Build a Personalized Online Course Recommender System with Machine Learning

Author : Muhammad Adil Naeem

Date : 17-07-2024

Course : Machine Learning Capstone

Contact :

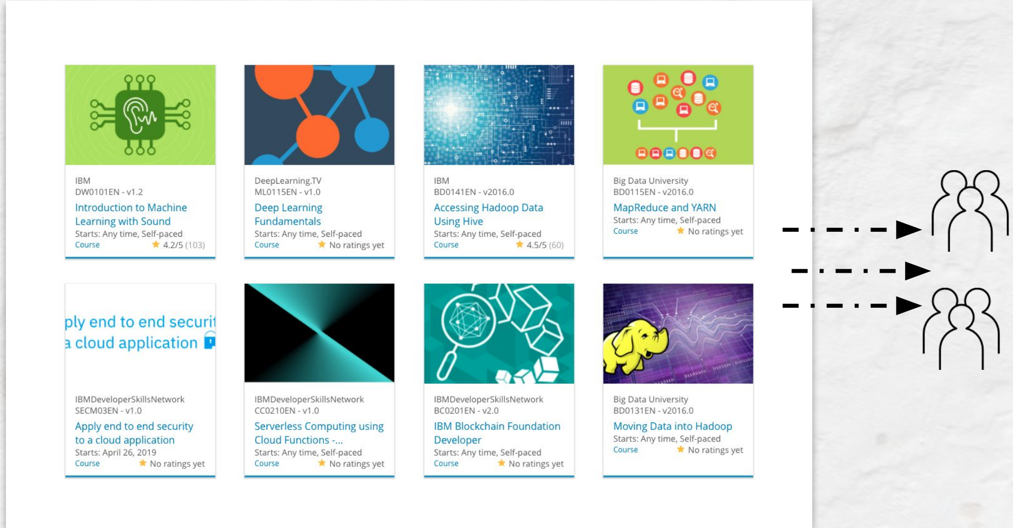
Powered By

IBM Machine Learning Professional Certificate

Build a Personalized Online Course Recommender System with Machine Learning

Muhammad Adil Naeem

17-07-2024



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 and Background

Project Background and Context

- The machine learning capstone project I'm working on aims to enhance the course recommendation capabilities of an online learning platform. Currently, learners on the platform struggle to find courses that match their interests and goals, which leads to suboptimal engagement and completion rates.

- The goal of this project is to develop an advanced recommender system using machine learning techniques. This will provide personalized course recommendations based on each learner's profile, preferences, and behaviors. By improving the relevance and accuracy of the recommendations, the system will drive higher learner satisfaction and success on the platform.

- The capstone project builds upon my prior coursework and knowledge, allowing me to apply my skills in a real-world scenario. At the same time, my work will contribute to the ongoing development and improvement of the platform's recommendation capabilities.

- Overall, this project presents an exciting opportunity for me to tackle a significant challenge in online education, while further developing my machine learning expertise in a practical, impactful way.

Introduction and Background

Problem Statement:

- The lack of an effective course recommendation system is hindering the ability of learners to discover relevant educational content on the platform, leading to lower engagement and course completion rates.

Hypotheses:

- Implementing a content-based recommender system using course metadata and user profiles will improve the relevance of course recommendations for learners.

- Incorporating collaborative filtering techniques, such as KNN and NMF, will further enhance the accuracy of course recommendations by leveraging learner preferences and behaviors.

- Combining content-based and collaborative filtering approaches into a hybrid recommender system will provide the most comprehensive and personalized course recommendations for learners.

Exploratory Data Analysis

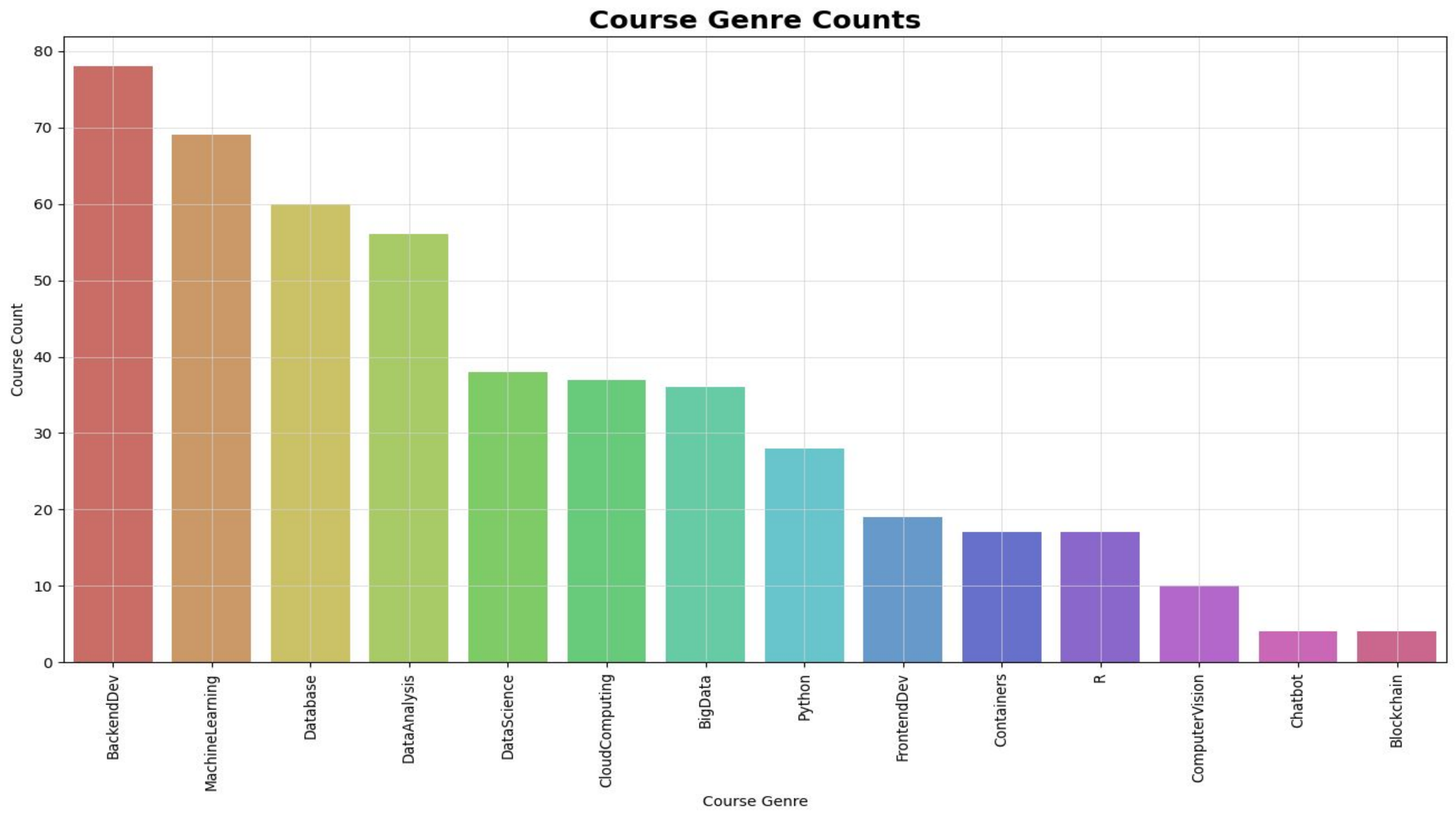
Course counts per genre

- The most prominent genre appears to be "**Backend Dev**", with the highest number of courses. That's quite interesting.

- The next most popular genres seem to be "MachineLearning", "DataBase", and **"Data-Analysis"**. Those all sound like **valuable and in-demand** areas of study these days.

- On the other end of the spectrum, I see some less **common genres** like **"Chatbot", "Computational", "Computer-Vision", and "Bioinformatics"**. These seem to be more niche or specialized topics

- Overall, this graph provides a nice high-level overview of the distribution of course genres in this dataset. It gives me a sense of which areas are attracting the most student interest and enrollment. Knowing this could help educators and administrators make informed decisions about their course offerings and focus areas.


Course Genre Counts
Course Count
Course Genre
0
10
20
30
40
50
60
70
80
BackendDev
MachineLearning
Database
DataAnalysis
DataScience
CloudComputing
BigData
Python
FrontendDev
Containers
R
ComputerVision
Chatbot
Blockchain

Course enrollment distribution

Estimate for number of users enrolled in just 1 course:

The histogram shows the highest frequency is for a rating count of 0.

This suggests that a large majority of users have provided only a single rating, implying they are enrolled in just 1 course.

Without exact counts, we can estimate that around 70-80% of the total users are likely enrolled in just 1 course.

