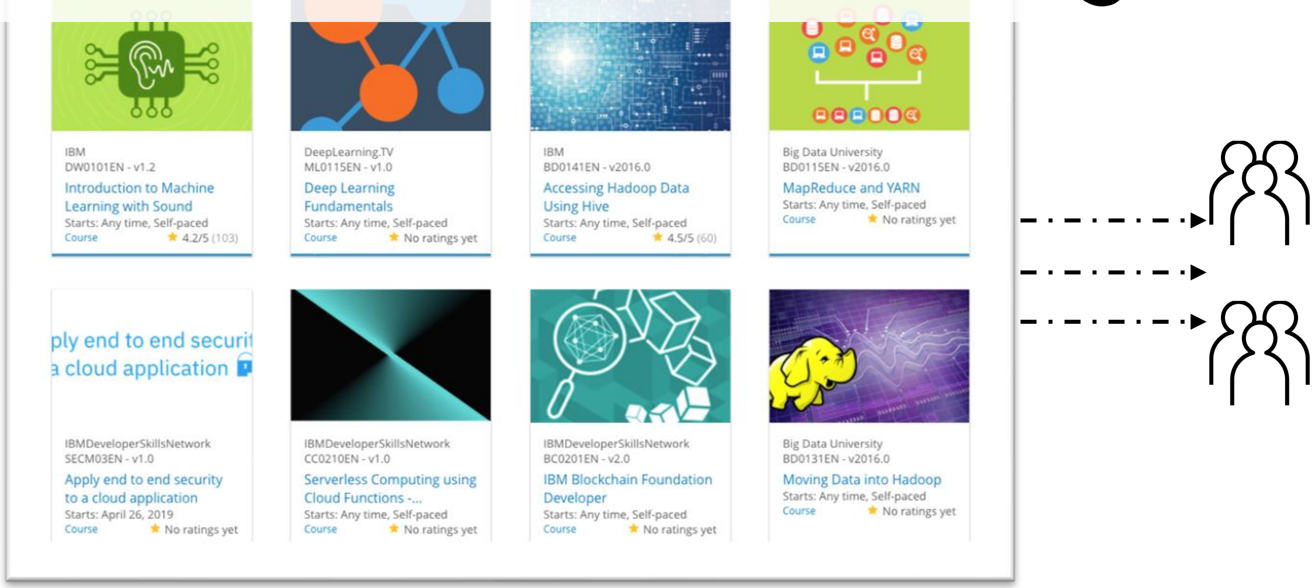
Build a Personalized Online Course Recommender System with Machine Learning

Walter Roye T. Fanka
29 August 2025



IBM
DW0101EN - v1.2
Introduction to Machine Learning with Sound
Starts: Any time, Self-paced
Course ★ 4.2/5 (103)

DeepLearning.TV
ML0115EN - v1.0
Deep Learning Fundamentals
Starts: Any time, Self-paced
Course ★ No ratings yet

IBM
BD0141EN - v2016.0
Accessing Hadoop Data Using Hive
Starts: Any time, Self-paced
Course ★ 4.5/5 (60)

Big Data University
BD0115EN - v2016.0
MapReduce and YARN
Starts: Any time, Self-paced
Course ★ No ratings yet

IBMDeveloperSkillsNetwork
SECM03EN - v1.0
Apply end to end security to a cloud application
Starts: April 26, 2019
Course ★ No ratings yet

IBMDeveloperSkillsNetwork
CC0210EN - v1.0
Serverless Computing using Cloud Functions -...
Starts: Any time, Self-paced
Course ★ No ratings yet

IBMDeveloperSkillsNetwork
BC0201EN - v2.0
IBM Blockchain Foundation Developer
Starts: Any time, Self-paced
Course ★ No ratings yet

Big Data University
BD0131EN - v2016.0
Moving Data into Hadoop
Starts: Any time, Self-paced
Course ★ No ratings yet

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

This project focuses on building a personalized online course recommender system using machine learning. The rapid growth of digital learning platforms has created an overwhelming amount of content, making it challenging for learners to identify courses most relevant to their needs and interests. A recommender system addresses this challenge by leveraging data to provide tailored suggestions.

- Problem states

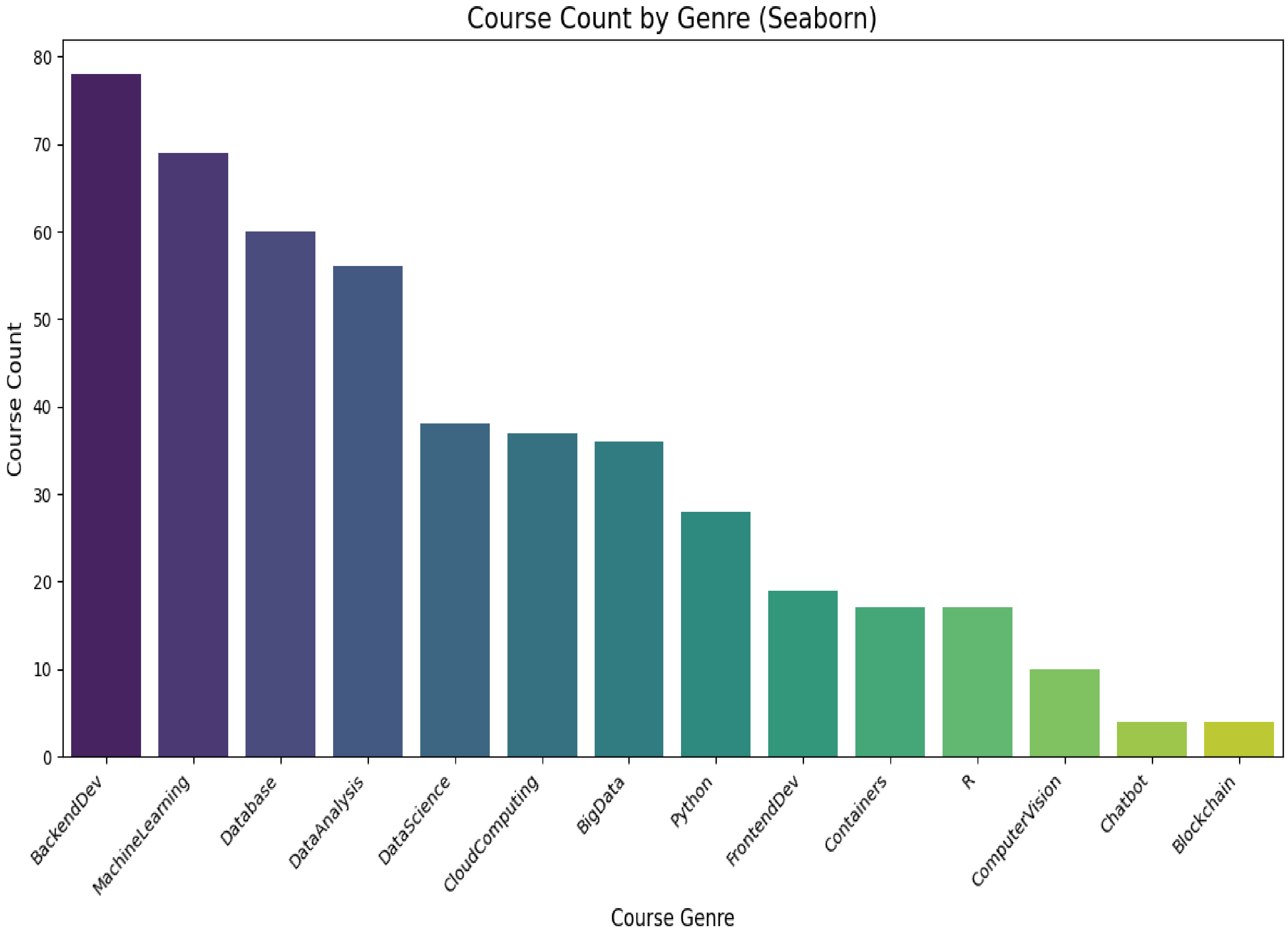
The problem at hand is that learners are often left with too many choices, leading to decision fatigue, under-enrollment in valuable courses, and difficulty sustaining engagement. By analyzing course features, user preferences, and historical activity, this project explores both content-based and collaborative-filtering methods to enhance course discovery and improve learner satisfaction.

- Hypotheses:

1. A content-based recommender system can cluster courses and suggest items aligned with a learner’s profile, improving relevance.
2. Collaborative-filtering models can capture hidden patterns in user-course interactions to provide more accurate recommendations.
3. Combining both approaches will outperform a single method in terms of accuracy, coverage, and learner satisfaction.

