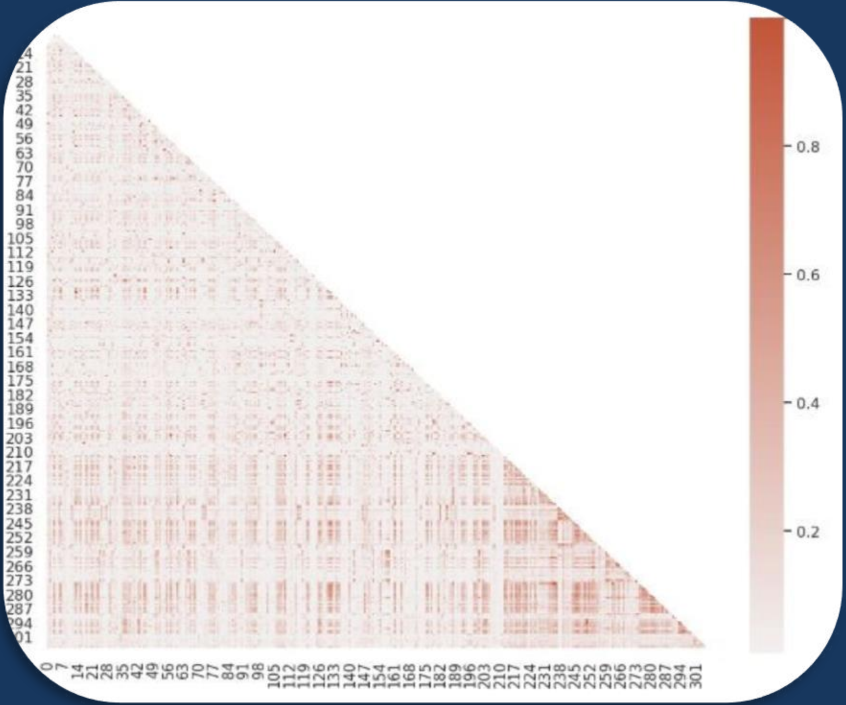
1. **Raw Data**: Refers to the initial dataset containing information about various courses, including their titles, descriptions, and other relevant attributes.
2. **Data Processing**: Involves preprocessing the raw data, which includes tokenization and lemmatization to break down the text into individual words and convert them to their base forms..
3. **Cleaned Dataset**: After processing, the dataset undergoes cleaning, which includes removing stopwords (commonly used words with little semantic value) and outliers (irrelevant or noisy data points).
4. **Feature Engineering**: This step involves transforming the cleaned dataset into numerical features that represent the courses. In this case, TF-IDF (Term Frequency-Inverse Document Frequency) vectors are calculated for each course based on the words they contain and their importance in the dataset.
5. **Features**: Refers to the final set of features used to represent each course, which are the TF-IDF vectors obtained from the feature engineering step. These features are then used to calculate similarities between courses and generate recommendations based on content similarity.

2. Content-based Course Recommender System using Course Similarities

Course similarity matrix:

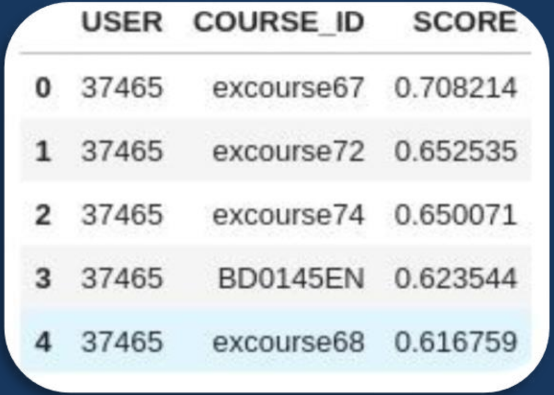


	0	1	2	3	4
0	1.000000	0.088889	0.088475	0.065556	0.048810
1	0.088889	1.000000	0.055202	0.057264	0.012182
2	0.088475	0.055202	1.000000	0.026463	0.039406
3	0.065556	0.057264	0.026463	1.000000	0.000000
4	0.048810	0.012182	0.039406	0.000000	1.000000
...	...	...	...	...	...