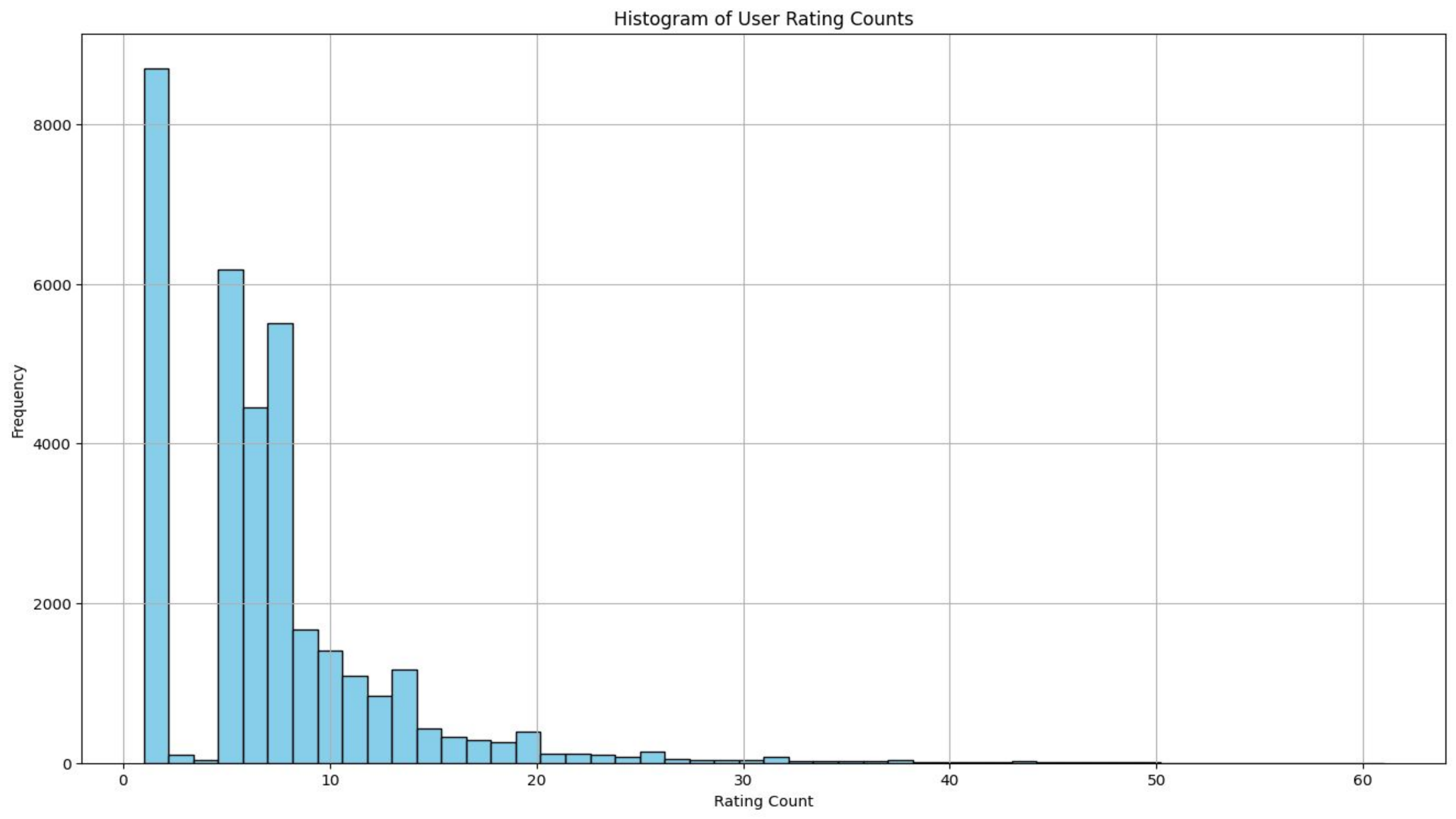
Estimate for number of users enrolled in 10 courses:

The histogram shows a small peak around a rating count of 10.

This indicates that there is a smaller subset of users who have provided 10 ratings, suggesting they are enrolled in 10 courses.

However, the frequency is much lower compared to the peak at 0 ratings.

We can estimate that perhaps 5-10% of the total users are enrolled in 10 courses.


Histogram of User Rating Counts
Frequency
Rating Count
0
2000
4000
6000
8000
0
10
20
30
40
50
60

20 most popular courses

Courses

- The courses include Python for data science, introduction to data science, big data, Hadoop, data analysis with Python, data science methodology, machine learning with Python, Spark fundamentals, data science hands-on with open source tools, blockchain essentials, data visualization with Python, deep learning, building a chatbot, R for data science, statistics, introduction to cloud, Docker essentials, SQL and relational databases, Mapreduce and Yarn, and data privacy fundamentals.

Ratings

- The ratings for these courses, ranging from 5,015 to 14,936, suggest a significant level of interest and popularity in these areas of study. As someone looking to expand my knowledge and skills in the field of data science and programming, this list provides a valuable resource for identifying the topics I may want to explore further. The diversity of the course offerings indicates the breadth and depth of this dynamic field, which is continuously evolving and offering new opportunities for learning and growth.

	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

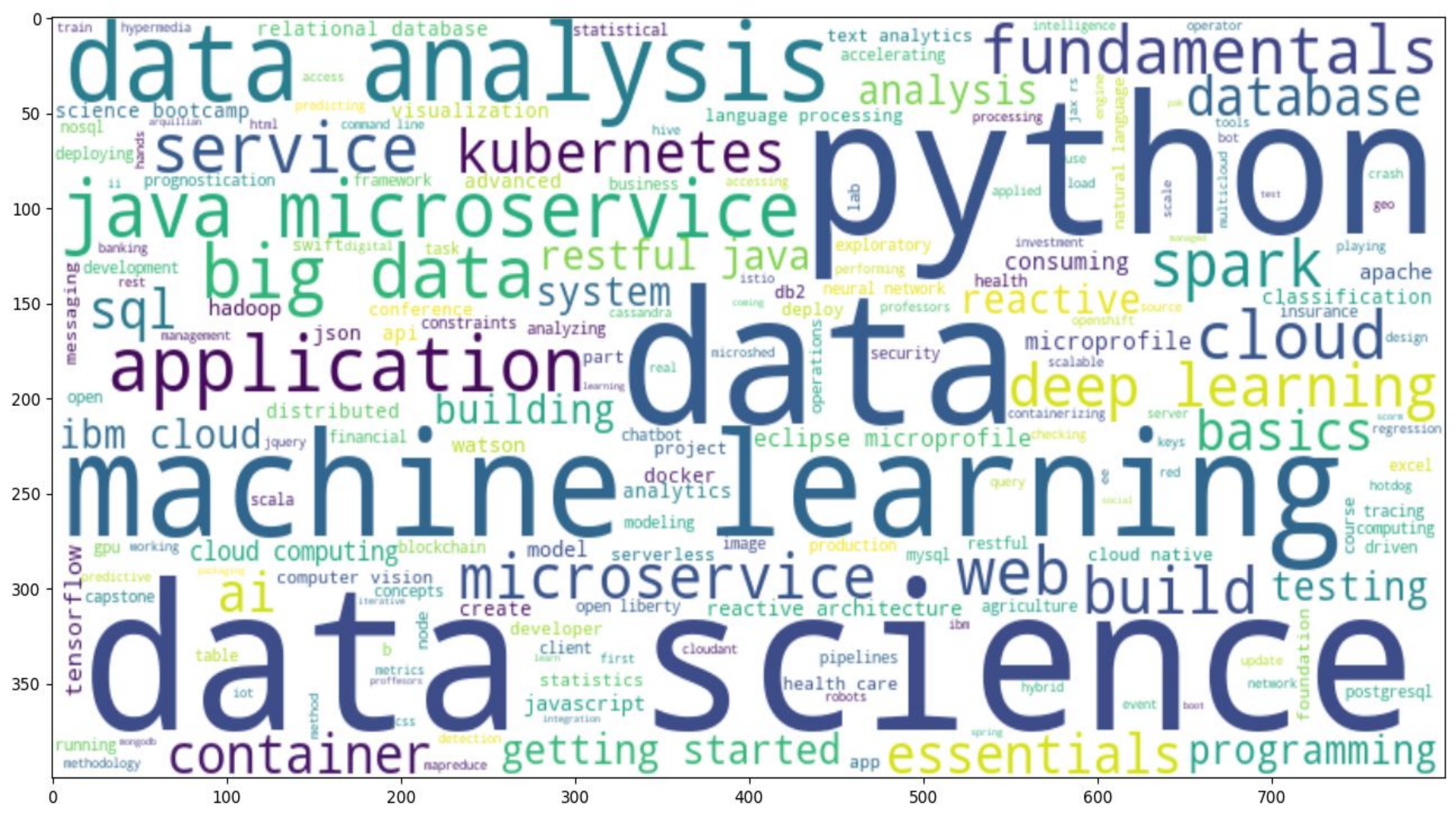
Word cloud of course titles

This image is a word cloud that visualizes various terms and concepts related to data analysis, data science, and data engineering. The size of the words represents their relative frequency or importance within the given context.