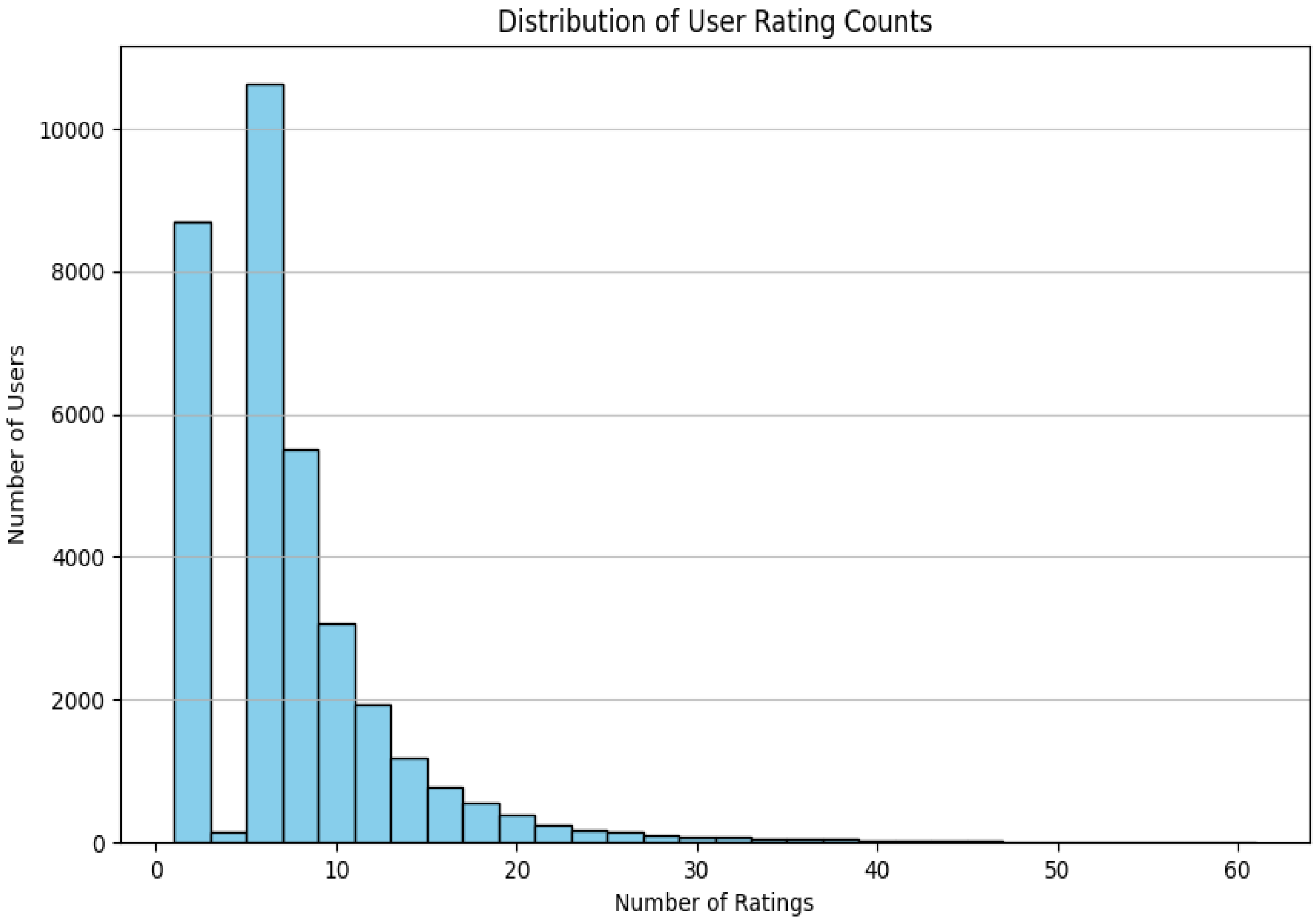
Exploratory Data Analysis

Course counts per genre



The chart provides a simple and effective visual summary of the data, allowing you to quickly identify the most popular and least popular course genres. For example, the tallest bar represents the genre with the most courses, while the shortest bars represent the genres with the fewest courses.

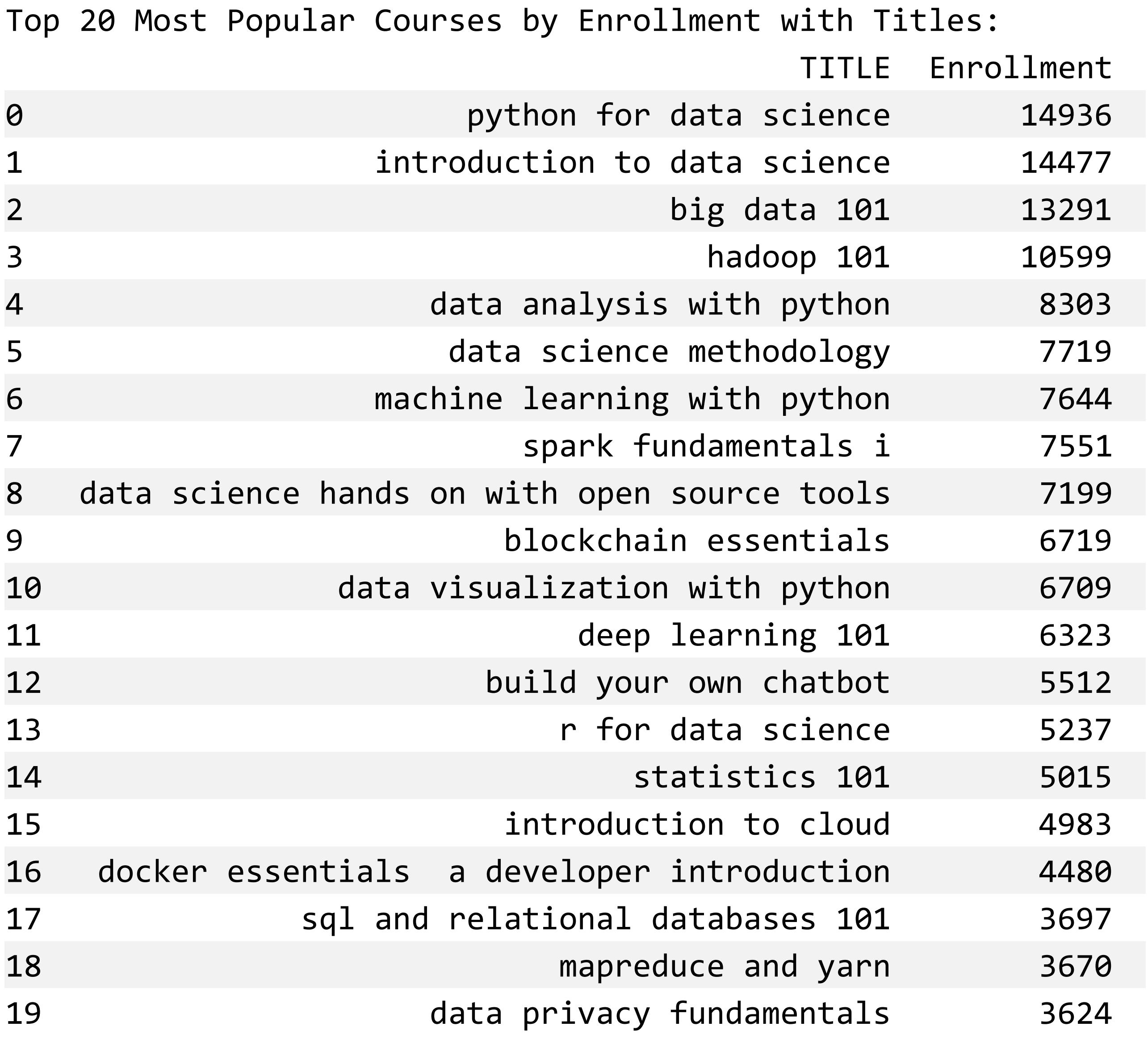
Course enrollment distribution



The histogram visualizes the distribution of user activity. It shows how many users have provided a certain number of ratings. The x-axis represents the number of ratings given by a user, and the y-axis represents the number of users who fall into that category.

The histogram is likely to be right-skewed, which is a common pattern in user engagement data. This would indicate that a large number of users have made only a few ratings, while a small number of "super-users" have contributed a high number of ratings.

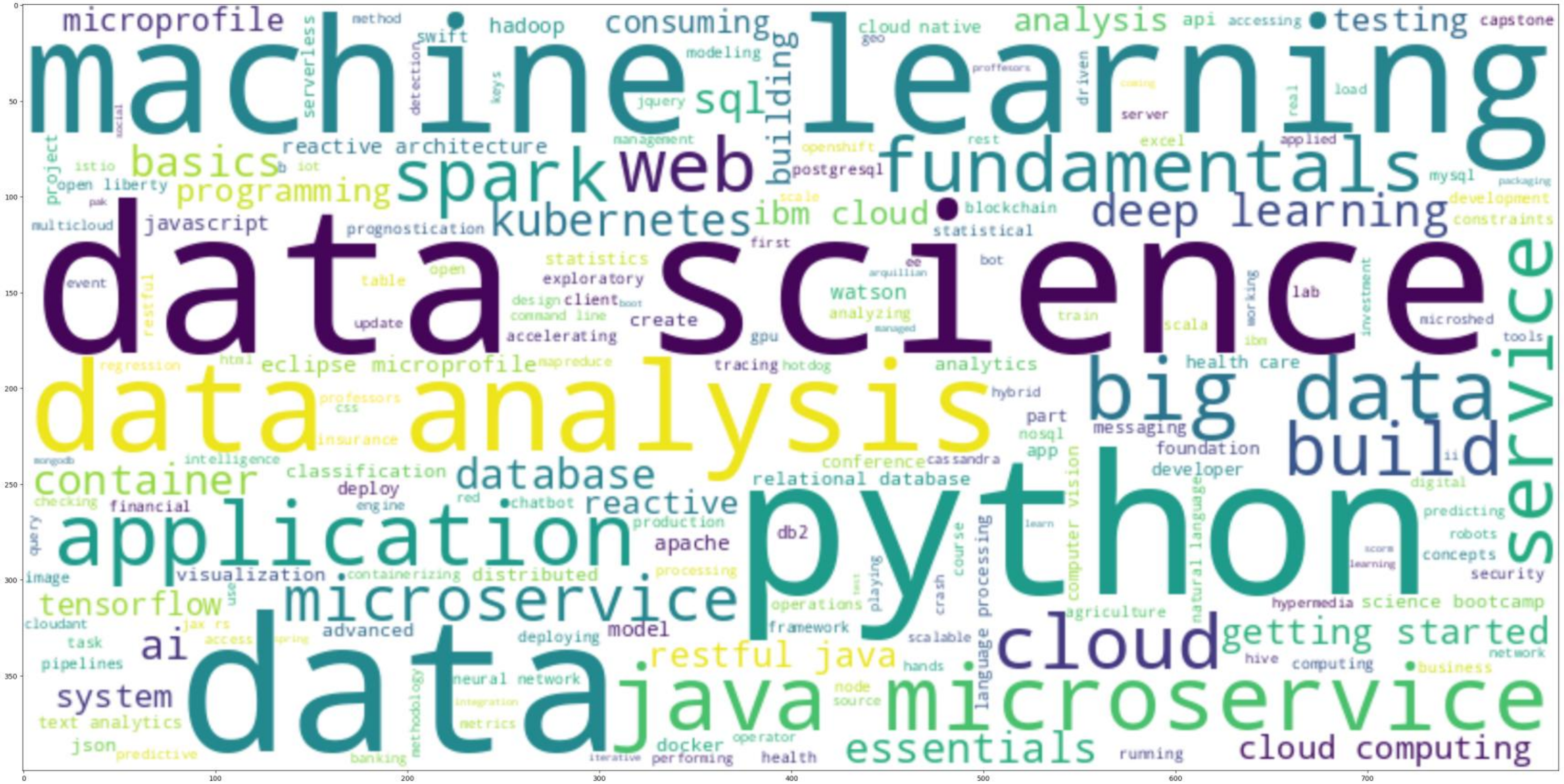
20 most popular courses



```
Top 20 Most Popular Courses by Enrollment with Titles:
                                          TITLE  Enrollment
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
```