2. Content-based Course Recommender System using Course Similarities

Generate course recommendations based on course similarities for all test users
Top-5 Most Frequently Recommended Courses



	USER	COURSE_ID	SCORE
0	37465	excourse67	0.708214
1	37465	excourse72	0.652535
2	37465	excourse74	0.650071
3	37465	BD0145EN	0.623544
4	37465	excourse68	0.616759

3. Clustering based Course Recommender System

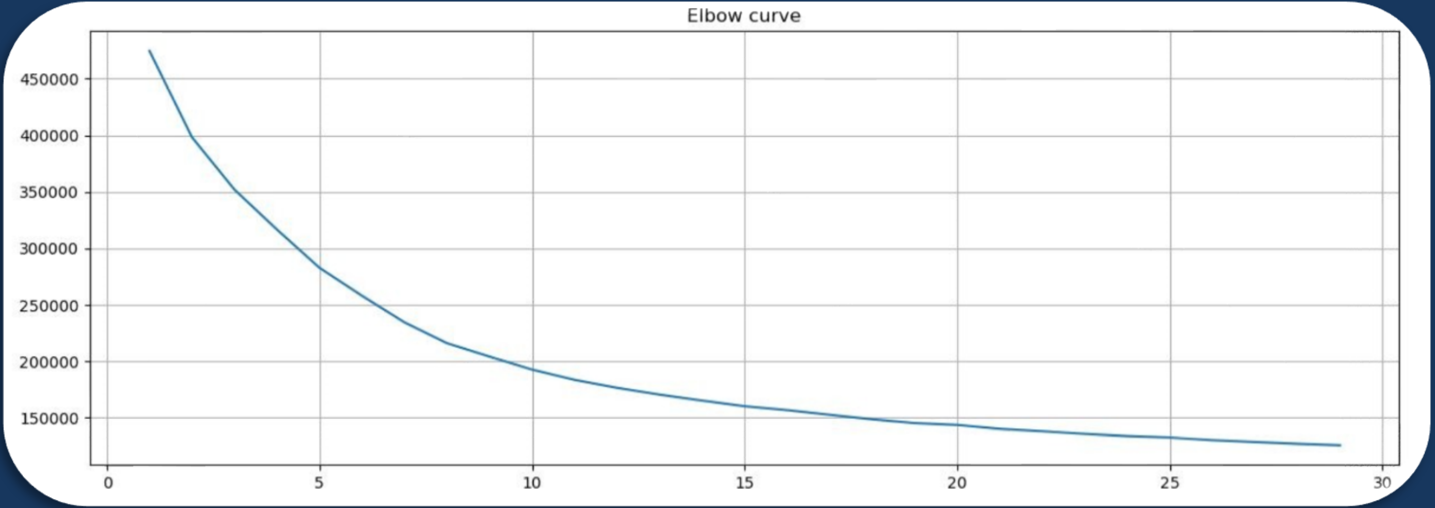
Raw data → Features → Clusters → Prediction

1. Raw data:
    - User profile dataframe: user_id, [genre_x, genre_y,...]
2. Features:
    - Normalise user profile features
    - Apply PCA to keep only important features
3. Apply Clustering algorithms to group similar courses
4. Make recommendation by taking courses in user's interest group

3. Clustering based Course Recommender System

For KMeans algorithm, one important hyperparameter is the number of clusters *n_cluster*, and a good way to find the optimized *n_cluster* is using to grid search a list of candidates and find the one with the best or optimized clustering evaluation metrics such as minimal sum of squared distance.