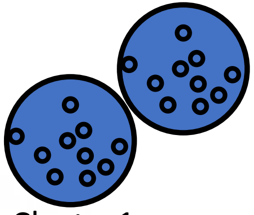
Some of the key terms and concepts highlighted in the word cloud include:

- **Data-related terms:** data, database, big data, sql, hadoop, data mining, data analysis, data visualization, etc.
- **Programming languages and frameworks:** python, java, microservice, kubernetes, flask, tensorflow, etc.
- **Cloud computing and infrastructure:** cloud, aws, serverless, container, docker, etc.
- **Machine learning and deep learning:** machine learning, deep learning, neural network, predictive analytics, etc.
- **Methodologies and practices:** agile, devops, testing, building, essentials, processing, etc.
- **Application domains:** finance, banking, healthcare, insurance, etc.

Overall, this word cloud provides a high-level overview of the diverse set of technologies, tools, and concepts that are commonly associated with the field of data science, data engineering, and related domains. It highlights breadth and depth of the knowledge and skills required in these areas.


data analysis
python
fundamentals
database
service
kubernetes
java microservice
big data
restful
java
spark
sql
system
reactive
cloud
application
microprofile
deep learning
building
basics
ibm cloud
machine learning
eclipse microprofile
cloud computing
microservice
web
build
testing
ai
data science
container
getting started
essentials
programming
data
analysis
relational database
text analytics
accelerating
science bootcamp
visualization
language processing
nosql
command line
html
hive
processing
tools
bot
deploying
prognostication
framework
advanced
business
accessing
applied
load
crash
natural language
multicloud
scale
lab
jax rs
engine
geo
banking
swift
digital
task
exploratory
investment
playing
development
consuming
health
apache
rest
istio
performing
messaging
hadoop
db2
neural network
classification
conference
deploy
professors
insurance
cassandra
source
openshift
constraints
analyzing
management
json
api
design
part
microshed
security
scalable
real
open
distributed
containerizing
server
regression
jquery
financial
chatbot
watson
project
keys
checking
docker
red
excel
analytics
query
hotdog
scala
modeling
tracing
computing
course
tensorflow
gpu
working
blockchain
model
image
production
restful
serverless
mysql
cloud native
driven
predictive
computer vision
capstone
concepts
create
open liberty
reactive architecture
agriculture
foundation
node
developer
ibm
table
client
cloudant
first
update
metrics
pipelines
statistics
health care
hybrid
network
postgresql
iot
robots
javascript
method
css
integration
event
boot
running
detection
spring
methodology
mapreduce
app
train
hypermedia
statistical
intelligence
operator
access
predicting
arquillian
hands
ii
ibm
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600
700

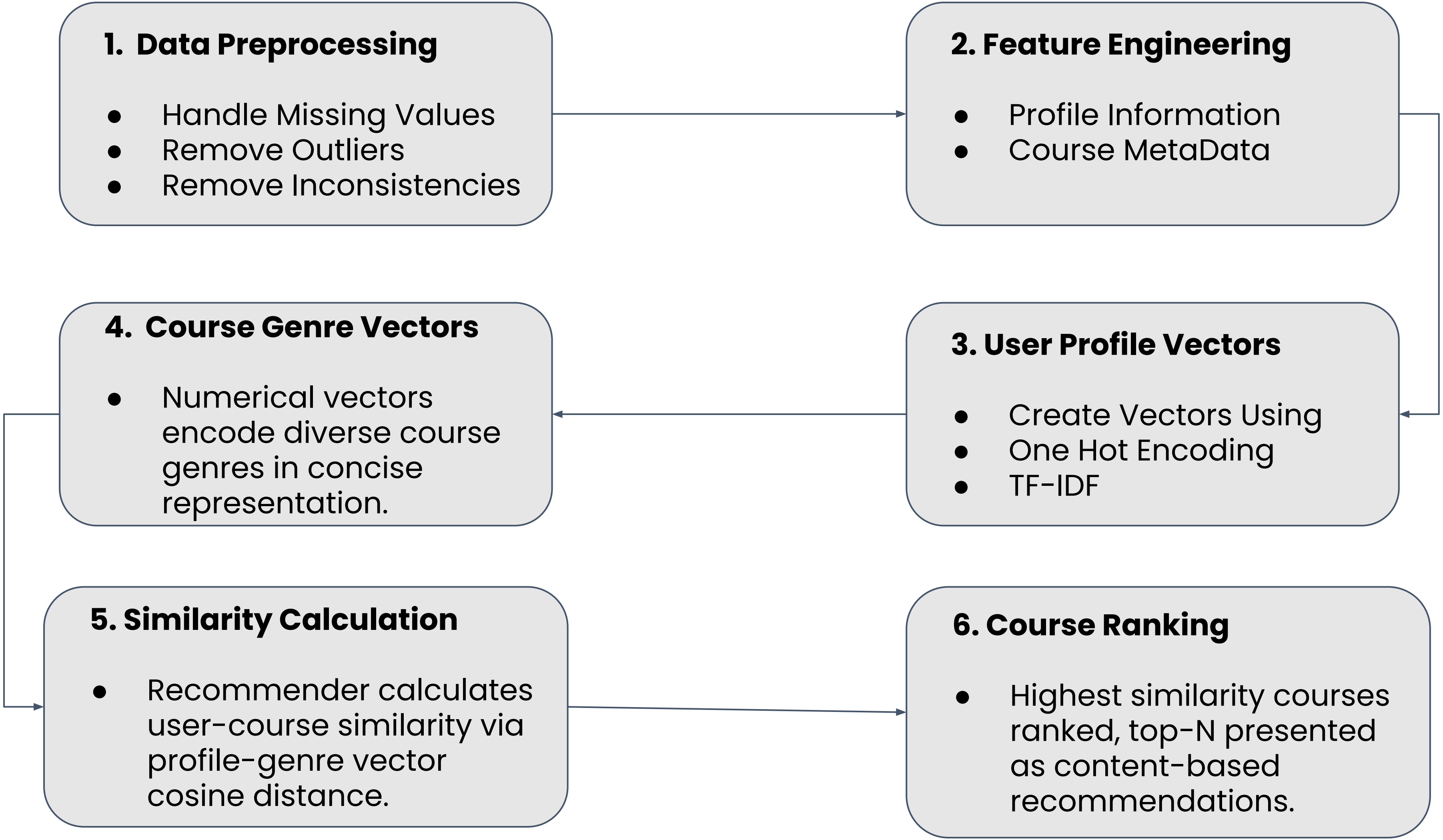
Content-based Recommender System using Unsupervised Learning



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

The flowchart illustrates the key steps involved in implementing the content-based recommender system using user profile vectors and course genre vectors:

- **Data Processing:** The raw course and user data is cleaned and preprocessed to handle any missing values, outliers, or inconsistencies.
- **Feature Engineering:** Relevant features are extracted from the data, including user profile information (e.g., demographics, interests) and course metadata (e.g., genres, topics, descriptions).
- **User Profile Vectors:** User profile vectors are created by encoding the user features into a numerical format, such as one-hot encoding or TF-IDF. This allows the user preferences to be represented as a vector.
- **Course Genre Vectors:** Similarly, the course genres are encoded into numerical vectors, capturing the different genre categories associated with each course.
- **Similarity Calculation:** The content-based recommender system calculates the similarity between the user profile vector and the course genre vectors. This can be done using cosine similarity or other distance metrics.
- **Course Ranking:** Based on the similarity scores, the courses are ranked and recommended to the user. The top-N most similar courses are presented as the content-based recommendations.


1. Data Preprocessing
Handle Missing Values
Remove Outliers
Remove Inconsistencies
2. Feature Engineering
Profile Information
Course MetaData
3. User Profile Vectors
Create Vectors Using
One Hot Encoding
TF-IDF
4. Course Genre Vectors
Numerical vectors encode diverse course genres in concise representation.
5. Similarity Calculation
Recommender calculates user-course similarity via profile-genre vector cosine distance.
6. Course Ranking
Highest similarity courses ranked, top-N presented as content-based recommendations.