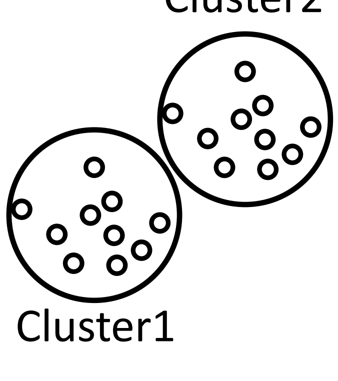
The table shows that Python for Data Science is the most popular course by enrollment while Data Privacy Fundamentals is the least among the top 20 most popular courses.

Word cloud of course titles

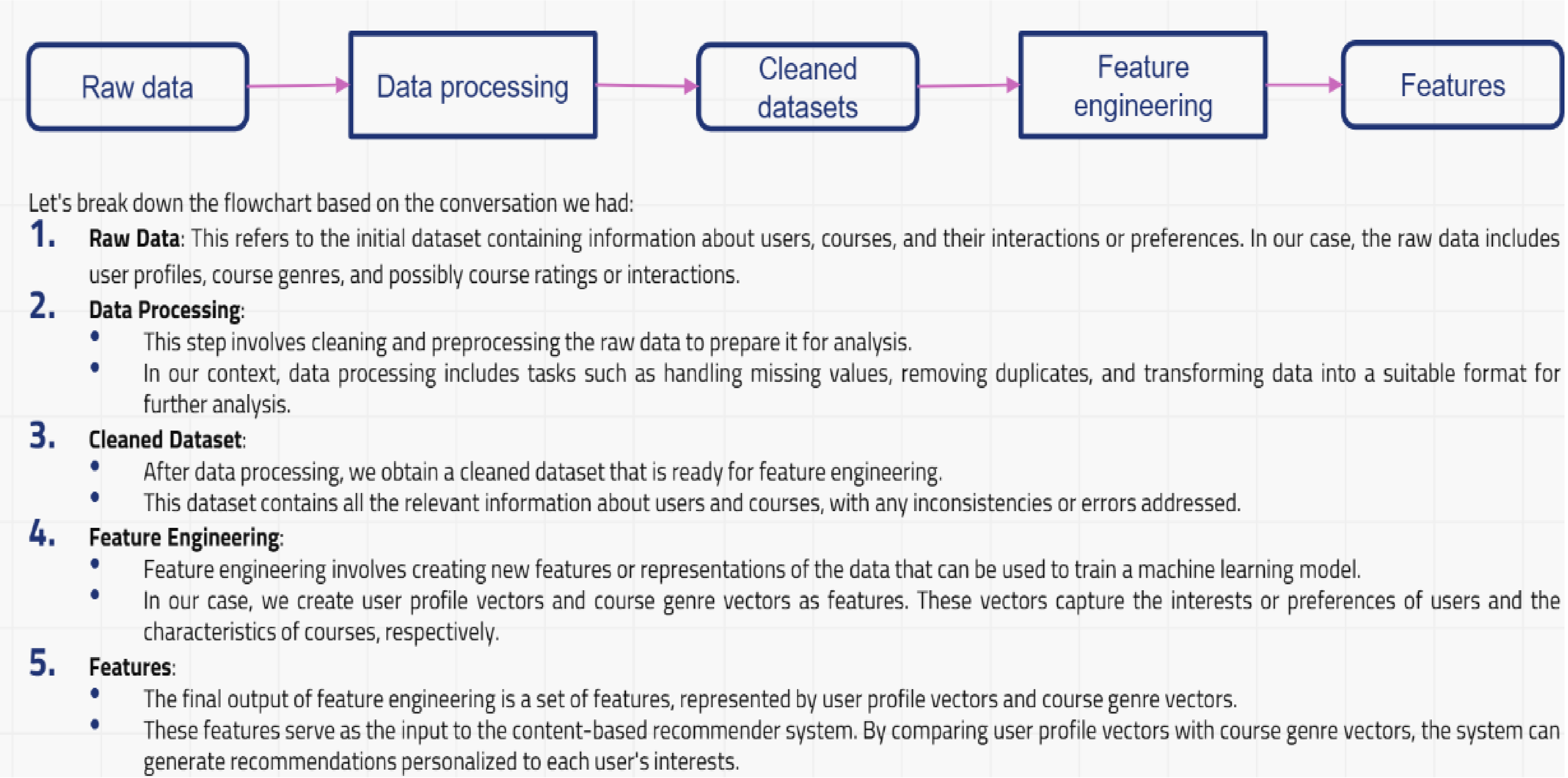


Visually, we see that the most used words are data science, data, python, and machine learning. This is closely followed by data analysis. While these are clearly legible, we notice several other words that are not legible at all unless one were to zoom them out.