Grid search the optimized n_cluster for KMeans() model

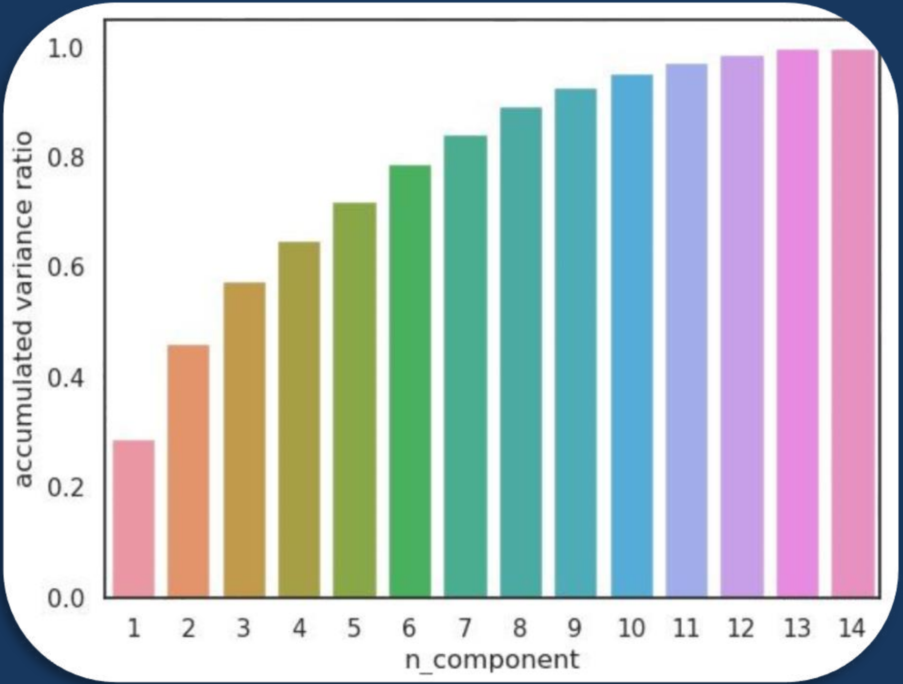


3. Clustering based Course Recommender System

Apply the PCA() provided by scikit-learn to find the main components in user profile feature vectors and see if we can reduce its dimensions by only keeping the main components.

If the accumulated variances ratio of a candidate n_components is larger than a threshold, e.g., 90%, then we can say the transformed n_components could explain about 90% of variances of the original data variance and can be considered as an optimized components size.

We select n_component = 8, due to the minimal ratio > 0.9



3. Clustering based Course Recommender System

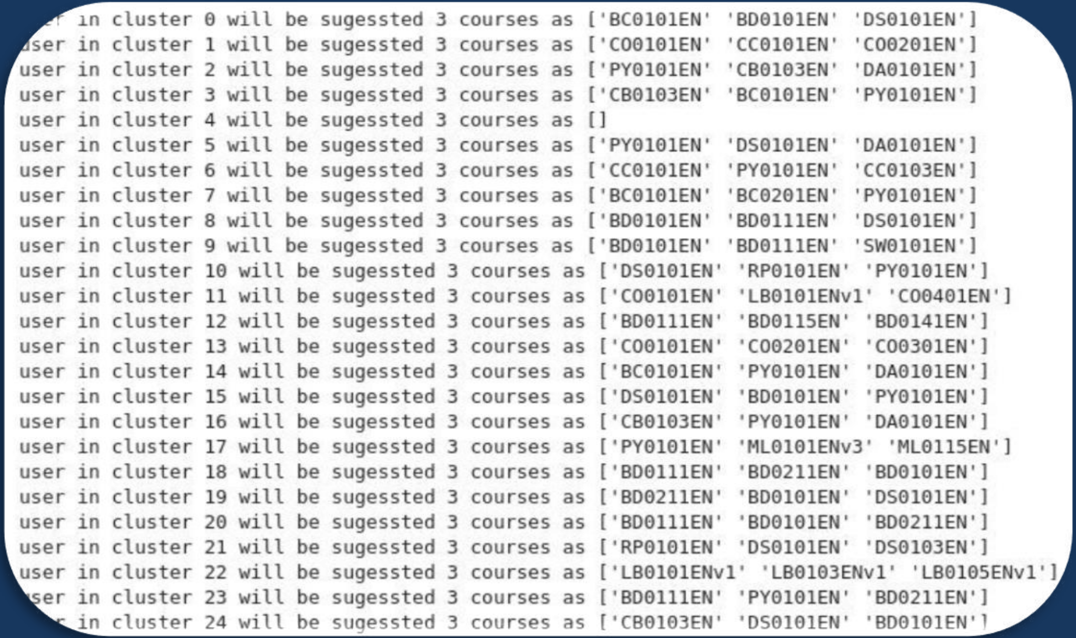
After applying PCA to features:

Find popular courses in clusters and suggest to user in cluster:

Insights:

- On average, how many new/unseen courses have been recommended per user (in the test user dataset)
- What are the most frequently recommended courses? Return the top-10 commonly recommended courses

Our analysis of the most frequently recommended courses revealed compelling insights into the preferences of users within each cluster.



```
r in cluster 0 will be sugessted 3 courses as ['BC0101EN' 'BD0101EN' 'DS0101EN']
user in cluster 1 will be sugessted 3 courses as ['CO0101EN' 'CC0101EN' 'CO0201EN']
user in cluster 2 will be sugessted 3 courses as ['PY0101EN' 'CB0103EN' 'DA0101EN']
user in cluster 3 will be sugessted 3 courses as ['CB0103EN' 'BC0101EN' 'PY0101EN']
user in cluster 4 will be sugessted 3 courses as []
user in cluster 5 will be sugessted 3 courses as ['PY0101EN' 'DS0101EN' 'DA0101EN']
user in cluster 6 will be sugessted 3 courses as ['CC0101EN' 'PY0101EN' 'CC0103EN']
user in cluster 7 will be sugessted 3 courses as ['BC0101EN' 'BC0201EN' 'PY0101EN']
user in cluster 8 will be sugessted 3 courses as ['BD0101EN' 'BD0111EN' 'DS0101EN']
user in cluster 9 will be sugessted 3 courses as ['BD0101EN' 'BD0111EN' 'SW0101EN']
user in cluster 10 will be sugessted 3 courses as ['DS0101EN' 'RP0101EN' 'PY0101EN']
user in cluster 11 will be sugessted 3 courses as ['CO0101EN' 'LB0101ENv1' 'CO0401EN']
user in cluster 12 will be sugessted 3 courses as ['BD0111EN' 'BD0115EN' 'BD0141EN']
user in cluster 13 will be sugessted 3 courses as ['CO0101EN' 'CO0201EN' 'CO0301EN']
user in cluster 14 will be sugessted 3 courses as ['BC0101EN' 'PY0101EN' 'DA0101EN']
user in cluster 15 will be sugessted 3 courses as ['DS0101EN' 'BD0101EN' 'PY0101EN']
user in cluster 16 will be sugessted 3 courses as ['CB0103EN' 'PY0101EN' 'DA0101EN']
user in cluster 17 will be sugessted 3 courses as ['PY0101EN' 'ML0101ENv3' 'ML0115EN']
user in cluster 18 will be sugessted 3 courses as ['BD0111EN' 'BD0211EN' 'BD0101EN']
user in cluster 19 will be sugessted 3 courses as ['BD0211EN' 'BD0101EN' 'DS0101EN']
user in cluster 20 will be sugessted 3 courses as ['BD0111EN' 'BD0101EN' 'BD0211EN']
user in cluster 21 will be sugessted 3 courses as ['RP0101EN' 'DS0101EN' 'DS0103EN']
user in cluster 22 will be sugessted 3 courses as ['LB0101ENv1' 'LB0103ENv1' 'LB0105ENv1']
ser in cluster 23 will be sugessted 3 courses as ['BD0111EN' 'PY0101EN' 'BD0211EN']
r in cluster 24 will be sugessted 3 courses as ['CB0103EN' 'DS0101EN' 'BD0101EN']
```