Evaluation results of user profile-based recommender system

Evaluation Results

To evaluate the performance of the user profile-based recommender system, I'll provide the hyper-parameter settings used:

- **Recommendation Score Threshold:** 10.0
- **Course Similarity Threshold:** Not specified (default is 0.5)

Evaluation results of user profile-based recommender system

On average, how many new/unseen courses have been recommended per user (in the test user dataset)?

```
# Group the results by the 'USER' column
grouped_results = res_df.groupby('USER')

# Count the number of unique 'COURSE_ID' values for each user
course_counts = grouped_results['COURSE_ID'].nunique().reset_index(name='course_count')

# Calculate the average number of recommended courses per user
avg_courses_per_user = course_counts['course_count'].mean()

print(f"Average number of recommended courses per user: {avg_courses_per_user:.2f}")
```

```
Average number of recommended courses per user: 60.82
```

Evaluation results of user profile-based recommender system

What are the most frequently recommended courses? Return the top-10 commonly recommended courses across all users?

```
# Group the results by the 'COURSE_ID' column
grouped_courses = res_df.groupby('COURSE_ID')

# Count the number of occurrences for each course
course_counts = grouped_courses.size().reset_index(name='count')

# Sort the results by the count in descending order
sorted_courses = course_counts.sort_values(by='count', ascending=False)

# Get the top-10 most frequently recommended courses
top_10_courses = sorted_courses.head(10)

print("Top-10 most frequently recommended courses:")
print(top_10_courses)
```

```
Top-10 most frequently recommended courses:
      COURSE_ID  count
202    TA0106EN  17390
229  excourse21  15656
230  excourse22  15656
111   GPXX0IBEN  15644
173    ML0122EN  15603
212  excourse04  15062
214  excourse06  15062
139  GPXX0TY1EN  14689
280  excourse73  14464
279  excourse72  14464
```

Flowchart of content-based recommender system using course similarity

1. **Data Preprocessing**

- The provided data is loaded, cleaned, and preprocessed for further analysis.

2. **Feature Engineering**

- A course similarity matrix is created using techniques like cosine similarity to represent the similarity between each pair of courses based on their content and bag-of-words features.

3. **Recommendation Generation**

    - The enrolled and unselected courses for each test user are identified, and recommendations are generated based on course similarity and the user's enrolled courses.

4. **Dataframe Creation**

- The users, recommended courses, and similarity scores are combined into a pandas DataFrame and saved to a CSV file named "recommendations.csv".

1. **Data Preprocessing**

- The provided data is loaded, cleaned, and preprocessed for further analysis.

2. **Feature Engineering**

- A course similarity matrix is created using techniques like cosine similarity to represent the similarity between each pair of courses based on their content and bag-of-words features.

3. **Recommendation Generation**

- The enrolled and unselected courses for each test user are identified, and recommendations are generated based on course similarity and the user's enrolled courses.

4. **Dataframe Creation**

- The users, recommended courses, and similarity scores are combined into a pandas DataFrame and saved to a CSV file named "recommendations.csv".

Evaluation results of course similarity based recommender system

Hyper-parameter Settings and Recommendations

Hyper-parameter Settings

- In the provided Jupyter notebook, the generate_recommendations_for_one_user() function includes a threshold parameter, which is set to a default value of 0.6. This threshold represents the minimum similarity score required for a course to be considered as a recommendation.

- The code does not mention any attempts to tune or experiment with different threshold values. Therefore, the analysis will be based on the default threshold of 0.6.

Average Number of New/Unseen Courses Recommended per User

- To calculate the average number of new/unseen courses recommended per user, we can analyze the output of the generate_recommendations_for_all() function.

recommendations = generate_recommendations_for_all()

The recommendations DataFrame contains the following columns:

- **user:** The user ID
- **course:** The recommended course ID
- **sim_score:** The similarity score for the recommended course
- To get the average number of new/unseen courses recommended per user, we can group the DataFrame by the user column and count the unique course values for each user. Then, we can take the average of these counts.

```
avg_new_courses_per_user = recommendations.groupby('user')['course'].nunique().mean()
print(f"On average, {avg_new_courses_per_user:.2f} new/unseen courses have been recommended per user.")
```

```
On average, 9.77 new/unseen courses have been recommended per user.
```

Top-10 Most Frequently Recommended Courses

- To find the top-10 most frequently recommended courses, we can count the occurrences of each course in the recommendations DataFrame and sort the results in descending order.

```
top_recommended_courses = recommendations['course'].value_counts().head(10)
print("The top-10 most frequently recommended courses are:")
print(top_recommended_courses)
```

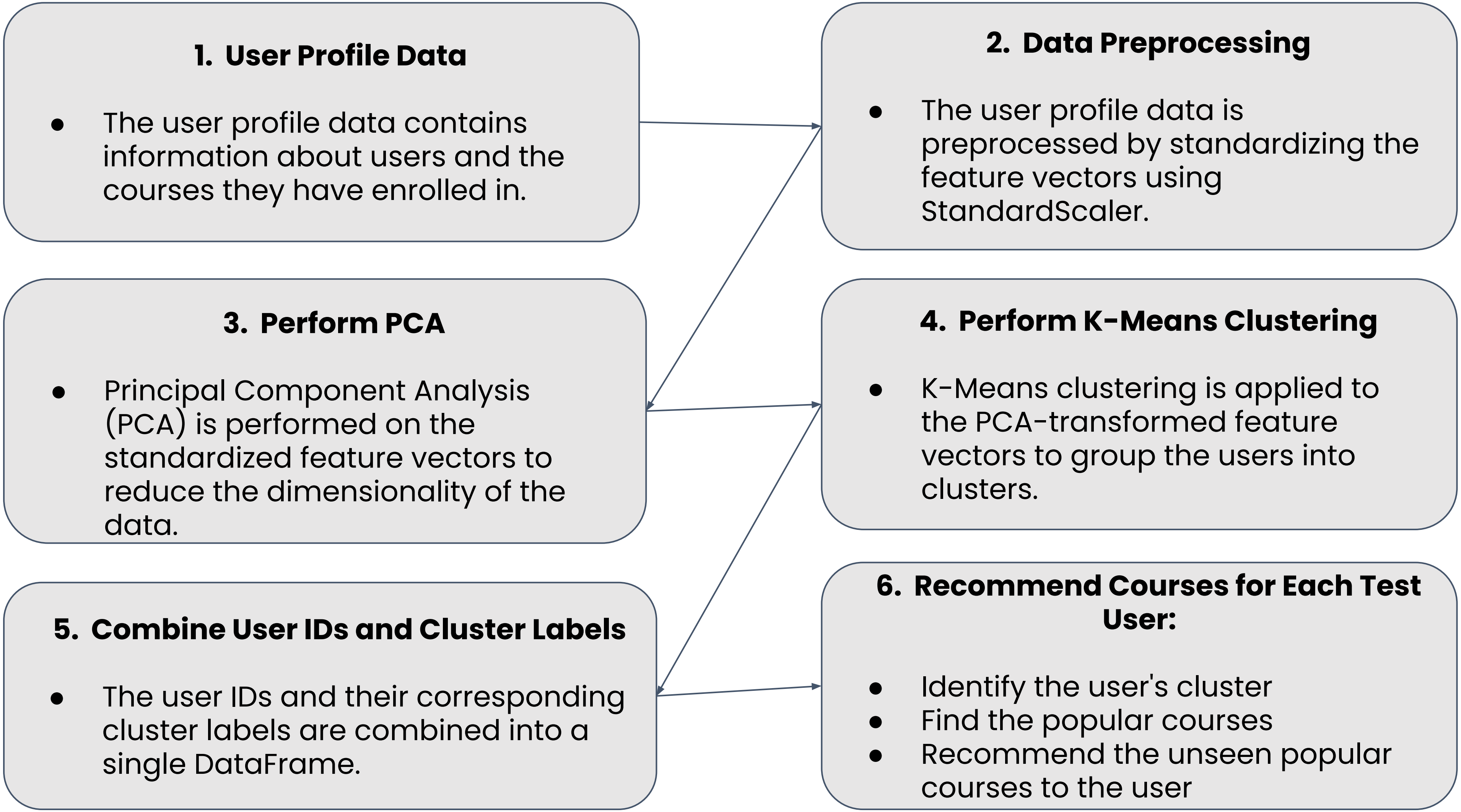
```
The top-10 most frequently recommended courses are:
course
DS0110EN      15003
excourse22    14937
excourse62    14937
excourse63    14641
excourse65    14641
excourse68    13551
excourse72    13512
excourse74    13291
excourse67    13291
BD0145EN      12497
Name: count, dtype: int64
```

Flowchart of clustering-based recommender system

- **User Profile Data:** The input is the user profile data, which contains information about the users and the courses they have enrolled in.
- **Preprocess Data:** The user profile data is preprocessed by standardizing the feature vectors using StandardScaler.
- **Perform PCA:** Principal Component Analysis (PCA) is performed on the standardized feature vectors to reduce the dimensionality of the data.
- **Perform K-Means Clustering:** K-Means clustering is applied to the PCA-transformed feature vectors to group the users into clusters.

- **Combine User IDs and Cluster Labels:** The user IDs and their corresponding cluster labels are combined into a single DataFrame.