Content-based Recommender System using Unsupervised Learning



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

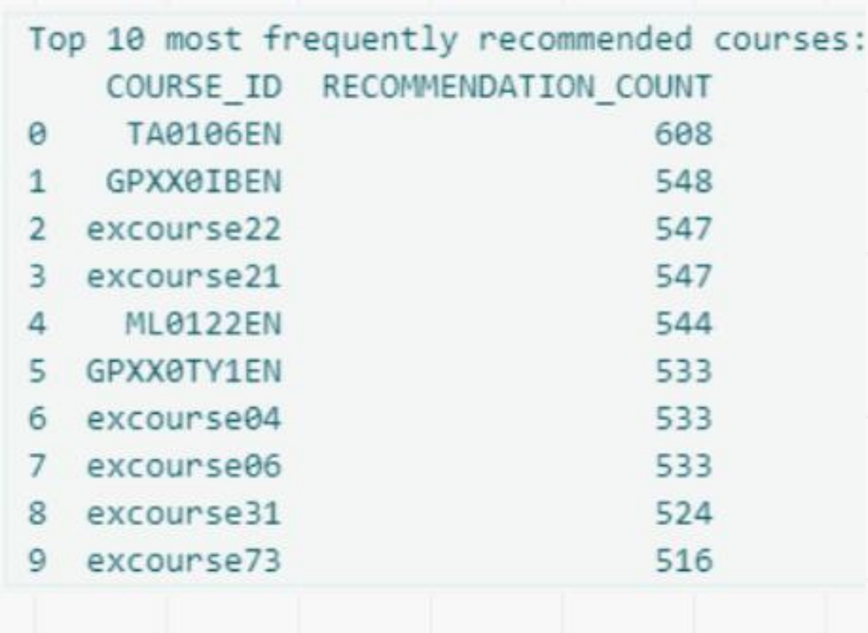


Raw data → Data processing → Cleaned datasets → Feature engineering → Features

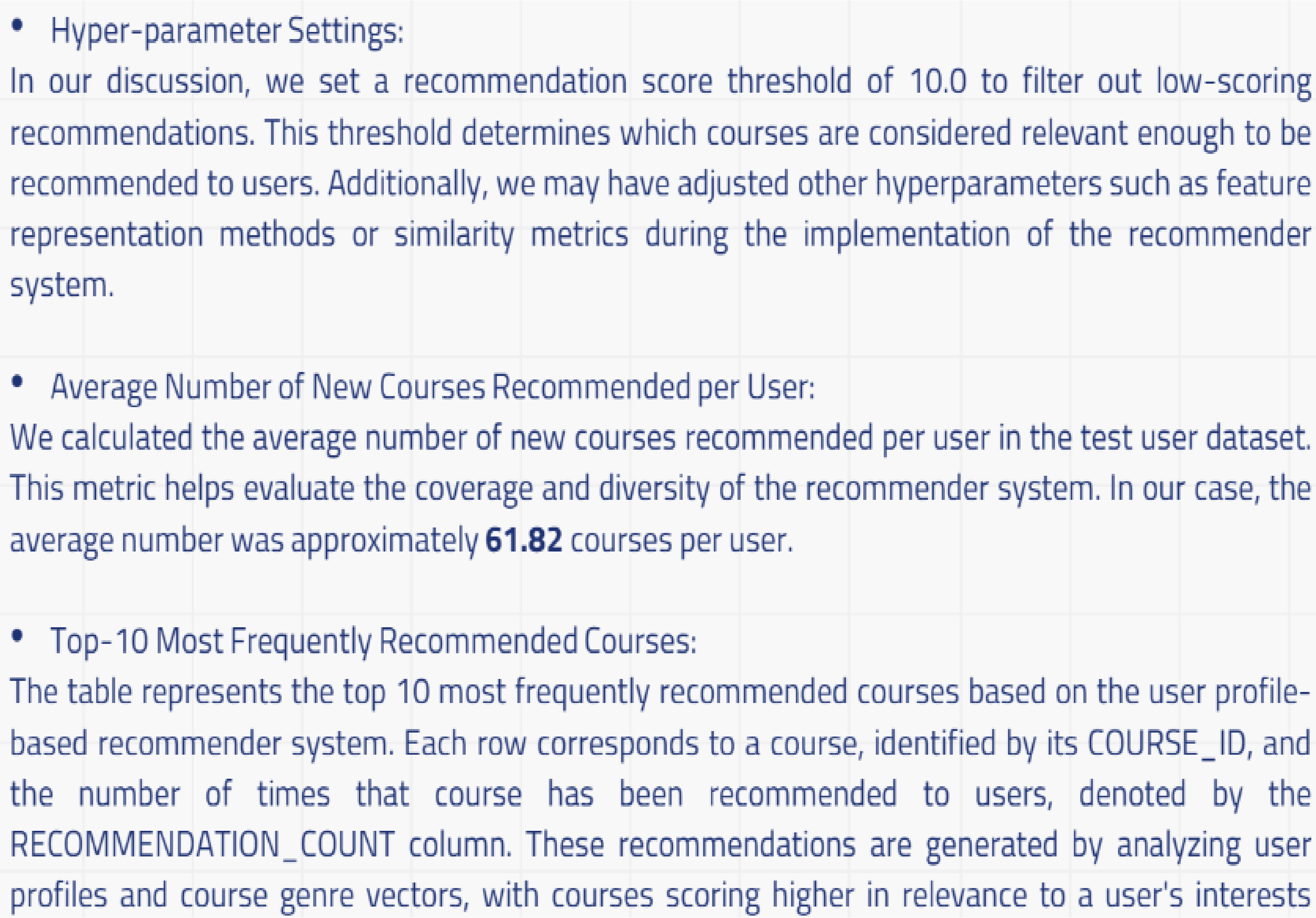
Let's break down the flowchart based on the conversation we had:

1. **Raw Data**: This refers to the initial dataset containing information about users, courses, and their interactions or preferences. In our case, the raw data includes user profiles, course genres, and possibly course ratings or interactions.
2. **Data Processing**:
   - This step involves cleaning and preprocessing the raw data to prepare it for analysis.
   - In our context, data processing includes tasks such as handling missing values, removing duplicates, and transforming data into a suitable format for further analysis.
3. **Cleaned Dataset**:
   - After data processing, we obtain a cleaned dataset that is ready for feature engineering.
   - This dataset contains all the relevant information about users and courses, with any inconsistencies or errors addressed.
4. **Feature Engineering**:
   - Feature engineering involves creating new features or representations of the data that can be used to train a machine learning model.
   - In our case, we create user profile vectors and course genre vectors as features. These vectors capture the interests or preferences of users and the characteristics of courses, respectively.
5. **Features**:
   - The final output of feature engineering is a set of features, represented by user profile vectors and course genre vectors.
   - These features serve as the input to the content-based recommender system. By comparing user profile vectors with course genre vectors, the system can generate recommendations personalized to each user's interests.

Evaluation results of user profile-based recommender system



```
Top 10 most frequently recommended courses:
    COURSE_ID  RECOMMENDATION_COUNT
0    TA0106EN                   608
1   GPXX0IBEN                   548
2  excourse22                   547
3  excourse21                   547
4    ML0122EN                   544
5  GPXX0TY1EN                   533
6  excourse04                   533
7  excourse06                   533
8  excourse31                   524
9  excourse73                   516
```



- 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.

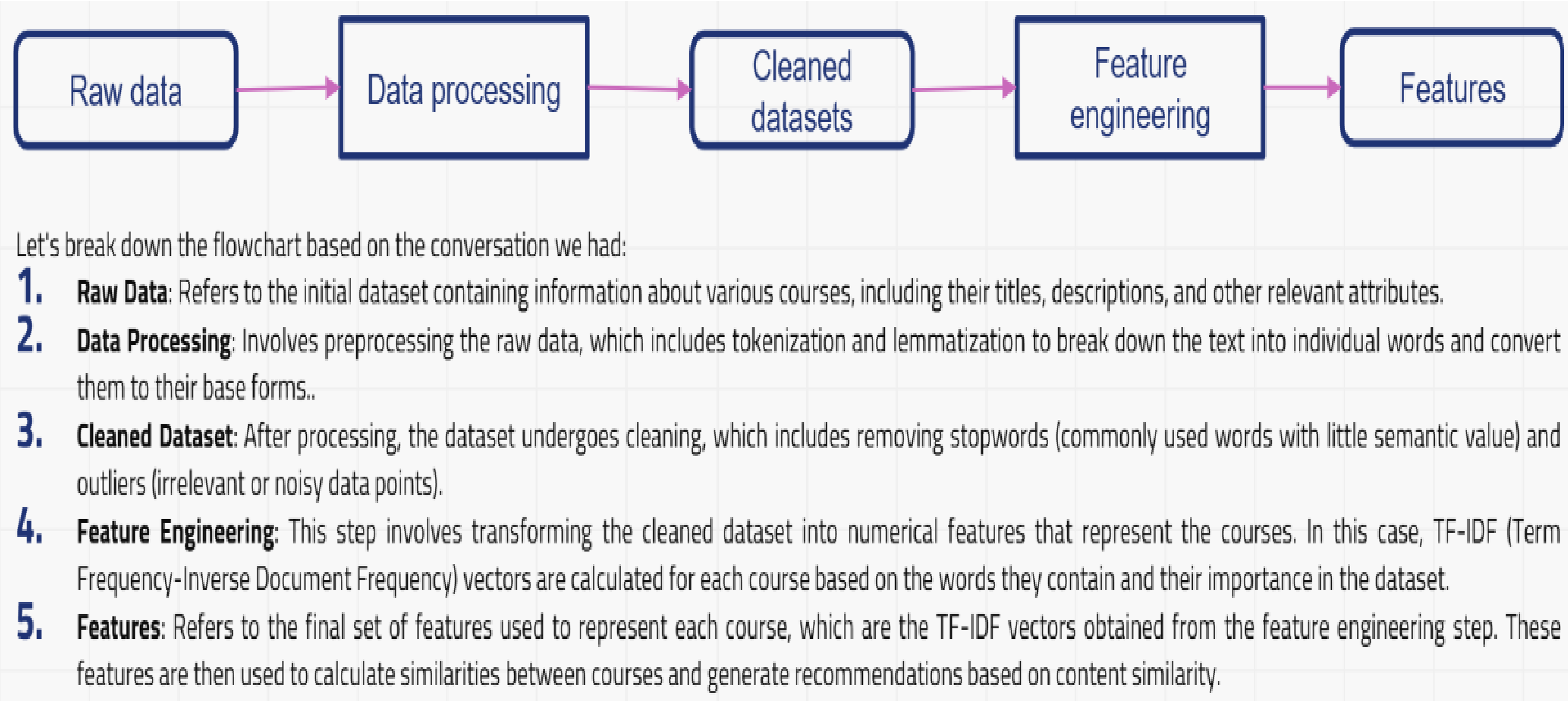
- Average Number of New Courses Recommended per User:

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

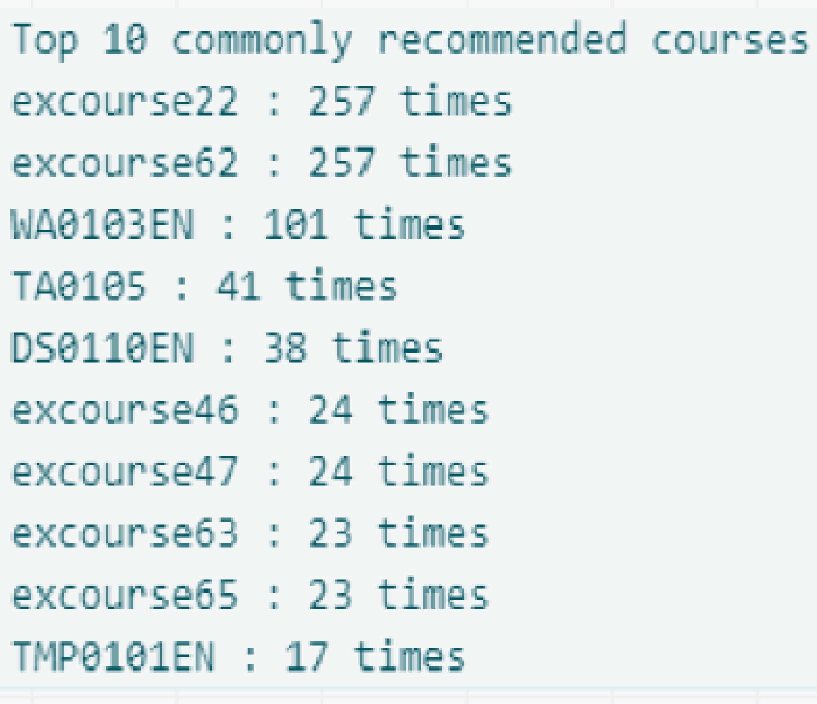
Flowchart of content-based recommender system using course similarity