Collaborative filtering-based recommendation system using supervised learning

1. CF using K Nearest Neighbour

Raw data

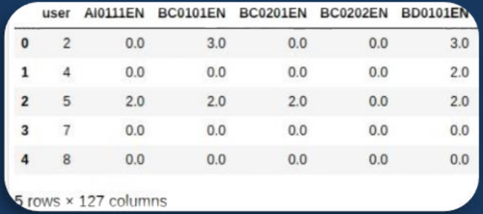
Spare data

Model

Prediction

User - item - rating - dataframe: `user_id, item, rating`

Use *pivot* method in pandas to turn data to features

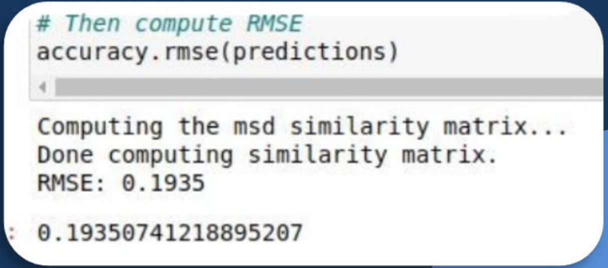


	user	AI0111EN	BC0101EN	BC0201EN	BC0202EN	BD0101EN
0	2	0.0	3.0	0.0	0.0	3.0
1	4	0.0	0.0	0.0	0.0	2.0
2	5	2.0	2.0	2.0	0.0	2.0
3	7	0.0	0.0	0.0	0.0	0.0
4	8	0.0	0.0	0.0	0.0	0.0

5 rows × 127 columns

Fit data to KNN model based on *surprise* library
Use distance metric listed in previous page

Make prediction by test data, use RMSE metric to evaluate model performance



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

```
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.1935

0.19350741218895207
```

2. CF using Non-negative Matrix Factorization -Pipeline & Result#

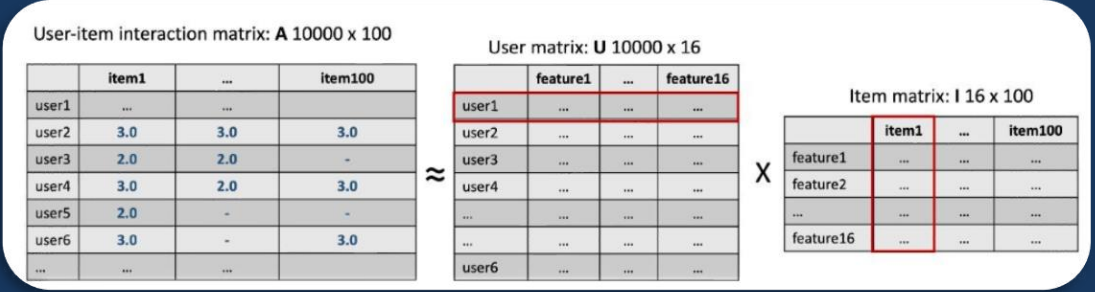
Raw data | Decomposed matrix | Model | Prediction

Raw data: User - item - rating - dataframe: `user_id, item, rating`

Decomposed matrix: Use *surprise* library to decompose full matrix to two smaller and denser ones: user matrix and item matrix

Model: Dot product each row in user matrix with each column in item matrix

Prediction: Make prediction by test data, use RMSE metric to evaluate model performance



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
...	...	...	