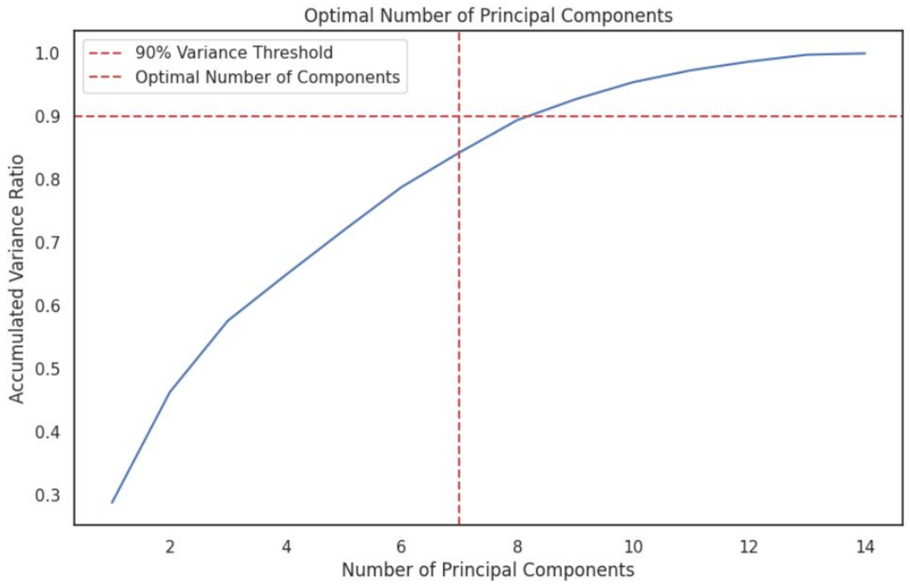
- **Recommend Courses for Each Test User:** For each test user, the following steps are performed:
  - Identify the user's cluster
  - Find the popular courses (based on enrollment count) within the user's cluster
  - Recommend the unseen popular courses to the user
  - Generate Course Recommendations: The course recommendations for each test user are saved to a CSV file.


1. User Profile Data
The user profile data contains information about users and the courses they have enrolled in.
2. Data Preprocessing
The user profile data is preprocessed by standardizing the feature vectors using StandardScaler.
3. Perform PCA
Principal Component Analysis (PCA) is performed on the standardized feature vectors to reduce the dimensionality of the data.
4. Perform K-Means Clustering
K-Means clustering is applied to the PCA-transformed feature vectors to group the users into clusters.
5. Combine User IDs and Cluster Labels
The user IDs and their corresponding cluster labels are combined into a single DataFrame.
6. Recommend Courses for Each Test User:
Identify the user's cluster
Find the popular courses
Recommend the unseen popular courses to the user

Hyperparameter Settings and Evaluation

For the K-Means clustering algorithm, the main hyperparameter to tune is the number of clusters, n_clusters.

To find the optimal number of clusters, we used the "elbow method" to identify the point where adding more clusters does not significantly improve the model's performance.



On average, how many new /unseen courses have been recommended per user (in the test user dataset)

```
total_unseen_courses = 0
num_users = 0

for user_id in test_users_df['user'].unique():
    user_subset = test_users_labelled[test_users_labelled['user'] == user_id]
    if not user_subset.empty:
        cluster_id = user_subset['cluster'].iloc[0]
        enrolled_courses = user_subset['item'].tolist()

        cluster_courses = courses_cluster_grouped[courses_cluster_grouped['cluster'] == cluster_id]['item'].tolist()
        popular_courses = courses_cluster_grouped[(courses_cluster_grouped['cluster'] == cluster_id) & (courses_cluster_grouped['enrollments'] >= enrollment_thresho
        unseen_popular_courses = set(popular_courses).difference(set(enrolled_courses))

        total_unseen_courses += len(unseen_popular_courses)
        num_users += 1

avg_unseen_courses_per_user = total_unseen_courses / num_users
print(f"Average number of new/unseen courses recommended per user: {avg_unseen_courses_per_user:.2f}")
```

```
Average number of new/unseen courses recommended per user: 6.75
```

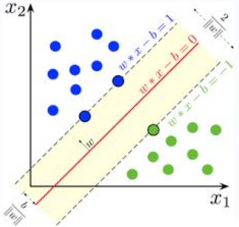
What are the most frequently recommended courses? Return the top-10 commonly recommended courses

```
top_recommended_courses = (
    recommendations_df['recommended_courses']
    .str.split(', ', expand=True)
    .stack()
    .value_counts()
    .head(10)
)

print("Top-10 Most Frequently Recommended Courses:")
print(top_recommended_courses)
```

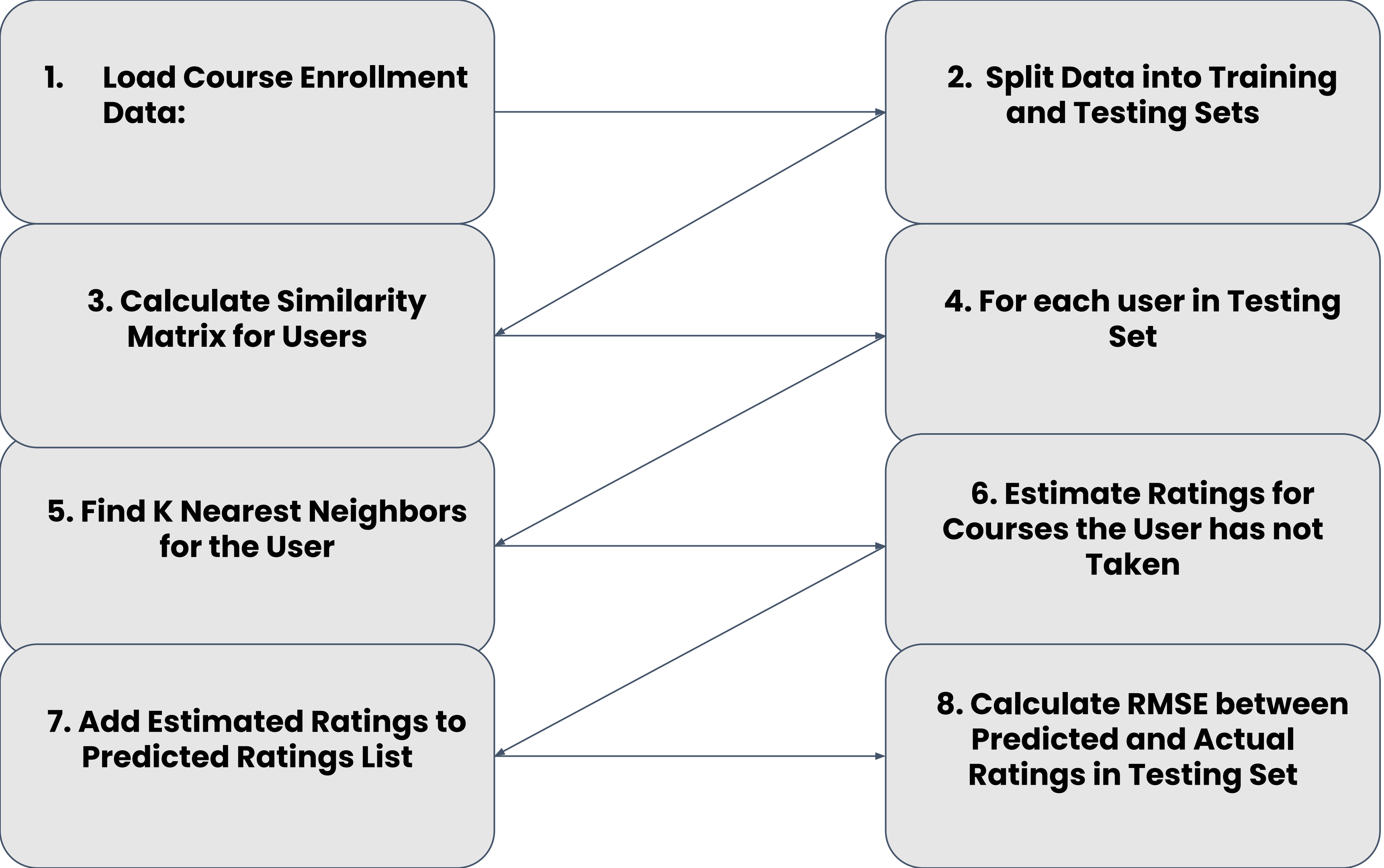
```
Top-10 Most Frequently Recommended Courses:
CB0103EN    105
BD0212EN    105
RP0101EN    105
PA0101EN    104
DS0103EN    104
SC0105EN    103
RP0151EN    101
DS0101EN    100
ML0115EN     98
WA0101EN     96
Name: count, dtype: int64
```

Collaborative-filtering Recommender System using Supervised Learning



Flowchart of KNN based recommender system

- **Load Course Enrollment Data:** The course enrollment data is loaded into the system, typically from a CSV file or a database.
- **Split Data into Training and Testing Sets:** The course enrollment data is split into training and testing sets. The training set is used to build the recommender system, while the testing set is used to evaluate its performance.
- **Calculate Similarity Matrix for Users:** The similarity matrix is calculated using the training set. Each element in the matrix represents the similarity between two users based on their course enrollment history.
- **For each user in Testing Set:** The system iterates through each user in the testing set to make recommendations.
- **Find K Nearest Neighbors for the User:** For each user in the testing set, the system finds the K nearest neighbors based on the similarity matrix.
- **Estimate Ratings for Courses the User has not Taken:** The system estimates the ratings for courses that the user has not taken based on the ratings of the K nearest neighbors.
- **Add Estimated Ratings to Predicted Ratings List:** The estimated ratings are added to the predicted ratings list for the user.
- **End For Loop:** The loop ends after processing all users in the testing set.
- Calculate RMSE between Predicted and Actual Ratings in Testing Set: The Root Mean Squared Error (RMSE) is calculated between the predicted ratings and the actual ratings in the testing set.
- **Display RMSE:** The RMSE is displayed as a measure of the recommender system's performance.