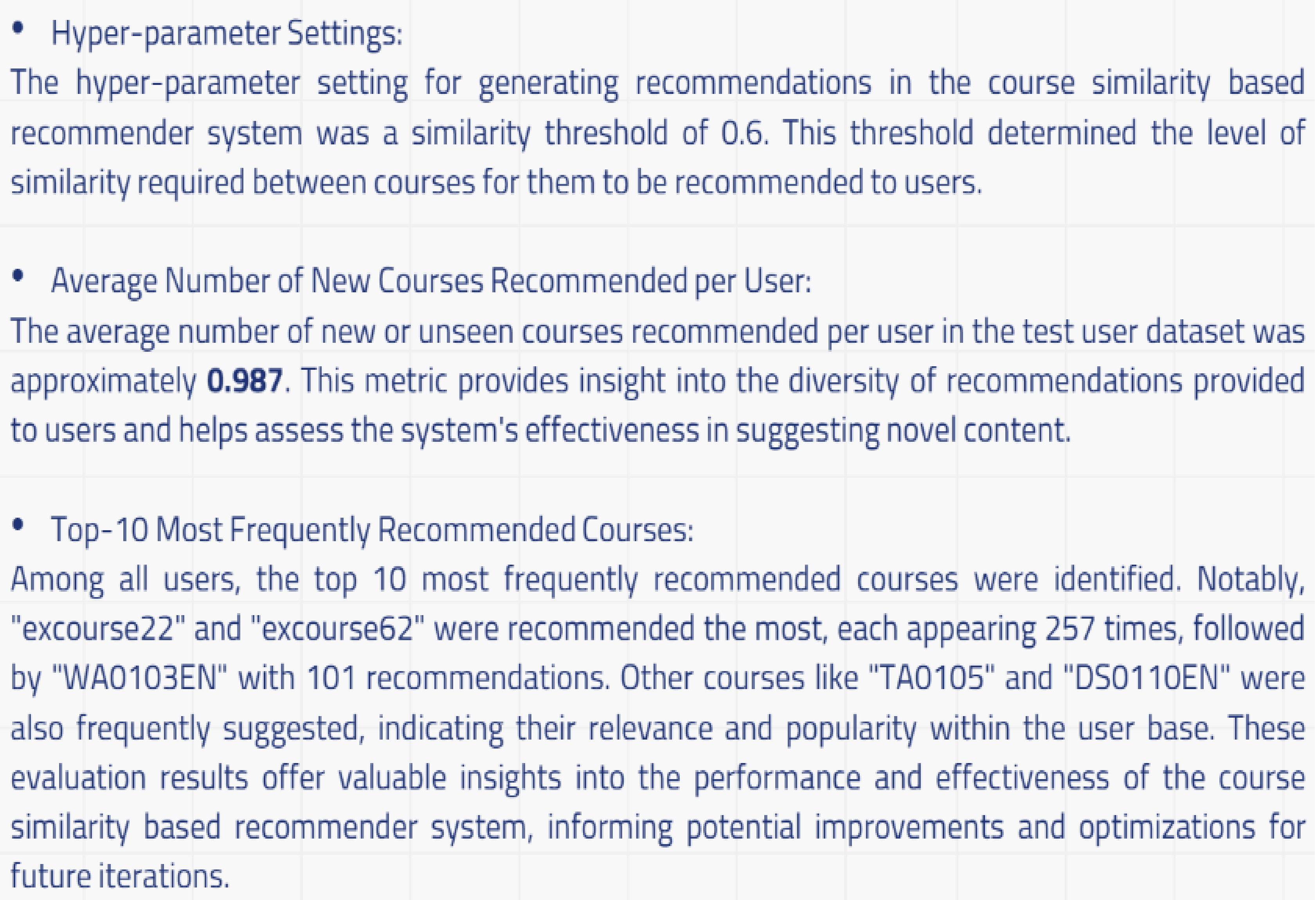
Let's break down the flowchart based on the conversation we had:

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.

Evaluation results of course similarity based recommender system



```
Top 10 commonly recommended courses:
excourse22 : 257 times
excourse62 : 257 times
WA0103EN : 101 times
TA0105 : 41 times
DS0110EN : 38 times
excourse46 : 24 times
excourse47 : 24 times
excourse63 : 23 times
excourse65 : 23 times
TMP0101EN : 17 times
```



- Hyper-parameter Settings:

The hyper-parameter setting for generating recommendations in the course similarity based recommender system was a similarity threshold of 0.6. This threshold determined the level of similarity required between courses for them to be recommended to users.

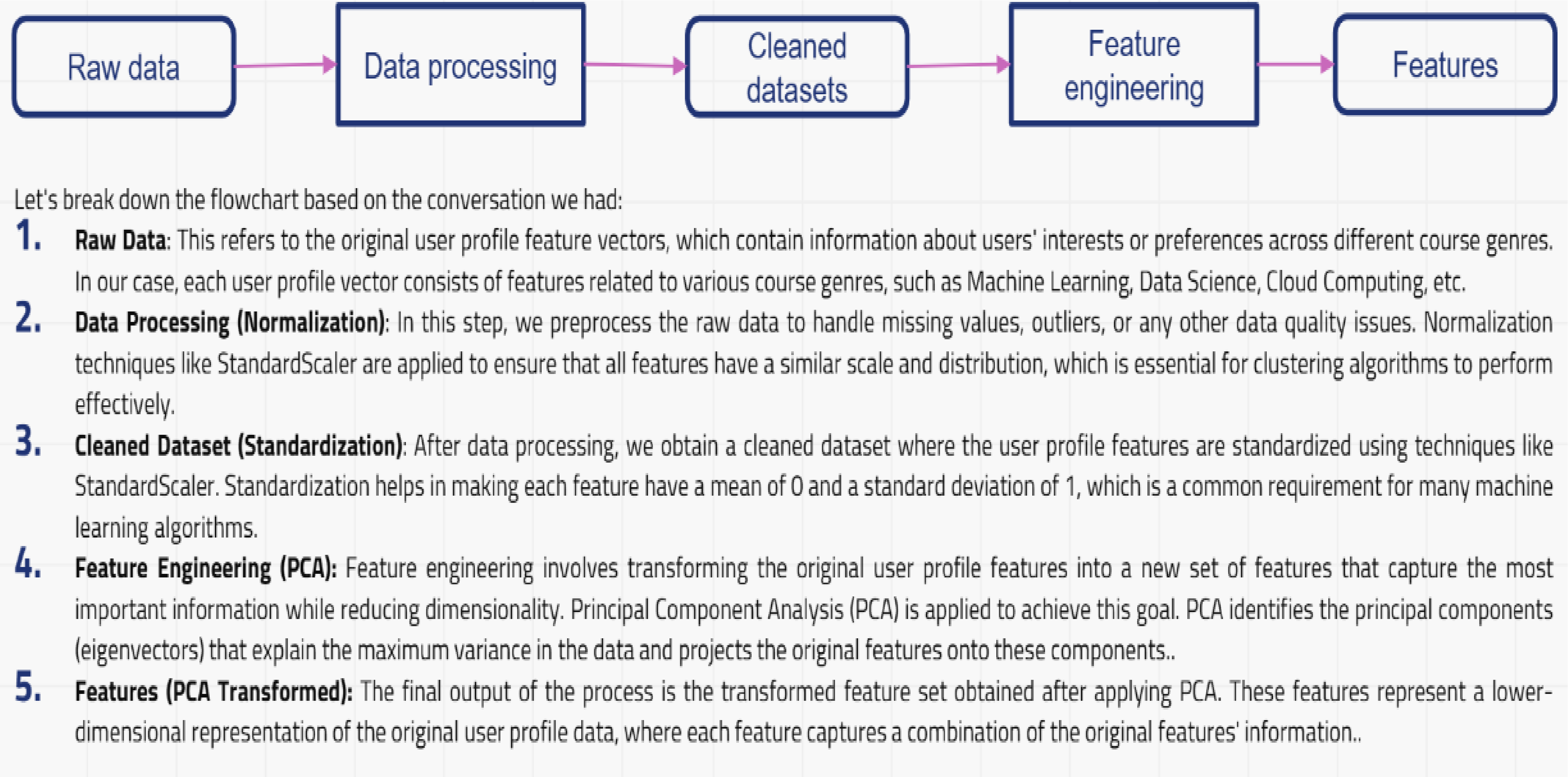
- Average Number of New Courses Recommended per User:

The average number of new or unseen courses recommended per user in the test user dataset was approximately **0.987**. This metric provides insight into the diversity of recommendations provided to users and helps assess the system's effectiveness in suggesting novel content.

- Top-10 Most Frequently Recommended Courses:

Among all users, the top 10 most frequently recommended courses were identified. Notably, "excourse22" and "excourse62" were recommended the most, each appearing 257 times, followed by "WA0103EN" with 101 recommendations. Other courses like "TA0105" and "DS0110EN" were also frequently suggested, indicating their relevance and popularity within the user base. These evaluation results offer valuable insights into the performance and effectiveness of the course similarity based recommender system, informing potential improvements and optimizations for future iterations.