$\approx$

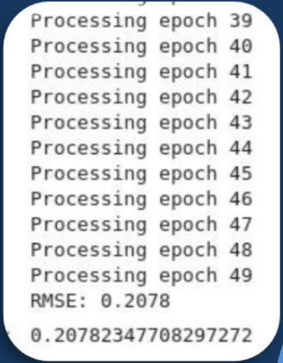
User matrix: **U** 10000 x 16

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

X

Item matrix: **I** 16 x 100

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

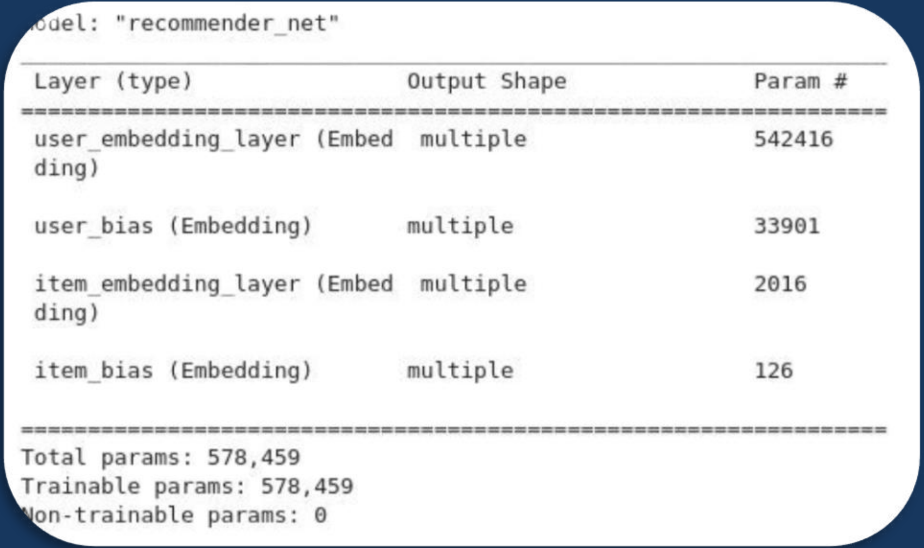


```
Processing epoch 39
Processing epoch 40
Processing epoch 41
Processing epoch 42
Processing epoch 43
Processing epoch 44
Processing epoch 45
Processing epoch 46
Processing epoch 47
Processing epoch 48
Processing epoch 49
RMSE: 0.2078

0.20782347708297272
```

3. Course Rating Prediction using Neural Networks

Model:

- Optimizer: Adam
- Loss: Mean Square Error
- Metric: Mean Square Error
- Epoch 12
- Batch size: 512



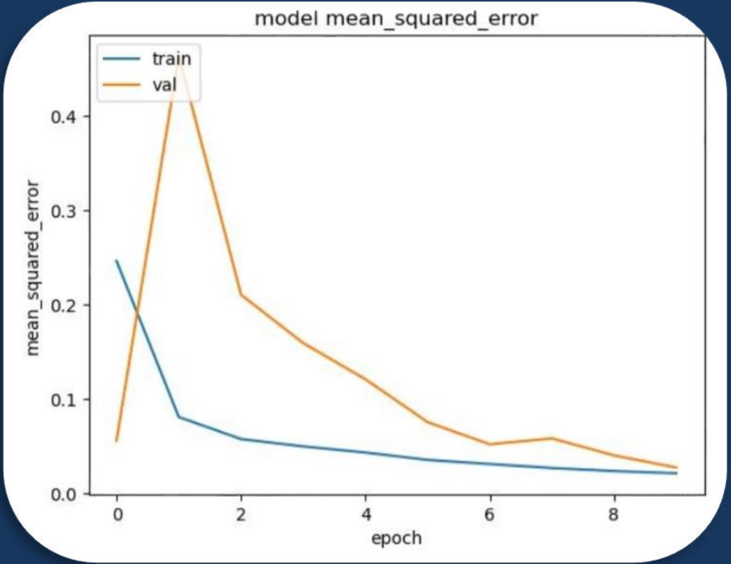
```
odel: "recommender_net"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 user_embedding_layer (Embed  multiple                 542416
 ding)

 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
on-trainable params: 0
```

3. Course Rating Prediction using Neural Networks

Result on test data:

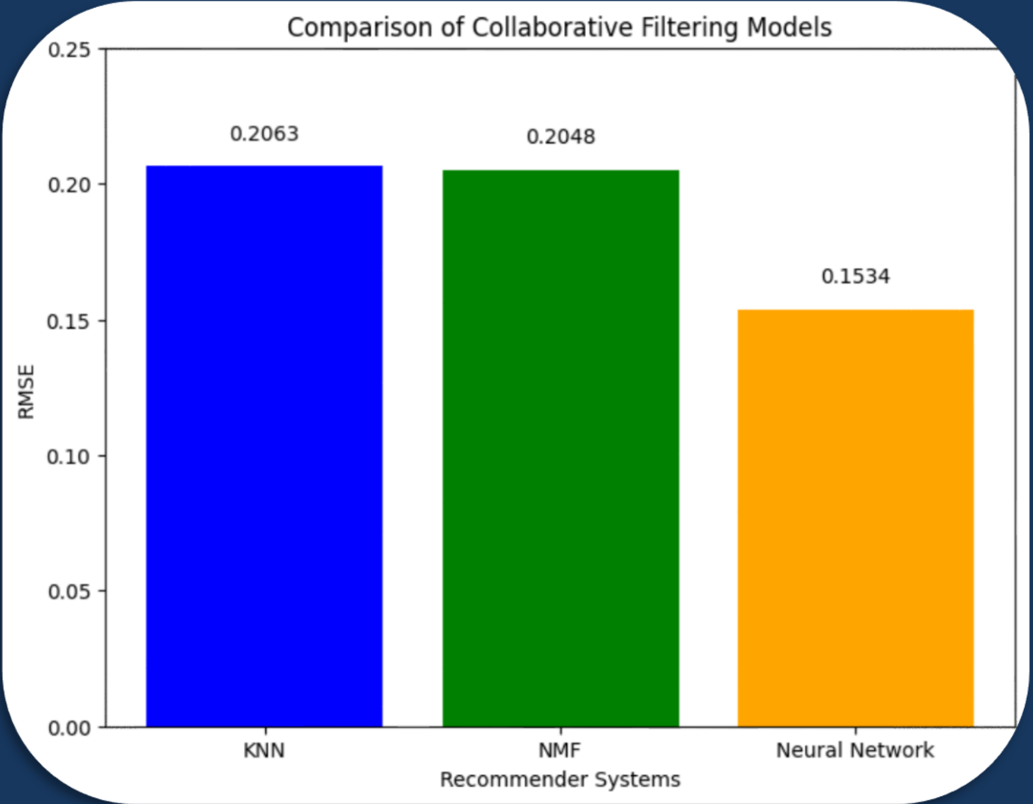
Mean squared error:: 0.0235

Root mean squared error: 0.1534



Compare the performance of collaborative-filtering models

Based on the evaluation results, the Neural Network Embedding-based recommender system achieved the lowest RMSE value of 0.1534, indicating the best performance among the three models in predicting user–item interactions. Therefore, we consider the Neural Network Embedding–based recommender system to be the most effective model for collaborative filtering in this scenario.



Conclusion

Conclusion of Exploratory Data Analysis EDA

From the various analyses conducted, several key insights can be drawn regarding the online learning platform's course offerings and user engagement. Firstly, the exploration of course genres revealed a diverse array of topics, with backend development, machine learning, and databases emerging as the most popular genres based on enrollment counts. Additionally, the analysis of course enrollments shed light on user behavior, indicating that a significant portion of users completed courses rather than merely auditing them. Furthermore, the examination of the top 20 most popular courses showcased a strong interest in data-related topics such as Python for Data Science, Introduction to Data Science, and Big Data 101, reflecting the growing demand for skills in data analysis and machine learning. Overall, these findings underscore the importance of providing a wide range of courses that cater to diverse interests while also highlighting the significance of data-driven insights in optimizing course offerings and enhancing user engagement on the platform.

Conclusion of Content-based recommender system using user profile

The Content-based Recommender System Using User Profile And Course Genres begins with setting up the task of generating course recommendations based on user profile and course genre vectors. The process involves loading user profiles and course genre dataframes, extracting user interests, identifying unknown courses for each user, computing recommendation scores, and filtering out courses below a specified threshold. After completing the recommendation scores for all test users, the analysis focuses on evaluating the system's performance. This includes determining the average number of recommended courses per user and identifying the top 10 most frequently recommended courses across all users. The average number of recommended courses per user is approximately **61.82**, indicating a substantial number of recommendations per user. The most frequently recommended courses include "TA0106EN," "GPXX0IBEN," and others, demonstrating their popularity among the recommended courses.

Conclusion of content-based recommender system using course similarity

The conclusion drawn from the content-based recommender system using course similarity is that a course similarity-based recommender system has been successfully implemented and evaluated. The system utilizes a similarity threshold of 0.6 to recommend courses to users based on their interests and previously selected courses. By analyzing course content and calculating similarities, the system provides personalized recommendations to users. The evaluation of the recommender system revealed valuable insights, including the average number of new courses recommended per user and the most frequently recommended courses. These findings highlight the effectiveness and performance of the system in suggesting relevant and engaging content to users. Additionally, the evaluation results inform potential optimizations and improvements for enhancing the recommender system's effectiveness in future iterations. Overall, the conclusion underscores the importance and effectiveness of utilizing course similarity-based approaches in providing personalized recommendations to users in educational settings.

Conclusion of cluster-based recommender system

In conclusion, our course clustering-based recommender system demonstrates robust performance in effectively grouping users based on their preferences and recommending relevant courses. The optimization of hyperparameters, including the determination of cluster numbers and PCA components, ensures that the system accurately captures user preferences while minimizing information loss. Through meticulous evaluation, we found that the system recommends an average of approximately 36.587 new or unseen courses per user in the test dataset, indicating its effectiveness in diversifying recommendations. Furthermore, the identification of frequently recommended courses such as "WA0101EN," "DB0101EN," and "DS0301EN" provides valuable insights into user preferences and highlights the system's ability to promote popular courses across different clusters. Overall, our recommender system serves as a valuable tool for enhancing user engagement and satisfaction by delivering personalized course recommendations aligned with individual learning objectives.

Conclusion of collaborative filtering (KNN, NMF, and Neural Network)

Based on the comparative analysis of the three collaborative filtering methods, including KNN-based, NMF-based, and Neural Network Embedding-based recommender systems, several insights can be gleaned. Firstly, both the KNN-based and NMF- based recommender systems exhibit relatively higher RMSE values compared to the Neural Network Embedding-based approach. This suggests that the **Neural Network Embedding-based** method better captures the underlying patterns and latent features within the user-item interaction data, resulting in more accurate predictions. Additionally, the Neural Network Embedding-based approach benefits from its ability to learn complex nonlinear relationships between users and items, allowing it to uncover subtle preferences and nuances in the data. However, it's worth noting that while the Neural Network Embedding- based method outperforms the other two approaches in terms of prediction accuracy, it may require more computational resources and longer training times due to its deeper architecture and higher complexity. Therefore, the choice of the most suitable collaborative filtering method depends on the specific requirements and constraints of the recommendation system, balancing between prediction performance and computational efficiency.

	KNN	**NMF**	**ANN**
RMSE	0.2063	0.2048	0.1534

Future Improvements

More improvements in the Future

This project showcases the implementation of an end-to-end machine learning pipeline. While it successfully meets the course objectives, there are several potential improvements to enhance accuracy and overall performance:

- Working with real-world customer data for more practical insights.
- Incorporating advanced pre-processing techniques.
- Efficiently managing sparse data to avoid excessive memory usage.

Thanks for your reading!