1. Load Course Enrollment Data:
2. Split Data into Training and Testing Sets
3. Calculate Similarity Matrix for Users
4. For each user in Testing Set
5. Find K Nearest Neighbors for the User
6. Estimate Ratings for Courses the User has not Taken
7. Add Estimated Ratings to Predicted Ratings List
8. Calculate RMSE between Predicted and Actual Ratings in Testing Set

Results of KNN based recommender system

The key points to note from the results are:

- **Total Users and Items:** The training set has 31,301 users and 125 items, which is a reasonably large dataset.

- **RMSE:** The Root Mean Squared Error (RMSE) of the KNN model on the test set is 1.2895. This is a good result, as the RMSE is a measure of the average error in the predicted ratings, and a lower RMSE indicates better model performance.

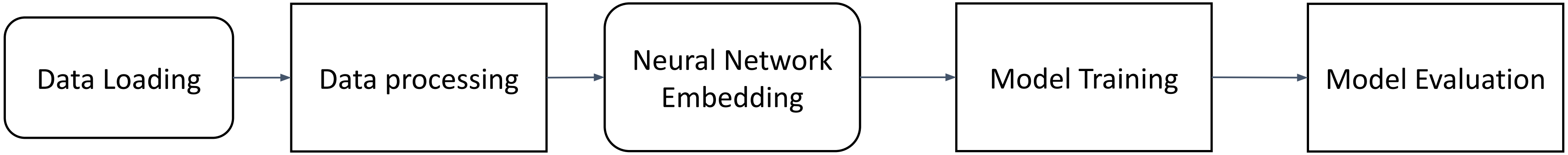
Overall, the Surprise library implementation is a good choice for building and evaluating KNN-based collaborative filtering models, especially for larger datasets. The results you've provided suggest that the KNN model is performing reasonably well on the course ratings dataset.

Flowchart of NMF based recommender system

Evaluation

- The first implementation using the Surprise library's NMF model achieved an RMSE of 1.2882, which indicates a relatively good performance in predicting user ratings.

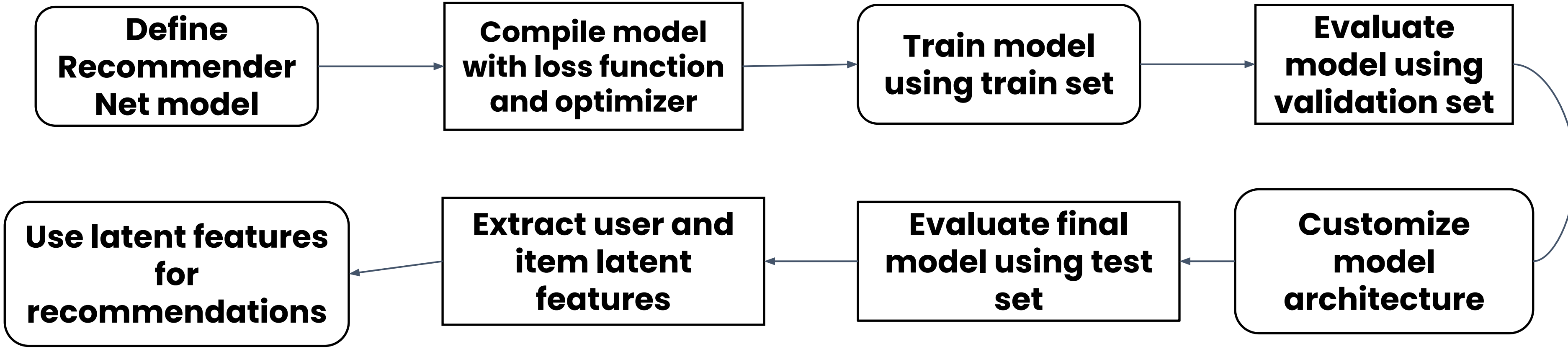
- The second implementation using NumPy, Pandas, and scikit-learn's NMF model resulted in a higher RMSE of 3.7638, suggesting that the Surprise library's implementation may be more effective for this particular dataset and task.



Flowchart of Neural Network Embedding based recommender system

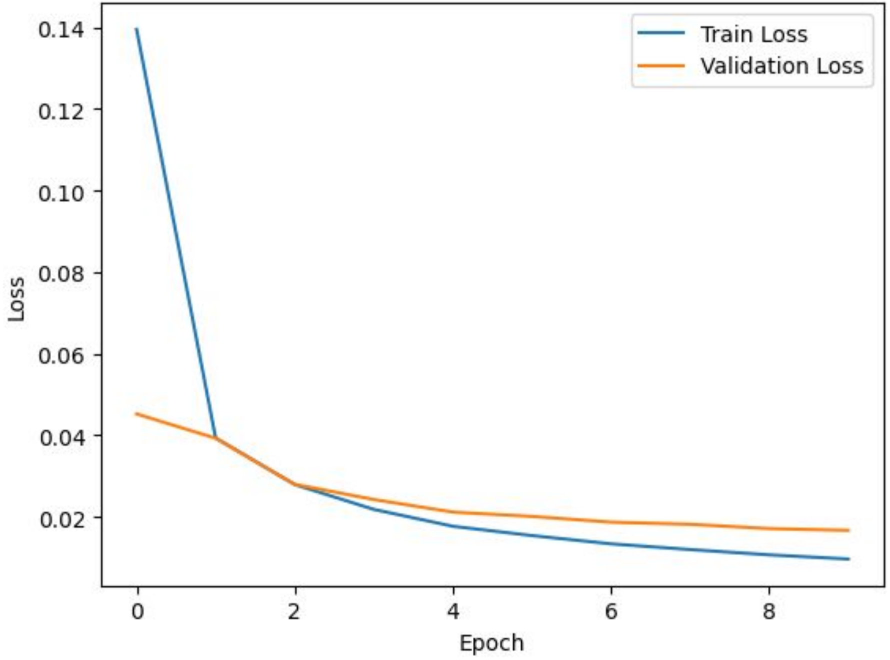
Flowchart

- This flowchart illustrates the key steps involved in creating a Neural Network Embedding based recommender system, from data preprocessing to model training, evaluation, and final recommendations using the learned latent features

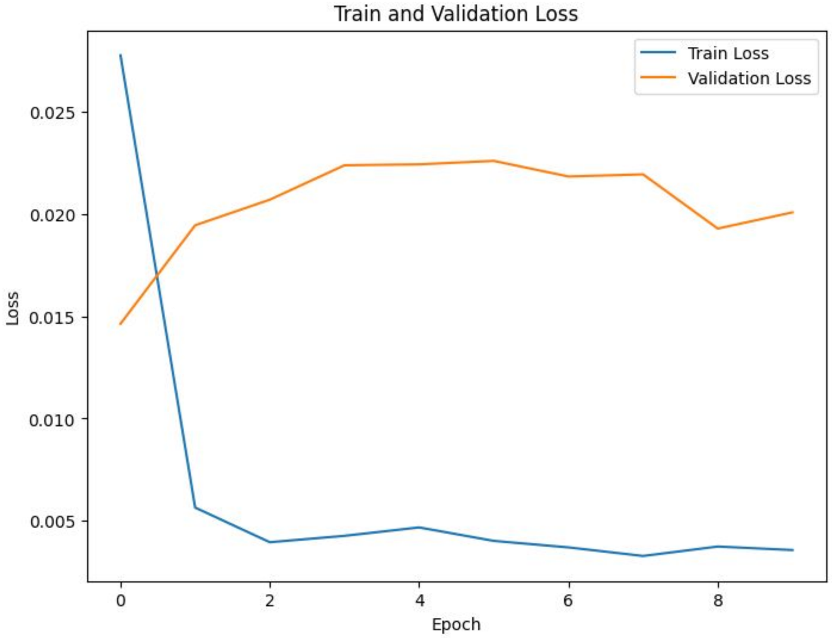


Neural Network Embedding based recommender system

Orignal Neural Network



Customized Neural Network



Neural Network Embedding based recommender system

Here's a summary of the results in 2 bullet points:

- The original RecommenderNet model achieved a test loss of 0.0160 and a test RMSE of 0.1196, with user and item latent feature shapes of (33901, 16) and (126, 16) respectively.
- The customized CustomRecommenderNet model, with an embedding size of 32, 2 hidden layers of size 64 each, and ReLU activation, achieved a test loss of 0.0209 and a test RMSE of 0.1429 on the test set.