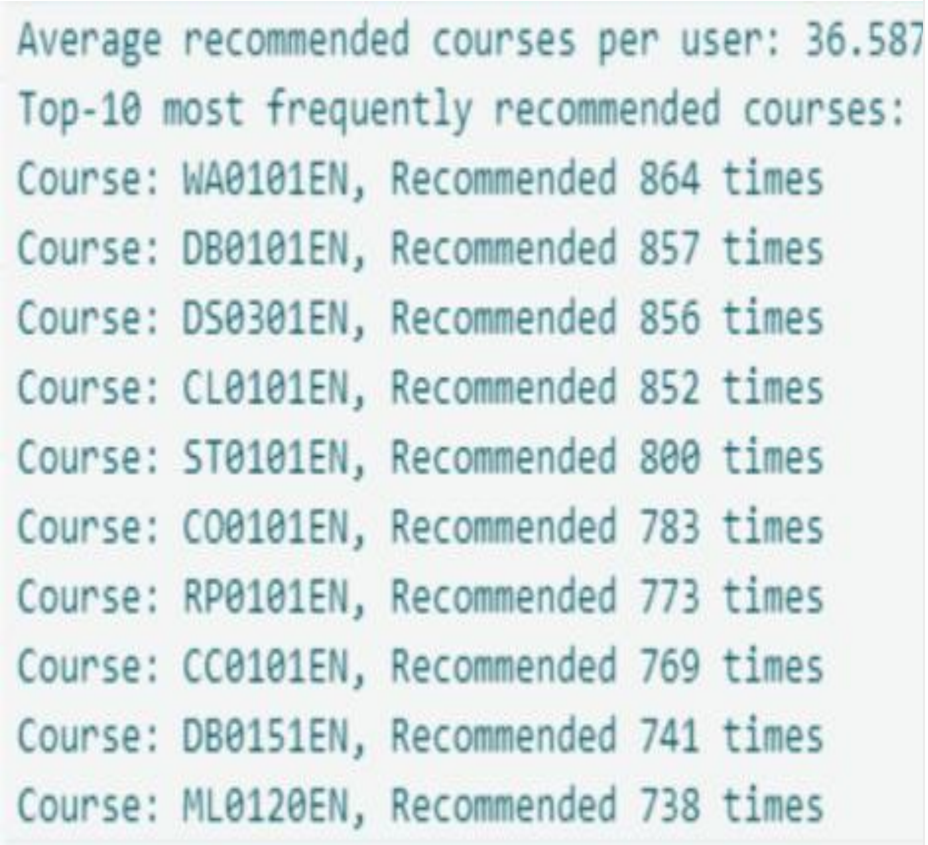
Flowchart of clustering-based recommender system



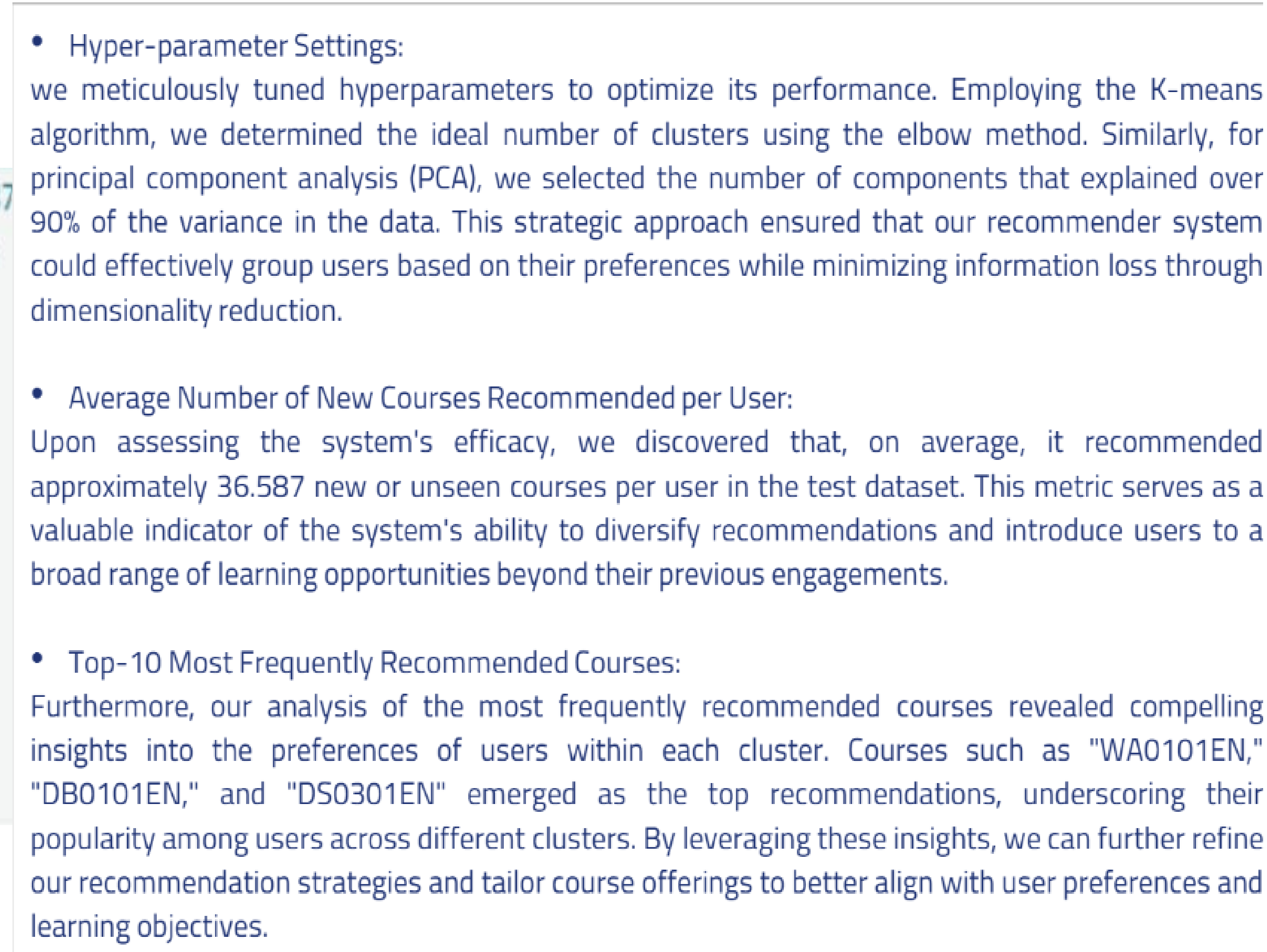
Let's break down the flowchart based on the conversation we had:

1. **Raw Data**: This refers to the original user profile feature vectors, which contain information about users' interests or preferences across different course genres. In our case, each user profile vector consists of features related to various course genres, such as Machine Learning, Data Science, Cloud Computing, etc.
2. **Data Processing (Normalization)**: In this step, we preprocess the raw data to handle missing values, outliers, or any other data quality issues. Normalization techniques like StandardScaler are applied to ensure that all features have a similar scale and distribution, which is essential for clustering algorithms to perform effectively.
3. **Cleaned Dataset (Standardization)**: After data processing, we obtain a cleaned dataset where the user profile features are standardized using techniques like StandardScaler. Standardization helps in making each feature have a mean of 0 and a standard deviation of 1, which is a common requirement for many machine learning algorithms.
4. **Feature Engineering (PCA):** Feature engineering involves transforming the original user profile features into a new set of features that capture the most important information while reducing dimensionality. Principal Component Analysis (PCA) is applied to achieve this goal. PCA identifies the principal components (eigenvectors) that explain the maximum variance in the data and projects the original features onto these components..
5. **Features (PCA Transformed):** The final output of the process is the transformed feature set obtained after applying PCA. These features represent a lower-dimensional representation of the original user profile data, where each feature captures a combination of the original features' information..

Evaluation results of clustering-based recommender system



```
Average recommended courses per user: 36.587
Top-10 most frequently recommended courses:
Course: WA0101EN, Recommended 864 times
Course: DB0101EN, Recommended 857 times
Course: DS0301EN, Recommended 856 times
Course: CL0101EN, Recommended 852 times
Course: ST0101EN, Recommended 800 times
Course: CO0101EN, Recommended 783 times
Course: RP0101EN, Recommended 773 times
Course: CC0101EN, Recommended 769 times
Course: DB0151EN, Recommended 741 times
Course: ML0120EN, Recommended 738 times
```



- Hyper-parameter Settings:

we meticulously tuned hyperparameters to optimize its performance. Employing the K-means algorithm, we determined the ideal number of clusters using the elbow method. Similarly, for principal component analysis (PCA), we selected the number of components that explained over 90% of the variance in the data. This strategic approach ensured that our recommender system could effectively group users based on their preferences while minimizing information loss through dimensionality reduction.

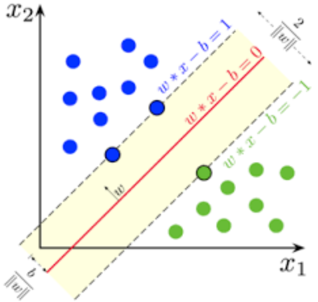
- Average Number of New Courses Recommended per User:

Upon assessing the system's efficacy, we discovered that, on average, it recommended approximately 36.587 new or unseen courses per user in the test dataset. This metric serves as a valuable indicator of the system's ability to diversify recommendations and introduce users to a broad range of learning opportunities beyond their previous engagements.