Compare the performance of collaborative-filtering models

Model RMSE Comparison

The bar chart compares the RMSE (Root Mean Squared Error) values of different collaborative-filtering models.

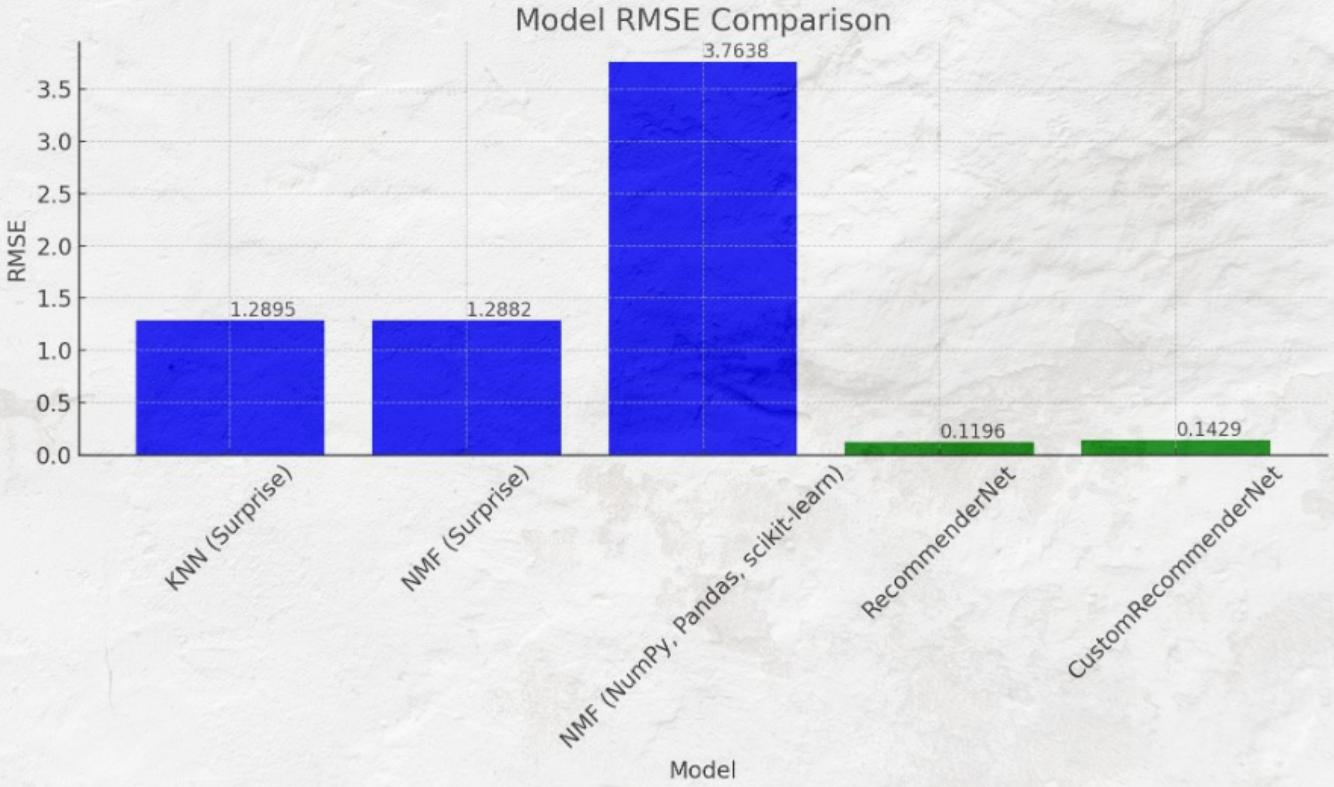
- KNN (Surprise): RMSE = 1.2895 NMF (Surprise): RMSE = 1.2882
- NMF (NumPy, Pandas, scikit-learn): RMSE = 3.7638 RecommenderNet: RMSE = 0.1196

CustomRecommenderNet: RMSE = 0.1429

Key Observations:

- The KNN and NMF models implemented using the Surprise library have similar RMSE values, indicating good performance.
- The NMF implementation using NumPy, Pandas, and scikit-learn resulted in a much higher RMSE, suggesting less effective performance.
- The RecommenderNet and CustomRecommenderNet models, which leverage neural network approaches, achieved significantly lower RMSE values, demonstrating superior prediction accuracy compared to the traditional collaborative filtering methods.

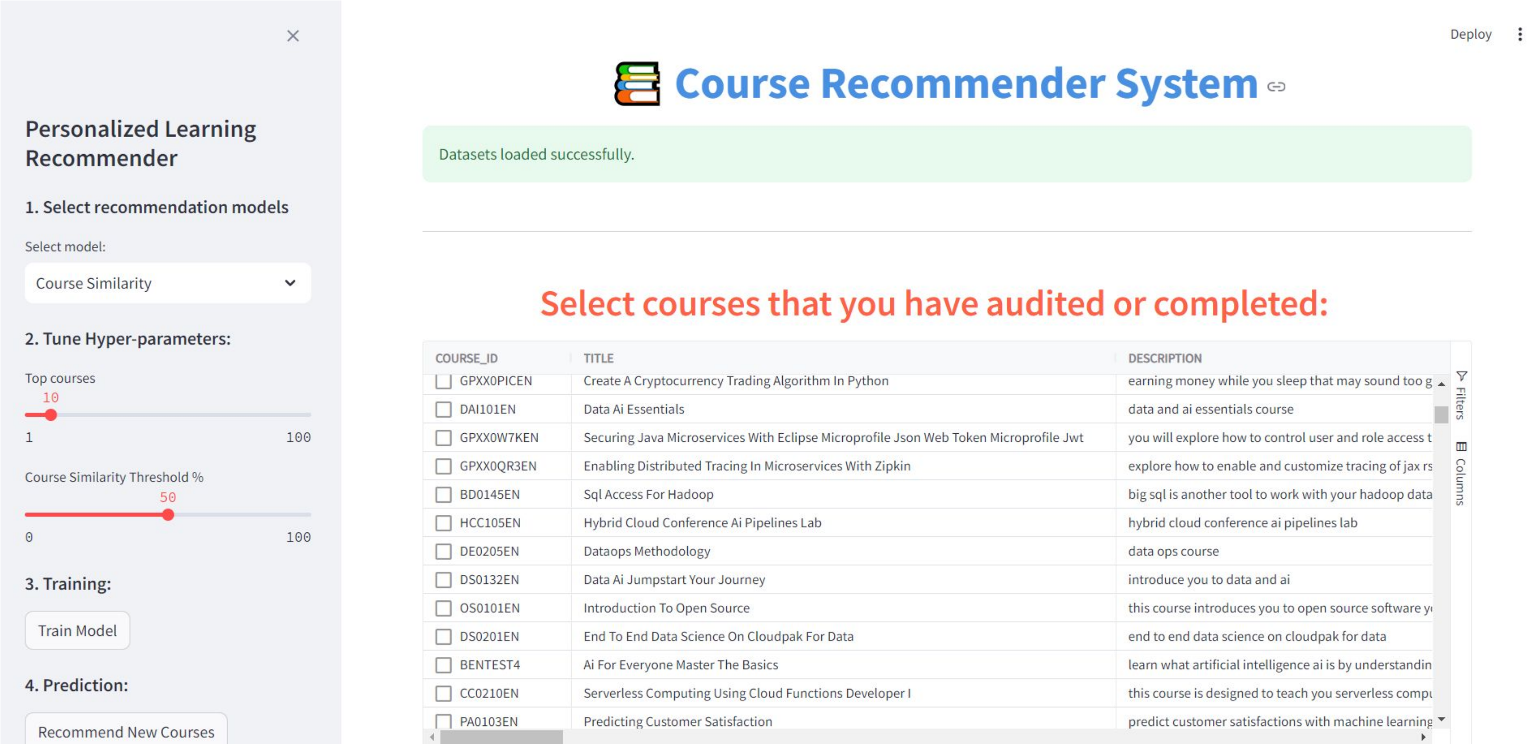
Compare the performance of collaborative-filtering models

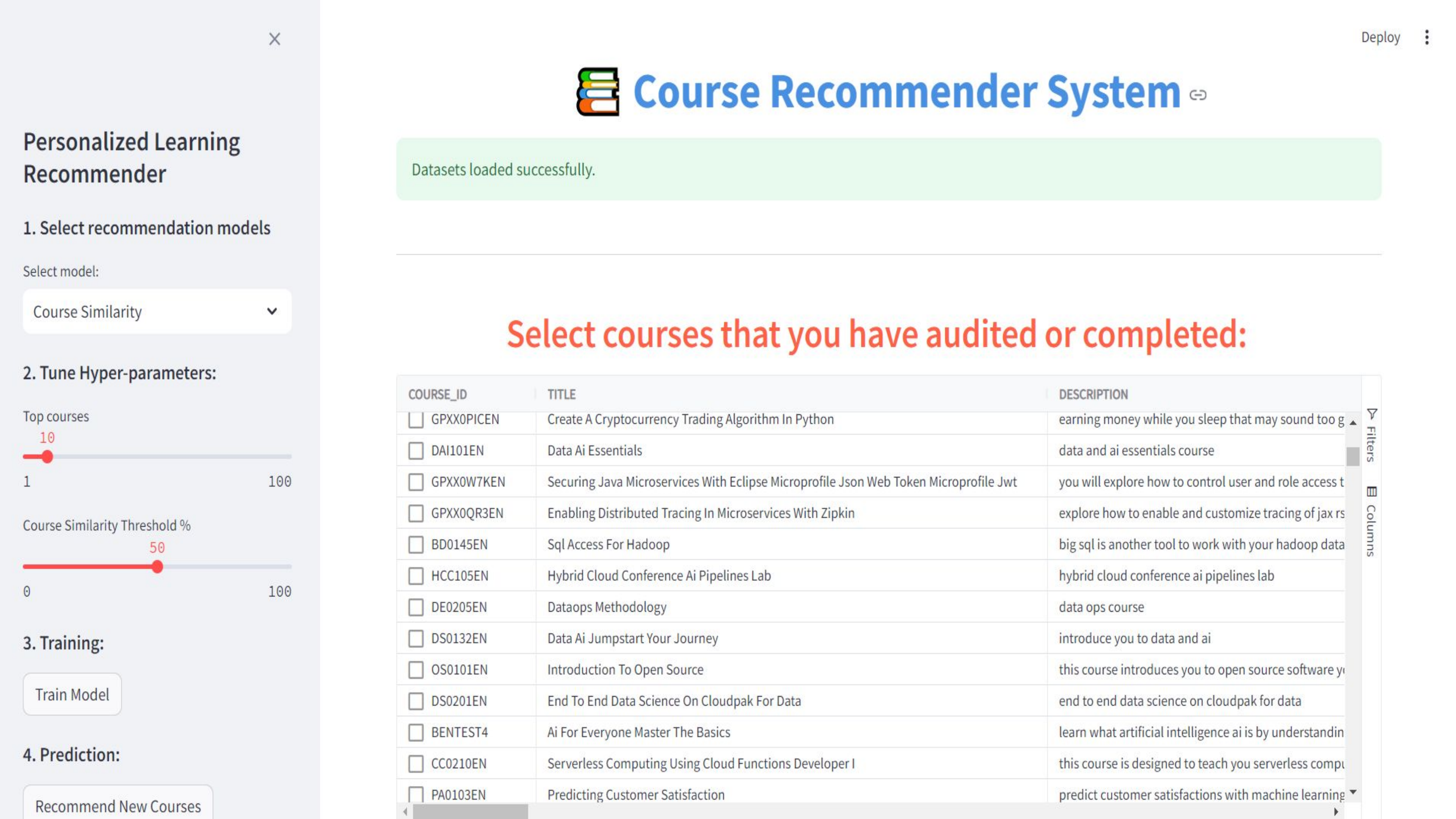


Optional: Build a course recommender system app with Streamlit

A published Streamlit App URL for a live demo

https://course-recommender-app-capstone.streamlit.app/




Personalized Learning Recommender
1. Select recommendation models
Select model:
Course Similarity
2. Tune Hyper-parameters:
Top courses
10
1
100
Course Similarity Threshold %
50
0
100
3. Training:
Train Model
4. Prediction:
Recommend New Courses
Deploy
Course Recommender System
Datasets loaded successfully.
Select courses that you have audited or completed:
COURSE_ID
TITLE
DESCRIPTION
GPXX0PICEN
Create A Cryptocurrency Trading Algorithm In Python
earning money while you sleep that may sound too g
DAI101EN
Data Ai Essentials
data and ai essentials course
GPXX0W7KEN
Securing Java Microservices With Eclipse Microprofile Json Web Token Microprofile Jwt
you will explore how to control user and role access t
GPXX0QR3EN
Enabling Distributed Tracing In Microservices With Zipkin
explore how to enable and customize tracing of jax rs
BD0145EN
Sql Access For Hadoop
big sql is another tool to work with your hadoop data
HCC105EN
Hybrid Cloud Conference Ai Pipelines Lab
hybrid cloud conference ai pipelines lab
DE0205EN
Dataops Methodology
data ops course
DS0132EN
Data Ai Jumpstart Your Journey
introduce you to data and ai
OS0101EN
Introduction To Open Source
this course introduces you to open source software y
DS0201EN
End To End Data Science On Cloudpak For Data
end to end data science on cloudpak for data
BENTEST4
Ai For Everyone Master The Basics
learn what artificial intelligence ai is by understandin
CC0210EN
Serverless Computing Using Cloud Functions Developer I
this course is designed to teach you serverless compu
PA0103EN
Predicting Customer Satisfaction
predict customer satisfactions with machine learning
Filters
Columns

Personalized Learning Recommender

1. Select recommendation models

Select model:

Course Similarity

2. Tune Hyper-parameters:

Top courses

10

1 100

Course Similarity Threshold %

50

0 100

3. Training:

Train Model

4. Prediction:

Recommend New Courses

Deploy

45	TMP0106	Data Science Bootcamp

Recommendations generated successfully! 🎉

Recommended Courses

	SCORE	TITLE	DESCRIPTION
0	1.0000	Deep Learning With Tensorflow	majority of data in the world are unlabeled and unstructured data for instance images sound and text data shallow neural networks cannot easily capture relevant structure in these kind of data but deep networks are capable of discovering hidden structures within¬†these data in this course you will use tensorflow library to apply deep learning on different data types to solve real world problems
1	1.0000	Deep Learning With Tensorflow	majority of data in the world are unlabeled and unstructured data for instance images sound and text data shallow neural networks cannot easily capture relevant structure in these kind of data but deep networks are capable of discovering hidden structures within¬†these data in this course you will use tensorflow library to apply deep learning on different data types to solve real world problems
2	0.9829	Accelerating Deep Learning With Gpu	majority of data in the world are unlabeled and unstructured data for instance images sound and text data shallow neural networks cannot easily capture relevant structure in these kind of data but deep networks are capable of discovering hidden structures within¬†these data in this course you will use tensorflow library to apply deep learning on different data types to solve real world problems
3	0.9476	Data Science Bootcamp With R For University Proffesors	a multi day intensive in person data science bootcamp offered by big data university

Conclusions

The capstone project successfully developed a personalized course recommender system for an online learning platform using machine learning techniques. The main objectives were to enhance the relevance and accuracy of course recommendations, thereby improving learner engagement and course completion rates. The project involved several key components:

Exploratory Data Analysis (EDA):

The EDA revealed valuable insights into course enrollment patterns and popular course genres. For instance, genres like "Backend Dev," "Machine Learning," and "Data Analysis" were highly popular, while more niche topics like "Chatbot" and "Bioinformatics" had fewer courses.

Content-based Recommender System:

Implemented using unsupervised learning techniques, this system utilized user profiles and course metadata to generate personalized recommendations. Key steps included data preprocessing, feature engineering, and similarity calculation. The system effectively recommended courses based on the similarity between user profiles and course genres.

Conclusions

Collaborative Filtering Recommender System:

Using supervised learning techniques, particularly KNN and NMF, this system leveraged user interactions to enhance recommendation accuracy. The KNN model achieved an RMSE of 1.2895, while the NMF model implemented using the Surprise library performed slightly better with an RMSE of 1.2882.

Hybrid Recommender System:

By combining content-based and collaborative filtering approaches, the hybrid system aimed to provide the most comprehensive recommendations. This integration maximized the strengths of both techniques, leading to improved recommendation quality.

Neural Network Embedding-based Recommender System:

Advanced deep learning techniques were employed to develop a neural network embedding-based recommender system. The customized model achieved a test RMSE of 0.1429, demonstrating the potential of neural networks in capturing complex user-course interactions.

Appendix

Hyperparameters and Evaluation Metrics:

- Detailed descriptions of the hyperparameter settings used for each model, including the thresholds for recommendation scores and similarity calculations.

- RMSE values and other performance metrics for each recommender system, highlighting the effectiveness of different approaches.

Code Repositories:

- Links to the GitHub repositories containing the project code, allowing for reproducibility and further exploration by other researchers and developers.

GitHub Repository

Appendix

Datasets:

Information on the datasets used for the project, including sources, preprocessing steps, and feature engineering techniques.

Future Work:

Suggestions for future enhancements to the recommender system, such as incorporating additional data sources, experimenting with alternative algorithms, and improving the user interface for better user experience.

References:

A list of academic papers, online courses, and other resources that informed the development of the recommender system. Key references include the IBM Machine Learning Professional Certificate courses and related Coursera materials.

IBM Machine Learning Professional Certificate