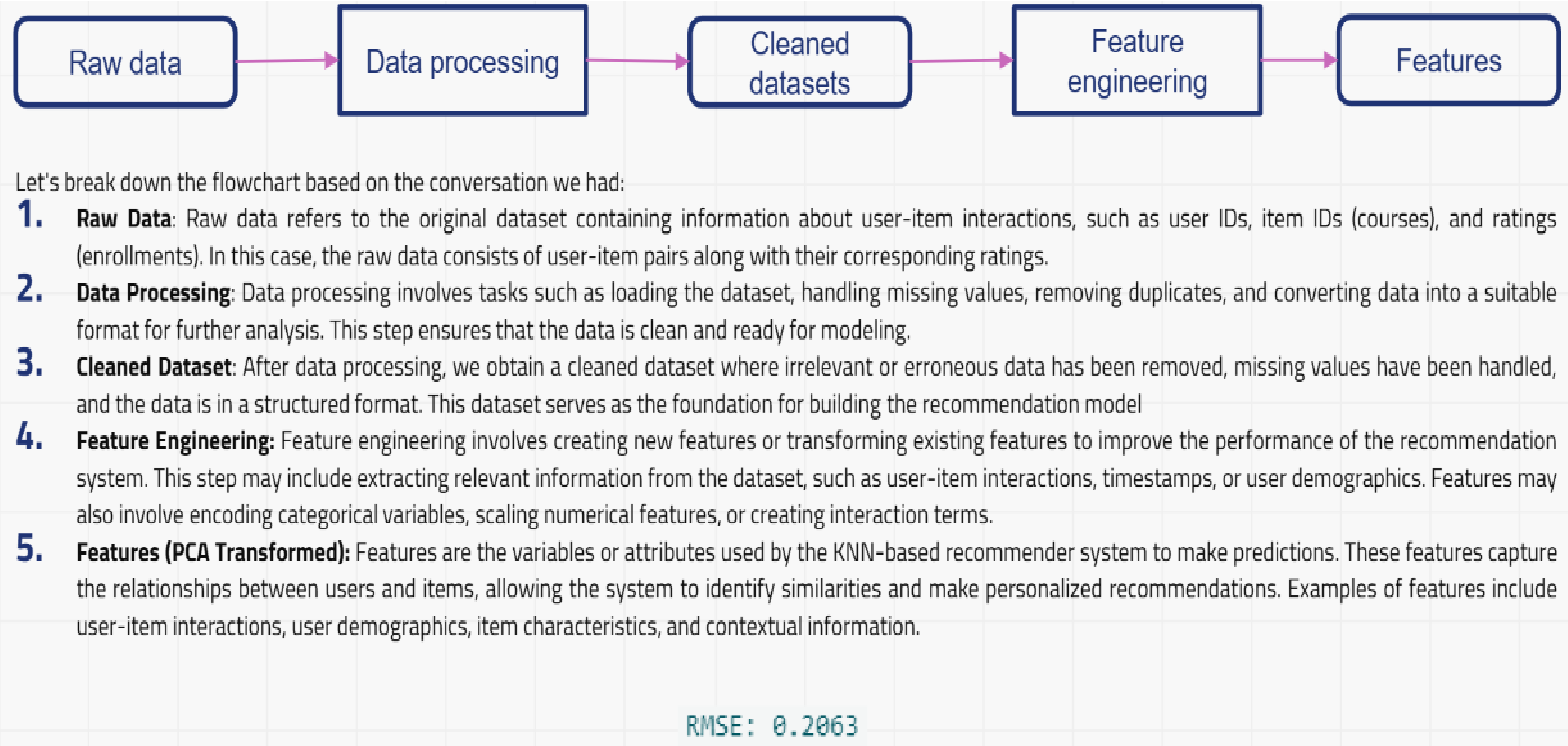
- Top-10 Most Frequently Recommended Courses:

Furthermore, our analysis of the most frequently recommended courses revealed compelling insights into the preferences of users within each cluster. Courses such as "WA0101EN," "DB0101EN," and "DS0301EN" emerged as the top recommendations, underscoring their popularity among users across different clusters. By leveraging these insights, we can further refine our recommendation strategies and tailor course offerings to better align with user preferences and learning objectives.

Collaborative-filtering Recommender System using Supervised Learning



Flowchart of KNN based recommender system

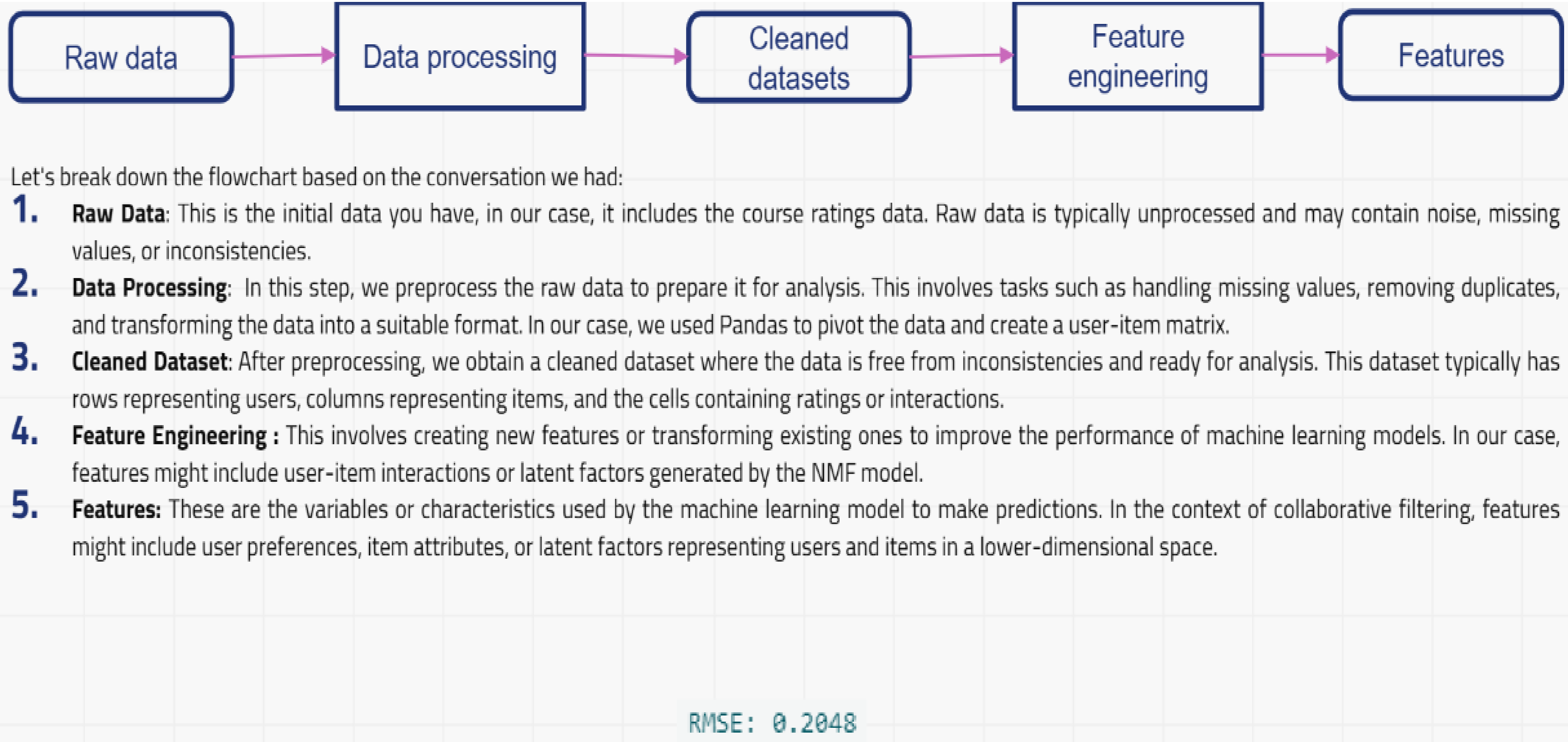


Let's break down the flowchart based on the conversation we had:

1. **Raw Data**: Raw data refers to the original dataset containing information about user-item interactions, such as user IDs, item IDs (courses), and ratings (enrollments). In this case, the raw data consists of user-item pairs along with their corresponding ratings.
2. **Data Processing**: Data processing involves tasks such as loading the dataset, handling missing values, removing duplicates, and converting data into a suitable format for further analysis. This step ensures that the data is clean and ready for modeling.
3. **Cleaned Dataset**: After data processing, we obtain a cleaned dataset where irrelevant or erroneous data has been removed, missing values have been handled, and the data is in a structured format. This dataset serves as the foundation for building the recommendation model
4. **Feature Engineering:** Feature engineering involves creating new features or transforming existing features to improve the performance of the recommendation system. This step may include extracting relevant information from the dataset, such as user-item interactions, timestamps, or user demographics. Features may also involve encoding categorical variables, scaling numerical features, or creating interaction terms.
5. **Features (PCA Transformed):** Features are the variables or attributes used by the KNN-based recommender system to make predictions. These features capture the relationships between users and items, allowing the system to identify similarities and make personalized recommendations. Examples of features include user-item interactions, user demographics, item characteristics, and contextual information.

```
RMSE: 0.2063
```

Flowchart of NMF based recommender system

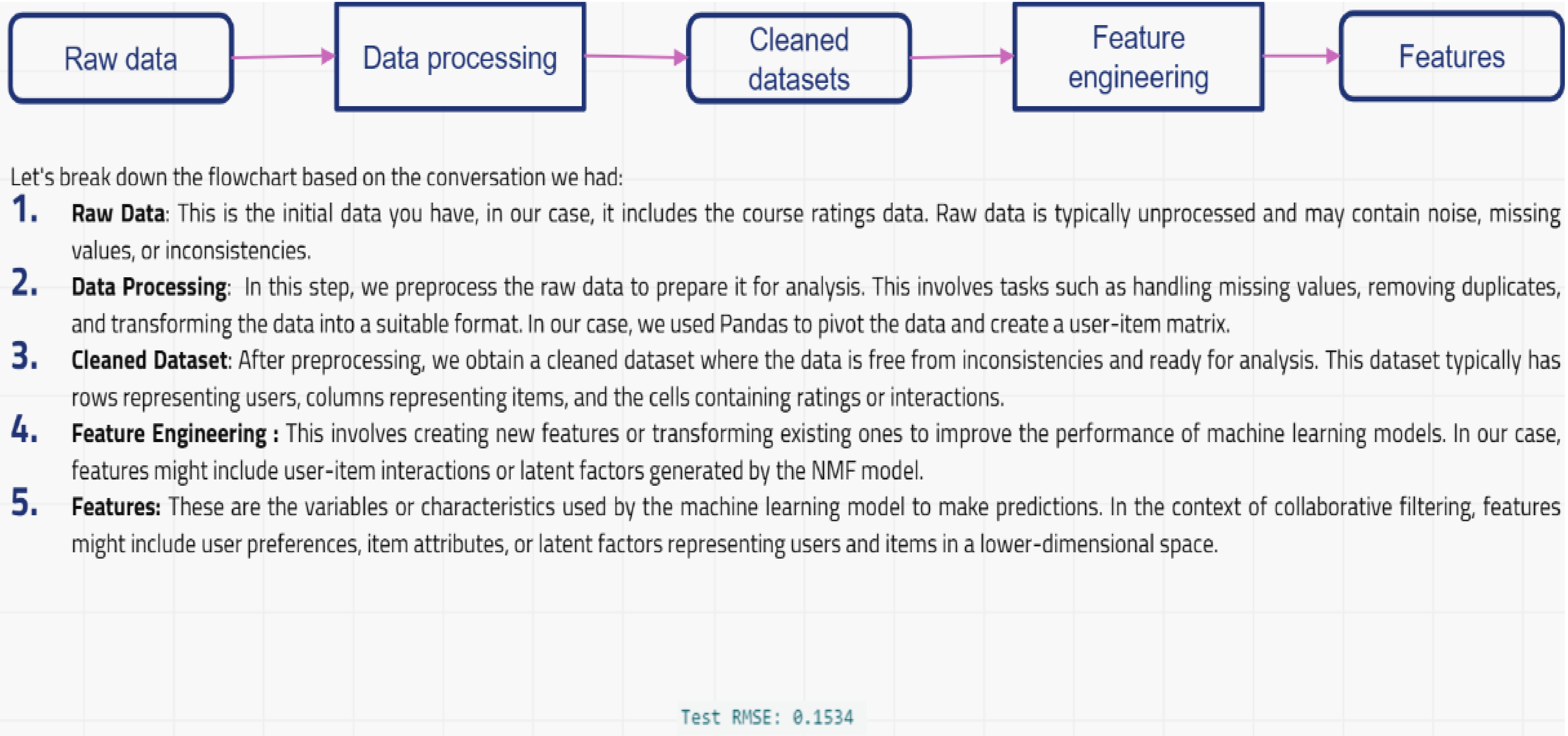


Let's break down the flowchart based on the conversation we had:

1. **Raw Data**: This is the initial data you have, in our case, it includes the course ratings data. Raw data is typically unprocessed and may contain noise, missing values, or inconsistencies.
2. **Data Processing**: In this step, we preprocess the raw data to prepare it for analysis. This involves tasks such as handling missing values, removing duplicates, and transforming the data into a suitable format. In our case, we used Pandas to pivot the data and create a user-item matrix.
3. **Cleaned Dataset**: After preprocessing, we obtain a cleaned dataset where the data is free from inconsistencies and ready for analysis. This dataset typically has rows representing users, columns representing items, and the cells containing ratings or interactions.
4. **Feature Engineering :** This involves creating new features or transforming existing ones to improve the performance of machine learning models. In our case, features might include user-item interactions or latent factors generated by the NMF model.
5. **Features:** These are the variables or characteristics used by the machine learning model to make predictions. In the context of collaborative filtering, features might include user preferences, item attributes, or latent factors representing users and items in a lower-dimensional space.

```
RMSE: 0.2048
```

Flowchart of Neural Network Embedding based recommender system

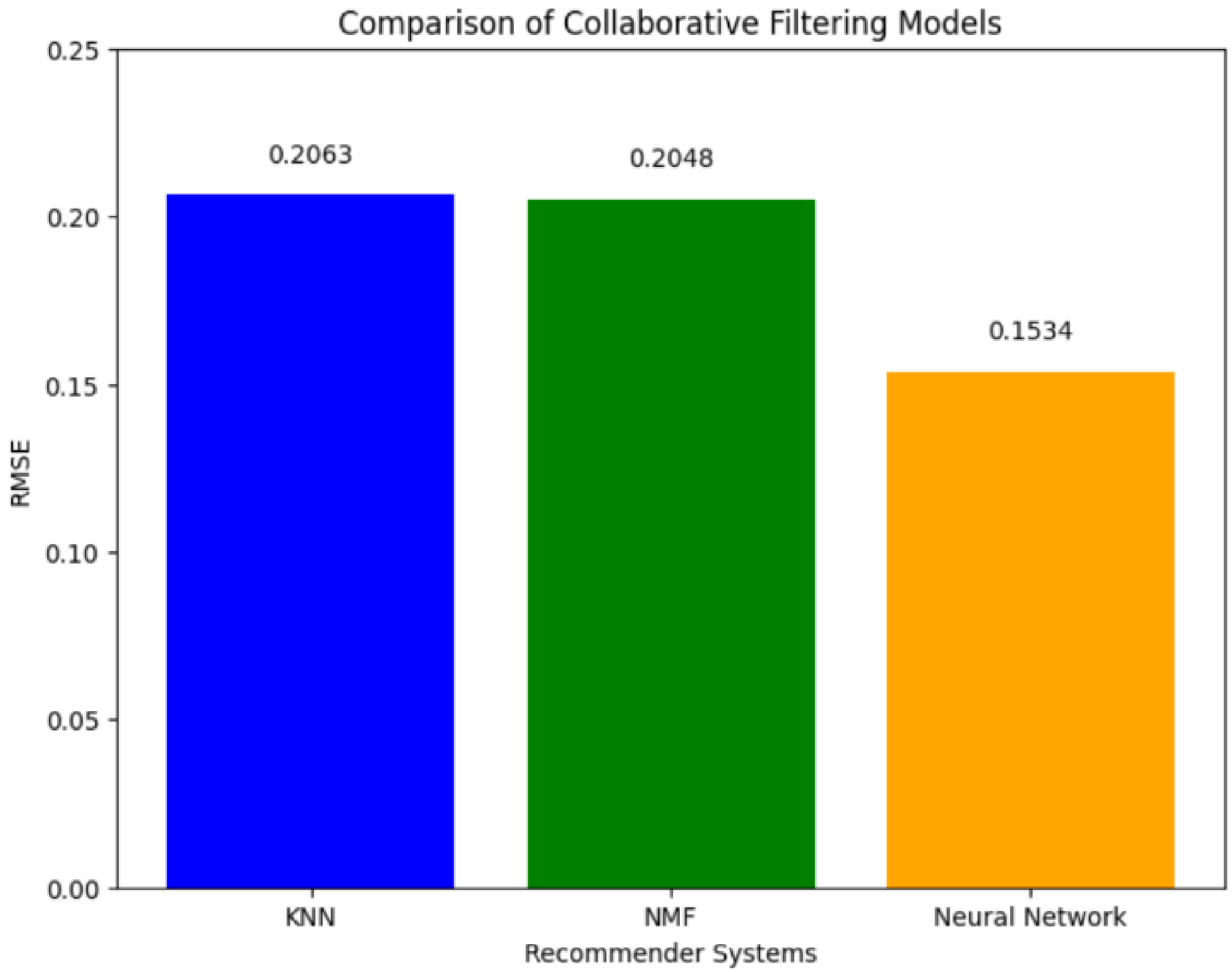


Let's break down the flowchart based on the conversation we had:

1. **Raw Data**: This is the initial data you have, in our case, it includes the course ratings data. Raw data is typically unprocessed and may contain noise, missing values, or inconsistencies.
2. **Data Processing**: In this step, we preprocess the raw data to prepare it for analysis. This involves tasks such as handling missing values, removing duplicates, and transforming the data into a suitable format. In our case, we used Pandas to pivot the data and create a user-item matrix.
3. **Cleaned Dataset**: After preprocessing, we obtain a cleaned dataset where the data is free from inconsistencies and ready for analysis. This dataset typically has rows representing users, columns representing items, and the cells containing ratings or interactions.
4. **Feature Engineering :** This involves creating new features or transforming existing ones to improve the performance of machine learning models. In our case, features might include user-item interactions or latent factors generated by the NMF model.
5. **Features:** These are the variables or characteristics used by the machine learning model to make predictions. In the context of collaborative filtering, features might include user preferences, item attributes, or latent factors representing users and items in a lower-dimensional space.

```
Test RMSE: 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.

Conclusions

- This project demonstrates the application of machine learning techniques in building effective recommender systems for online learning platforms. Through exploratory data analysis, we identified popular courses, user engagement patterns, and content similarities. We then developed and evaluated both content-based and collaborative-filtering models, including clustering, KNN, NMF, and neural embeddings.
- Key findings show that while content-based approaches effectively personalize recommendations based on course attributes, collaborative methods capture deeper user behavior patterns, offering improved accuracy. The comparative analysis confirms that hybridizing these methods holds promise for achieving balanced performance across precision, recall, and user satisfaction.
- Overall, the project highlights how recommender systems can reduce choice overload, enhance learner engagement, and ultimately support more effective online education. Future work may integrate real-time feedback loops and advanced deep learning models to further refine personalization.

Appendix

GitHub: