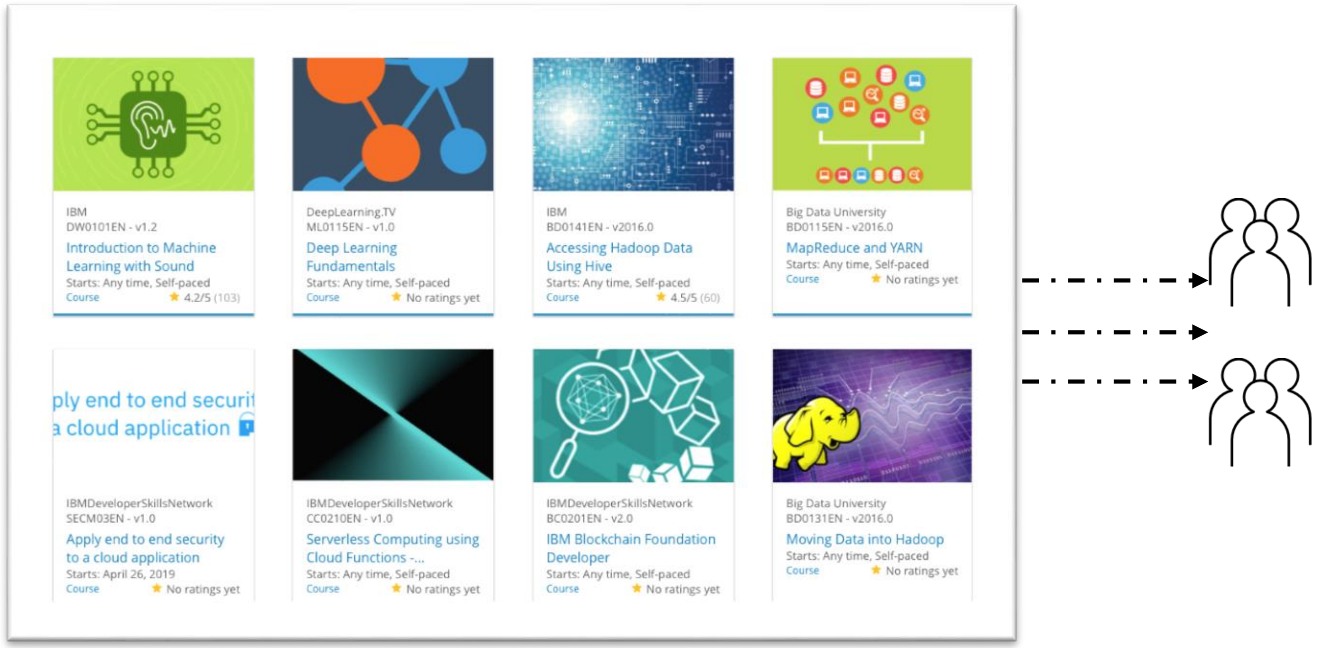
Personalized Online Course Recommender System with Machine Learning

Çağla Çağlar
2025-03-12



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

ply end to end securit
a cloud application

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

Project Background and Context

With the rapid expansion of online education, learners are presented with an overwhelming number of courses across various domains, including Machine Learning, Artificial Intelligence, Data Science, Cloud Computing, and Software Development. As the volume of available courses grows, identifying relevant and personalized learning opportunities becomes increasingly challenging. This project aims to develop an intelligent course recommender system that enhances course discovery by leveraging machine learning-based recommendation techniques.

Problem Statement and Hypothesis

The core challenge is the difficulty learners face in selecting courses that align with their interests and expertise levels, given the vast and continuously expanding course catalog. A data-driven recommendation system can help mitigate this challenge by analyzing learner interactions, course metadata, and content similarities. A hybrid approach combining content-based filtering, collaborative filtering, and deep learning techniques is expected to improve the quality of recommendations by accurately predicting relevant courses for each learner. To achieve this, supervised and unsupervised learning models, including KNN, NMF, deep neural networks, and gradient boosting classifiers, are implemented and compared to determine the most effective method for personalized course recommendations.

Exploratory Data Analysis

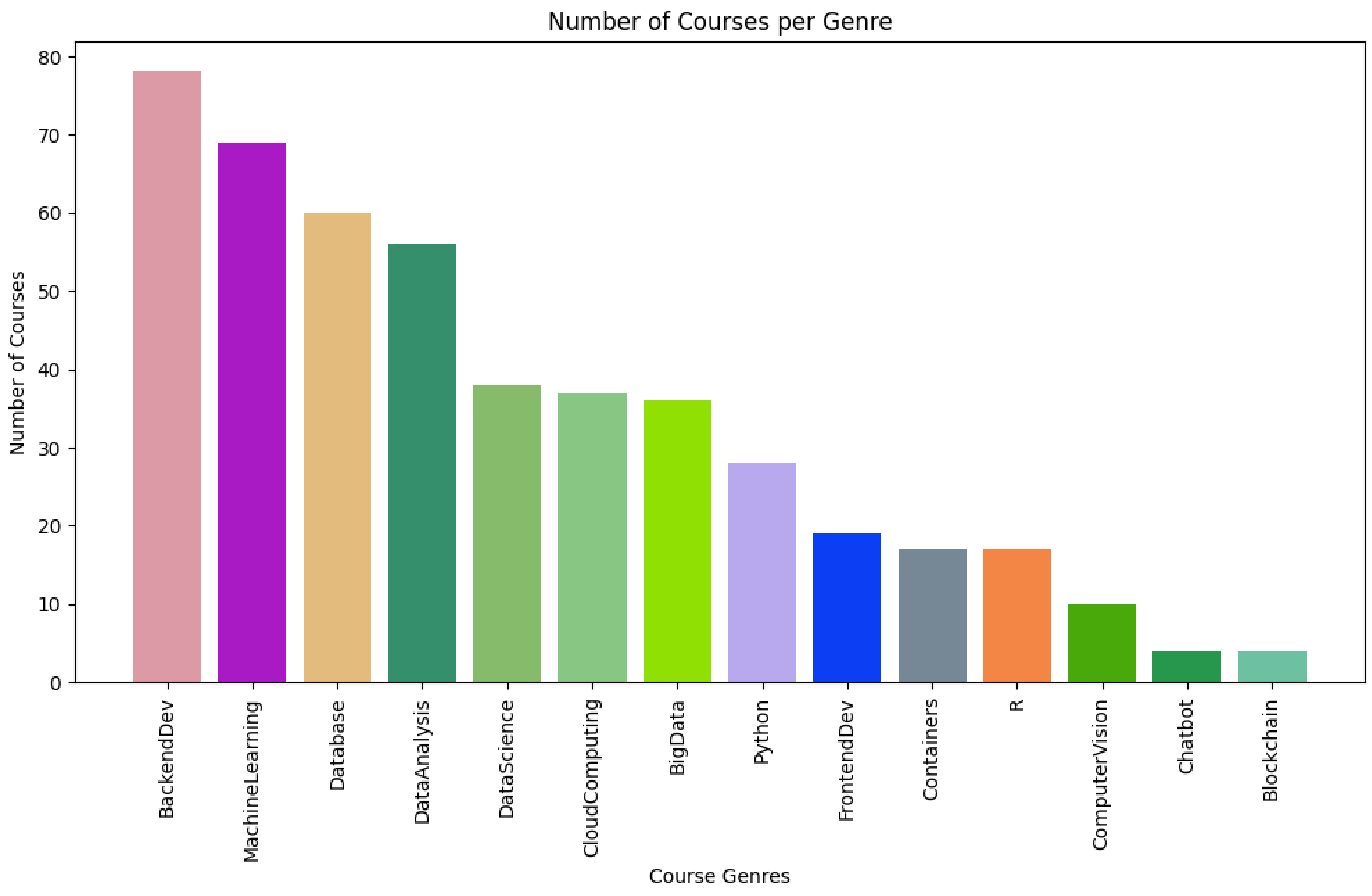
Course counts per genre

Dataset Overview

The dataset consists of **307 unique courses** with metadata such as course titles, genres, and enrollment information. A total of **233,306 enrollments** were recorded. The objective of this analysis is to identify **popular course genres, enrollment patterns, and keyword trends** that will guide the recommendation system.

Course Genre Distribution

Backend Development (78 courses) is the most frequent category, followed by **Machine Learning (69)** and **Databases (60)**. Niche topics like **Chatbots (4)** and **Blockchain (4)** have fewer courses, indicating limited content availability.

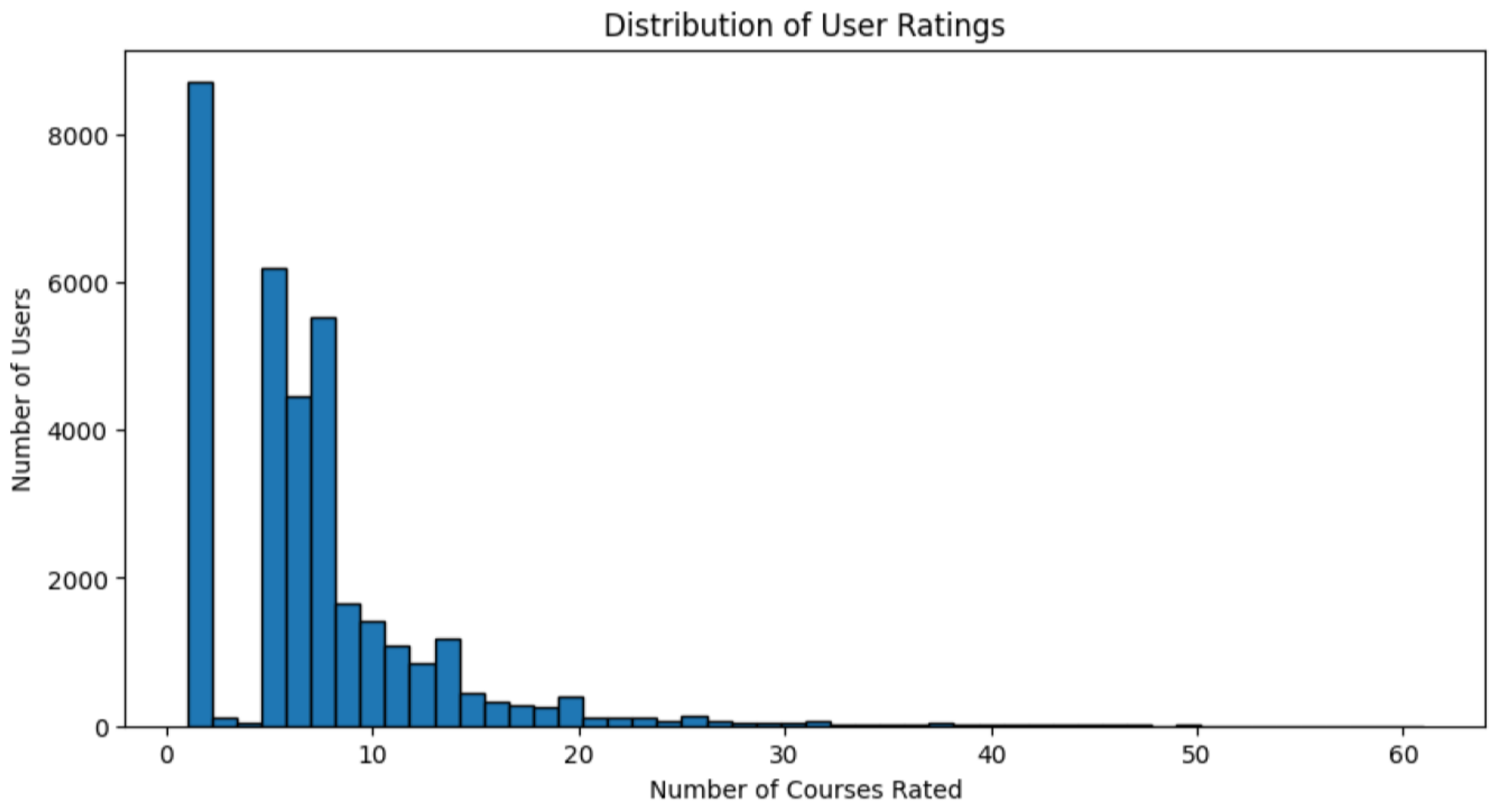


Insights & Impact on Recommendation System:

- Highly represented genres may require filtering to avoid redundant recommendations.
- Less frequent genres might struggle with the cold-start problem, requiring additional modeling strategies.

Course enrollment distribution

- The dataset contains **233,306 enrollment records** from **33,901 unique users**, highlighting a substantial engagement with online courses. Analyzing enrollment patterns provides key insights into user behavior, course popularity, and potential biases in recommendations.
- The histogram illustrates the **distribution of user interactions**, revealing that a **majority of users enroll in only a few courses**, while a smaller group engages with a significantly larger number of courses. The **median number of enrollments per user is 6**, with the highest engagement reaching **61 courses per user**. This skewed distribution suggests that while some courses attract a broad audience, others have a more niche appeal.



- Understanding these enrollment dynamics is crucial for designing a balanced recommender system. Highly enrolled courses tend to dominate recommendations, potentially introducing **popularity bias**. To ensure diversity, balancing techniques such as normalization and personalized filtering strategies should be considered.

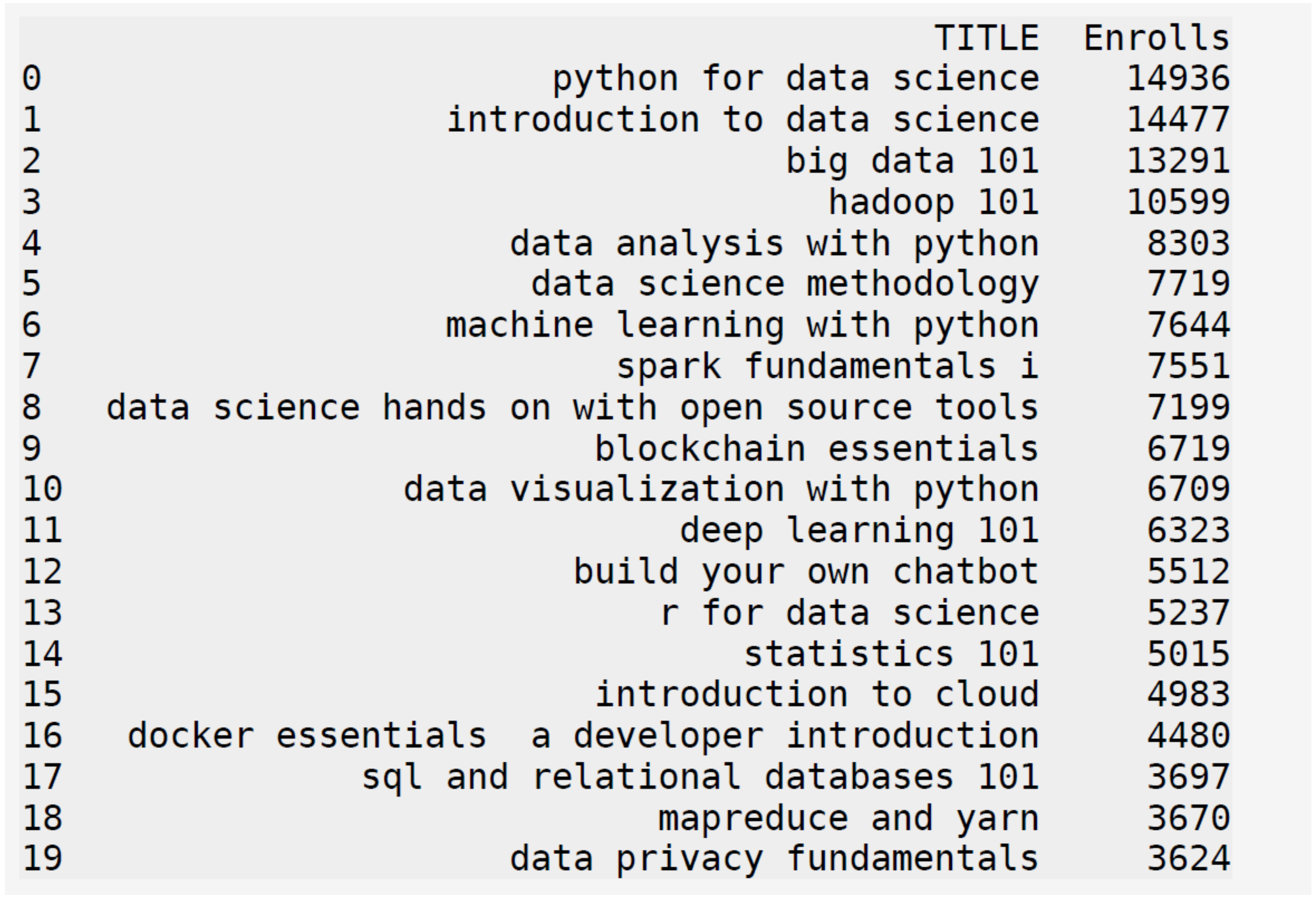
20 most popular courses

Most Popular Courses

• The **top 20 most enrolled courses** account for **63.3% of total enrollments.**
• Courses like **"Python for Data Science" (14,936 enrollments)** and **"Introduction to Data Science" (14,477 enrollments)** dominate.

Insights &Impact on the Recommendation System:

• **Highly enrolled courses serve as primary reference points** in recommendations, shaping learner engagement patterns.
• **Over-representation of popular courses may lead to "popularity bias,"** reducing exposure to less frequently enrolled but potentially valuable courses.
• **Re-ranking and diversity-aware filtering methods** can mitigate recommendation imbalances.



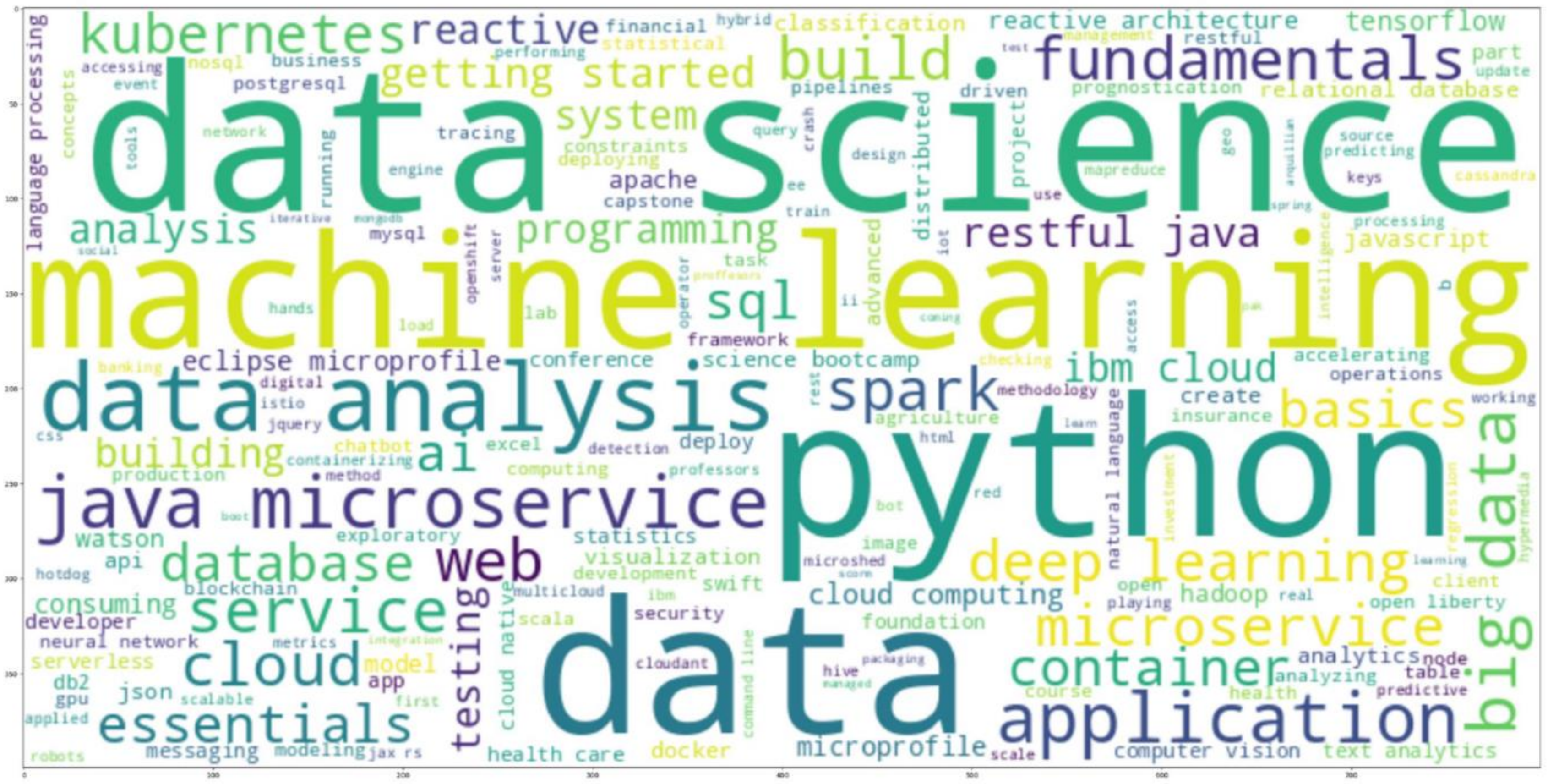
```
                                         TITLE  Enrolls
0                      python for data science    14936
1                 introduction to data science    14477
2                                 big data 101    13291
3                                   hadoop 101    10599
4                    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

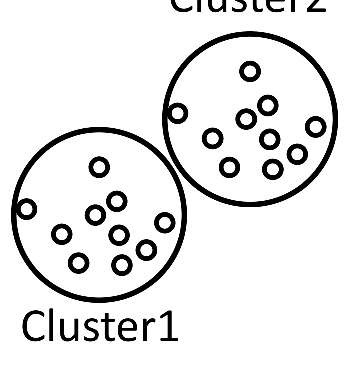
- Frequent terms include **"Data Science," "Machine Learning," "Python," "Big Data,"** and **"AI."**
- Indicates a strong focus on **IT, AI, and cloud-related fields.**

Insights & Impact on the Recommendation System:

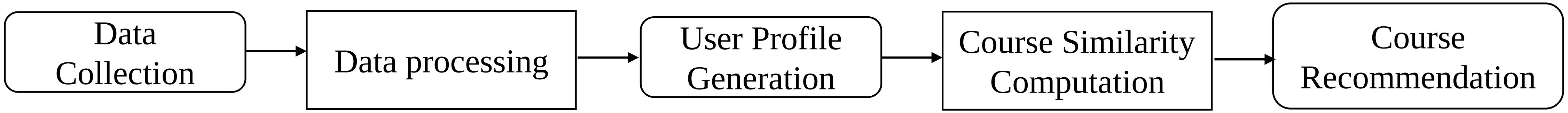
- **Text-based matching** can improve **content-based filtering.**
- Emerging trends in course content can be **identified through keyword analysis.**



Content-based Recommender System using Unsupervised Learning



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



This flowchart illustrates the **step-by-step process** of generating course recommendations based on user profiles and course genre vectors. The recommendation system follows these structured phases:

1. Data Collection:

Collects raw data, including user interactions, ratings, and course metadata such as course genres.

2. Data Preprocessing:

Cleans and processes the dataset by handling missing values, standardizing formats, and converting categorical variables (course genres) into numerical vectors.

3. User Profile Generation:

Constructs a **user profile vector** by computing weighted genre vectors based on past course ratings and interactions.

4. Course Similarity Computation:

Computes similarity scores using dot product similarity between user profile vectors and course genre vectors.

5. Course Recommendation:

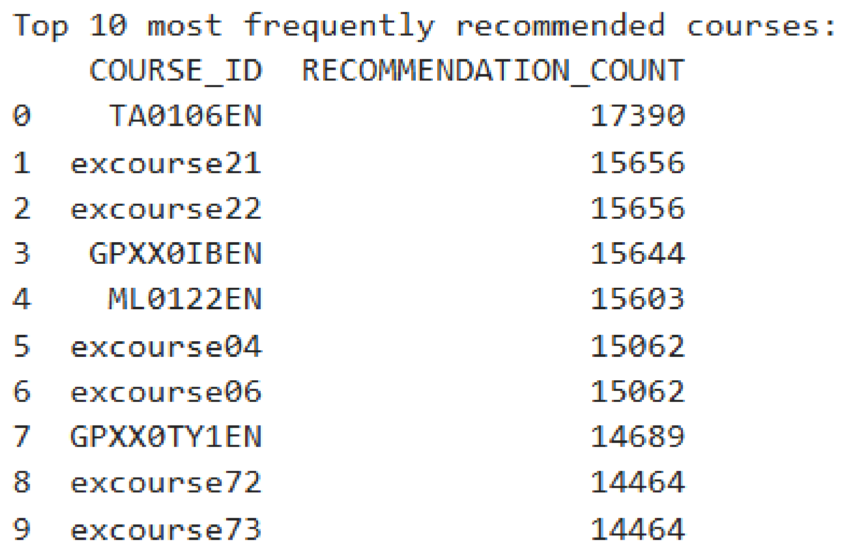
Ranks courses based on computed similarity scores and recommends the most relevant courses to each user.

This workflow ensures that recommendations remain **personalized, structured, and scalable** by dynamically updating user profiles based on their interactions. The content-based filtering method focuses on **feature extraction from courses** to align with user preferences.

Evaluation results of user profile-based recommender system

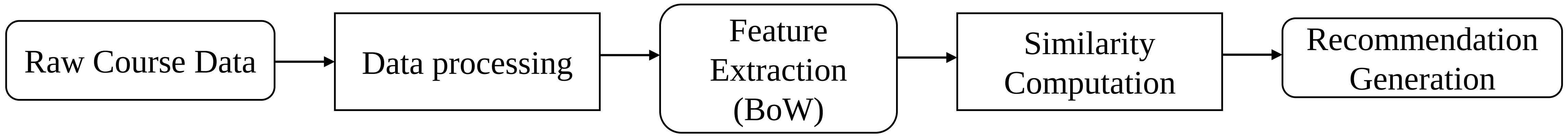
Hyperparameter Settings
The recommendation system utilizes a **dot product similarity** between user profile vectors and course genre vectors to generate recommendation scores. A **score threshold of 10.0** was applied to ensure high-confidence recommendations while filtering out less relevant courses.

- On average, each test user received **60.82 new course recommendations**. The number of recommendations per user depends on the threshold setting; lowering the threshold would increase the number of recommendations but may introduce less relevant results.
- The table presents the **top 10 most frequently recommended courses** based on the user profile-based recommender system. **TA0106EN** received the highest number of recommendations (**17,390 times**), followed by **excourse21** and **excourse22**, each recommended over **15,600 times**. The ranking reflects the **similarity between user profiles and course genre vectors**, ensuring that highly relevant courses are prioritized in recommendations.
- The user profile-based recommendation system effectively identifies and suggests courses aligned with users' past learning preferences. The use of profile vectors and genre similarity measures enables personalized recommendations with a high degree of relevance. Further refinements, such as adaptive threshold tuning or hybrid recommendation techniques, could enhance system performance.



```
Top 10 most frequently recommended courses:
    COURSE_ID  RECOMMENDATION_COUNT
0    TA0106EN                 17390
1  excourse21                 15656
2  excourse22                 15656
3   GPXX0IBEN                 15644
4    ML0122EN                 15603
5  excourse04                 15062
6  excourse06                 15062
7  GPXX0TY1EN                 14689
8  excourse72                 14464
9  excourse73                 14464
```

Flowchart of content-based recommender system using course similarity



A content-based recommender system suggests courses by analyzing textual similarities in their descriptions. The process follows these steps:

1. **Raw Course Data**: The dataset consists of course titles and descriptions, which serve as the input for feature extraction.
2. **Preprocessing**: Textual data undergoes **tokenization, stopword removal, and normalization** to ensure consistency.
3. **Feature Extraction (Bag of Words—BoW and Course Genres):** A Bag of Words (BoW) representation is created, transforming text into numerical vectors based on word frequencies, while course genres are also incorporated as an additional feature to enhance recommendation quality.
4. **Similarity Computation**: The **cosine similarity** metric is applied to the BoW vectors to construct a **course similarity matrix**, which quantifies the textual relevance between courses.
5. **Recommendation Generation**: For each user, the system identifies courses most similar to their enrolled courses by leveraging the similarity matrix. **Threshold-based filtering** ensures relevance.

By capturing course similarities through textual analysis, this method enables personalized course recommendations tailored to users’ learning preferences.

Evaluation results of course similarity based recommender system

Hyperparameter Settings & Methodology

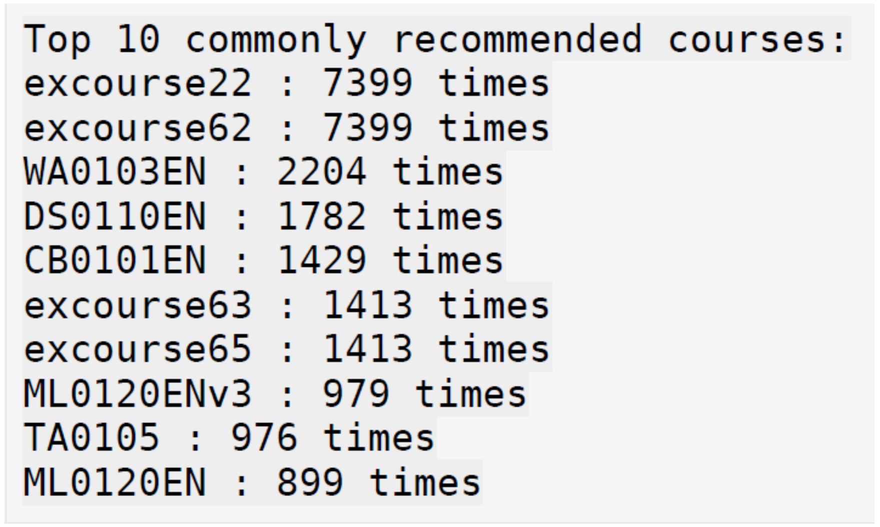
- A **content-based filtering approach** was used to generate recommendations by computing **cosine similarity** between course descriptions using **Bag of Words (BoW) features**.
- **Similarity threshold:** 0.6 (selected to ensure relevance while maintaining diversity).
- **Recommendation process:** The system ranks courses based on textual similarity and suggests the top-scoring ones for each user.
- **Capping:** A maximum of **20 recommendations per user** was enforced to prevent excessive suggestions.

New/Unseen Course Recommendations Per User

- **Average new courses recommended per user: 9.30**
- This suggests that, on average, users received a substantial number of personalized recommendations that they had not previously interacted with.
- The system effectively balances **exploration (new courses)** and **exploitation (similar courses).**

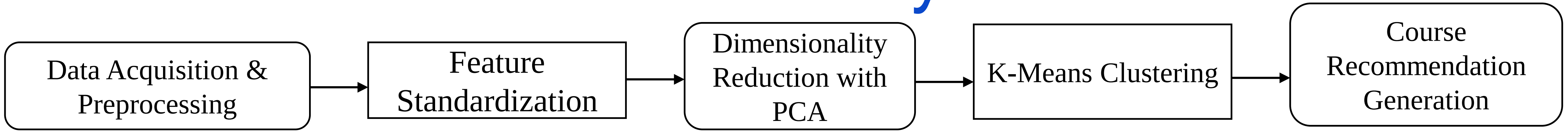
Key Insights

- The content-based recommender system primarily suggested "excourse22" and "excourse62," each recommended 7,399 times, indicating that certain courses dominate the recommendations. The distribution shows that the system tends to favor specific courses, likely due to their high textual similarity with multiple courses. While this concentration suggests a bias towards frequently occurring terms in course descriptions, the presence of other frequently recommended courses implies that the model does not entirely neglect diversity.



```
Top 10 commonly recommended courses:
excourse22 : 7399 times
excourse62 : 7399 times
WA0103EN : 2204 times
DS0110EN : 1782 times
CB0101EN : 1429 times
excourse63 : 1413 times
excourse65 : 1413 times
ML0120ENv3 : 979 times
TA0105 : 976 times
ML0120EN : 899 times
```

Flowchart of clustering-based recommender system


Data Acquisition & Preprocessing
Feature Standardization
Dimensionality Reduction with PCA
K-Means Clustering
Course Recommendation Generation

This workflow represents the **clustering-based recommendation system**, which groups users with similar learning preferences and suggests courses based on cluster-wide popularity. The approach follows five key steps:

1. Data Acquisition & Preprocessing

- The user profile dataset is loaded, containing course enrollment data across various domains (e.g., Machine Learning, Data Science, Cloud Computing, etc.).
- Missing values are handled, and data is cleaned for further analysis.

2. Feature Standardization

- Features are normalized using **StandardScaler** to ensure equal weightage across different attributes.
- Each course genre is transformed to have a mean of **0** and a standard deviation of **1** to improve clustering performance.

3. Dimensionality Reduction with PCA

- **Principal Component Analysis (PCA)** is applied to reduce feature dimensionality while retaining over **90% of variance**.
- Redundant and correlated features are eliminated, improving computational efficiency.

4. K-Means Clustering

- The Elbow Method, Silhouette Score, and Davies-Bouldin Index are used to determine the optimal number of clusters, balancing within-cluster compactness and between-cluster separation.
- K-Means clustering is performed on both original and PCA-transformed feature sets.
- Users are assigned to specific learning clusters based on their course enrollment patterns.

5. Course Recommendation Generation

- Popular courses within each cluster are identified based on enrollment frequency. A **popularity threshold** (e.g., courses with at least 50 enrollments) is applied to ensure that only widely preferred courses are recommended.
- For each test user, courses already enrolled in are excluded.
- The system recommends the most popular courses within the same cluster, ensuring personalized recommendations based on shared learning interests.

Evaluation results of clustering-based recommender system

Hyperparameter Optimization

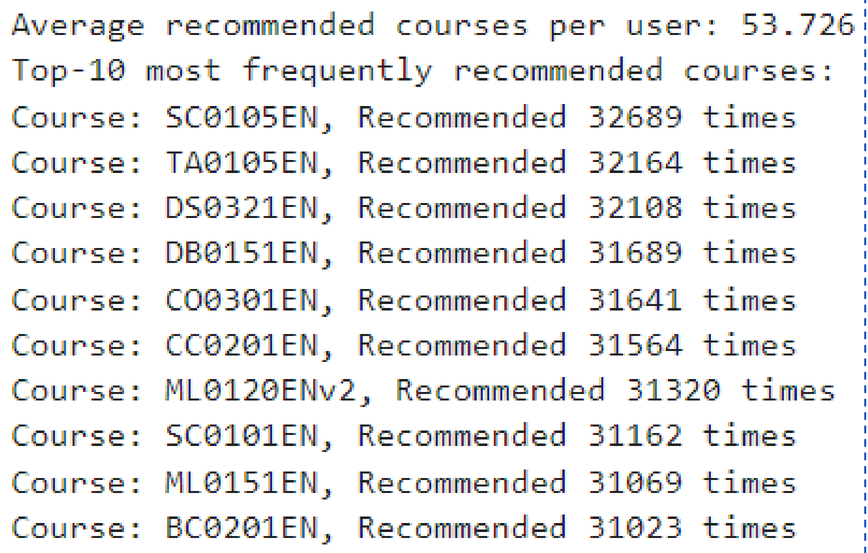
•The **Elbow Method**, **Silhouette Score**, and **Davies-Bouldin Index** were used to determine the optimal number of clusters.

•The best cluster size for **K-Means** was found to be **8**, ensuring well-separated user groups with minimal intra-cluster variance.

• **Principal Component Analysis (PCA)** was applied to reduce dimensionality while preserving over **90% variance**, leading to improved clustering performance.

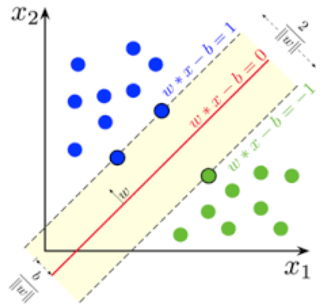
Recommendation Performance Analysis

- **Average Recommended Courses per User**
- On average, each test user received **53.73** new/unseen course recommendations.
- Recommendations were generated by identifying the most popular courses within the user's assigned cluster while filtering out already enrolled courses.
- A **popularity threshold** of **50 enrollments** was applied to ensure recommendations are based on widely preferred courses.
- **Conclusion**
- The clustering-based recommendation system effectively groups users based on their course preferences and suggests relevant courses.
- By applying **hyperparameter tuning** and **popularity-based filtering**, personalized recommendations were improved, reducing noise and ensuring high-quality suggestions.
- **Alternative clustering techniques (DBSCAN, Hierarchical Clustering)** were explored for their potential in user segmentation.
- Future enhancements may include **adaptive popularity thresholds** and investigating **ensemble clustering approaches** to further optimize recommendations.

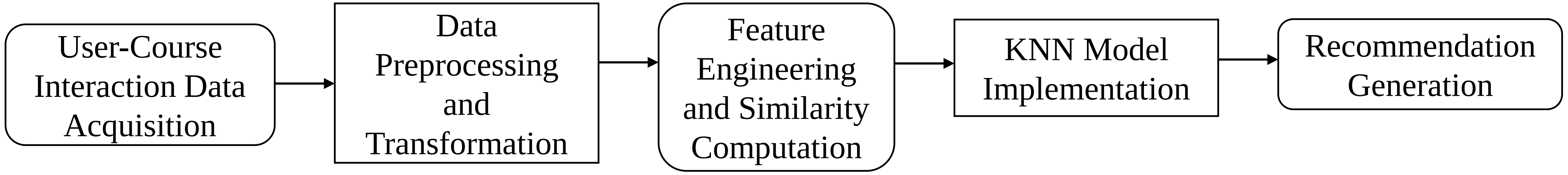


```
Average recommended courses per user: 53.726
Top-10 most frequently recommended courses:
Course: SC0105EN, Recommended 32689 times
Course: TA0105EN, Recommended 32164 times
Course: DS0321EN, Recommended 32108 times
Course: DB0151EN, Recommended 31689 times
Course: CO0301EN, Recommended 31641 times
Course: CC0201EN, Recommended 31564 times
Course: ML0120ENv2, Recommended 31320 times
Course: SC0101EN, Recommended 31162 times
Course: ML0151EN, Recommended 31069 times
Course: BC0201EN, Recommended 31023 times
```

Collaborative-filtering Recommender System using Supervised Learning



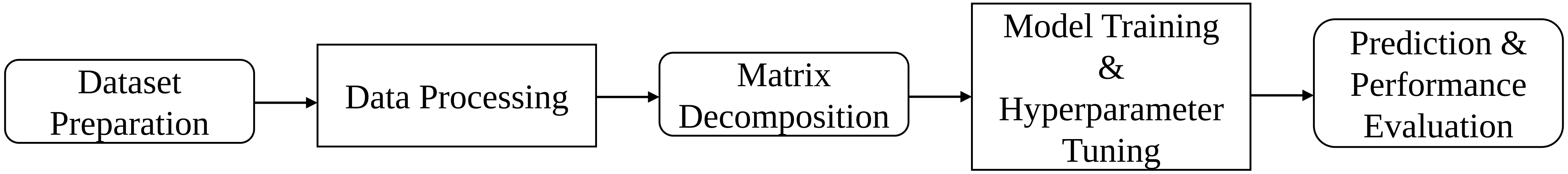
Flowchart of KNN based recommender system


User-Course Interaction Data Acquisition
Data Preprocessing and Transformation
Feature Engineering and Similarity Computation
KNN Model Implementation
Recommendation Generation

This flowchart provides a structured representation of the **K-Nearest Neighbors (KNN) collaborative filtering-based recommendation system** using course enrollment data. The methodology ensures efficient similarity-based course recommendations through structured data processing and modeling steps.

1. **User-Course Interaction Data Acquisition** → The dataset consists of user-course interactions, where each row represents a user, each column represents a course, and the values indicate user engagement levels (e.g., ratings, enrollments, or completions).
2. **Data Preprocessing and Transformation** → The raw data is cleaned by handling missing values and inconsistencies. A **user-item interaction matrix** is generated, where each user-course pair is represented in a structured format. Sparse matrix representations are used to optimize storage and computational efficiency.
3. **Feature Engineering and Similarity Computation** → The matrix is transformed into a format suitable for similarity calculations. Cosine similarity is applied to determine relationships between users or courses, identifying patterns based on shared behaviors.
4. **KNN Model Implementation** → The K-Nearest Neighbors algorithm is applied to identify the most similar users (user-based filtering) or the most similar courses (item-based filtering). Two implementation options are provided: a streamlined approach using the Surprise library, which specializes in recommendation systems and offers built-in evaluation metrics, and a more customizable implementation using sklearn, numpy, and pandas, allowing for greater flexibility and control. The similarity scores define the nearest neighbors, which form the basis for recommendation.
5. **Recommendation Generation** → Using the weighted contributions of similar users or items, the system predicts missing ratings and suggests courses that the user is most likely to engage with.

Flowchart of NMF based recommender system

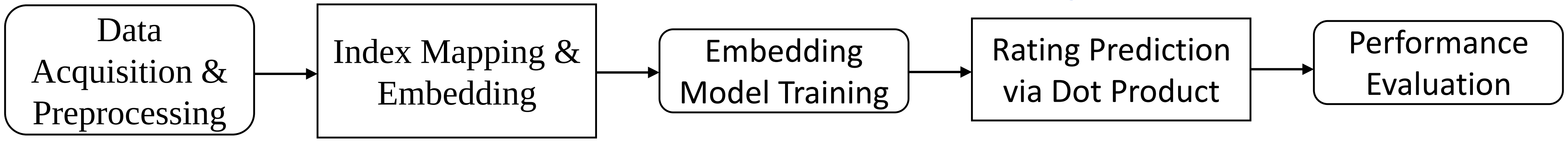

Dataset Preparation
Data Processing
Matrix Decomposition
Model Training & Hyperparameter Tuning
Prediction & Performance Evaluation

Non-negative Matrix Factorization (NMF) is a matrix decomposition technique used to reduce the dimensionality of sparse user-item interaction data. This approach enables the extraction of latent factors that represent underlying user preferences and item characteristics, facilitating personalized recommendations.

1. **Dataset Preparation**: The user-item interaction dataset is collected, where each entry represents a rating given by a user to a specific item. The dataset is structured to ensure consistency and completeness.
2. **Data Processing**: The raw dataset is converted into a structured matrix format. Missing values are handled through imputation techniques, and the data is normalized to enhance model stability and accuracy.
3. **Matrix Decomposition**: The user-item interaction matrix is factorized into two lower-dimensional dense matrices, one representing latent user features and the other representing latent item features. This decomposition enables capturing underlying patterns in user preferences.
4. **Model Training & Hyperparameter Tuning**: The NMF model is trained on the processed data using a stochastic gradient descent optimization approach. Key hyperparameters, such as the number of latent factors and initialization parameters, are systematically tuned to enhance predictive performance. Additionally, an alternative NMF implementation is performed using a combination of numpy, pandas, and scikit-learn, which applies a multiplicative update rule instead of stochastic gradient descent.
5. **Prediction & Performance Evaluation**: The trained model estimates missing ratings by computing the dot product of the decomposed matrices. The model's performance is evaluated using metrics such as Root Mean Square Error (RMSE) to assess accuracy and effectiveness.

This structured workflow ensures an efficient and scalable recommendation system capable of providing accurate and personalized predictions.

Flowchart of Neural Network Embedding based recommender system


Data Acquisition & Preprocessing
Index Mapping & Embedding
Embedding Model Training
Rating Prediction via Dot Product
Performance Evaluation

This study presents a neural network-based collaborative filtering system for predicting user ratings in an online course recommendation scenario. The model automatically learns latent user and item features without requiring explicit feature engineering.

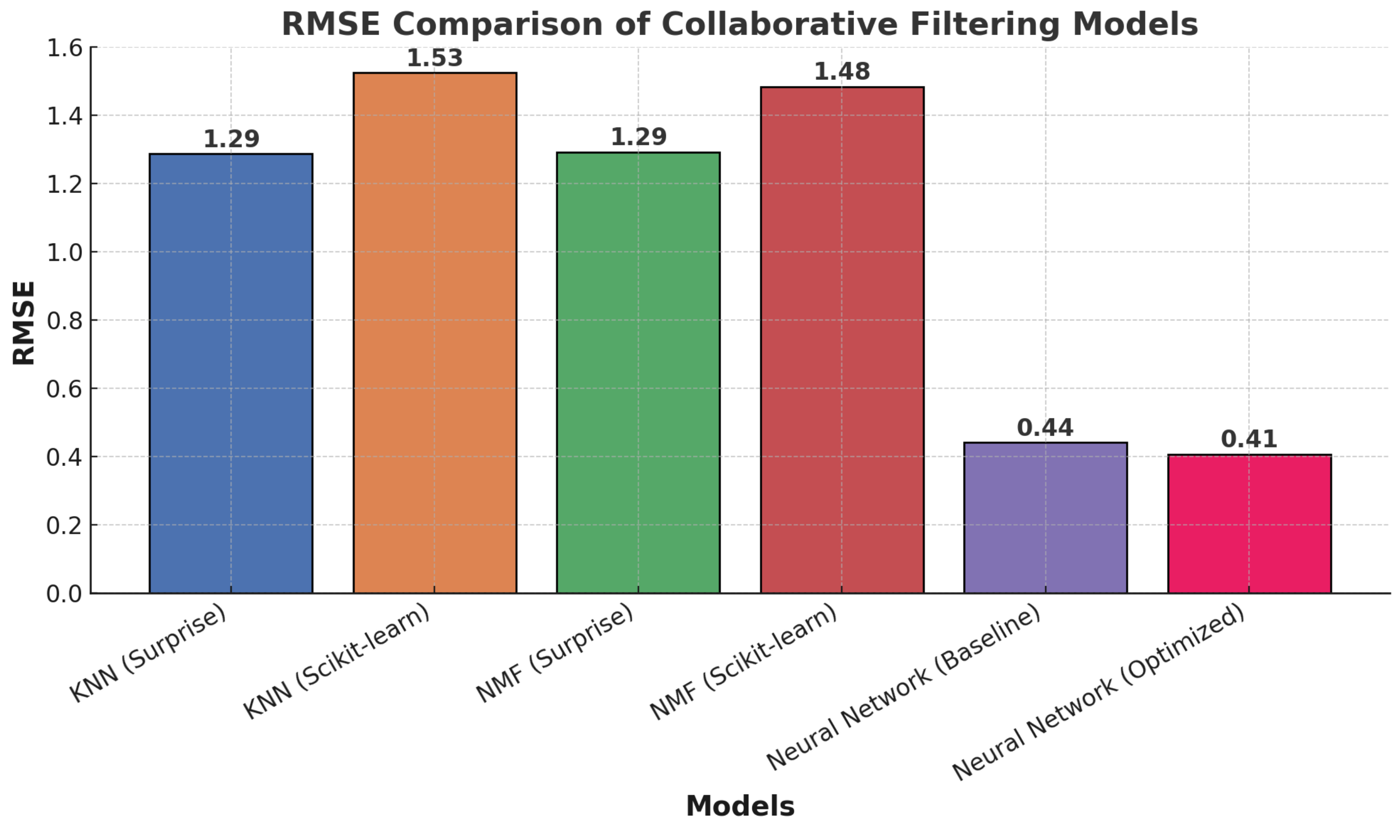
- **Data Ingestion & Preprocessing:** The dataset comprises user-course interactions structured with three key attributes: user_id, course_id, and rating. Data integrity checks are performed, missing values are handled, and the dataset is split into training (80%), validation (10%), and test (10%) subsets.
- **Index Mapping & Embeddings:** User and course IDs are mapped to unique integer indices and embedded into the neural network. This transformation enables the model to learn latent feature representations efficiently.
- **Embedding Model Training:** A deep learning-based recommendation model is built using TensorFlow and Keras. User and course embeddings are learned via dense latent feature representations, reducing high-dimensional sparse vectors into compact numerical representations. The embedding size is set to 16, and the model is optimized using Mean Squared Error (MSE) loss with the Adam optimizer. Additionally, an optimized version of the neural network was implemented with hyperparameter tuning, additional hidden layers, dropout, and batch normalization to improve performance. This enhanced model, incorporating bias terms, yielded a lower RMSE, demonstrating the benefits of deeper architectures and regularization techniques.
- **Rating Prediction via Dot Product:** The trained embeddings are combined via a dot product operation, generating predicted user-course ratings. User and course-specific bias terms are included to enhance prediction accuracy.
- **Performance Evaluation:** The model's accuracy is assessed using Root Mean Squared Error (RMSE) to quantify prediction error. The optimized model, with added regularization and deeper architecture, further improved RMSE performance, demonstrating the benefits of refined embeddings. This neural network-based approach eliminates the need for explicit feature engineering, allowing the model to autonomously learn user preferences and course characteristics, improving recommendation quality and efficiency.

Performance Comparison of Collaborative Filtering Models

This bar chart presents a comparative analysis of different collaborative filtering models based on **Root Mean Squared Error (RMSE)**. The results highlight the varying performance levels across models:

- The **KNN model (Surprise implementation)** and **NMF model (Surprise implementation)** exhibit **1.29 RMSE**, demonstrating a relatively lower prediction error compared to other collaborative filtering implementations. However, the **Scikit-learn implementations of KNN and NMF** have higher error rates, reaching **1.53 and 1.48 RMSE**, respectively. This suggests that implementation differences and hyperparameter tuning significantly affect performance.
- The **baseline neural network model** demonstrates a much lower RMSE (**0.44**) compared to traditional collaborative filtering approaches. The **optimized neural network model further improves performance**, achieving the lowest RMSE of **0.41**, demonstrating its effectiveness in capturing complex patterns in user-course interactions.

General observation: The findings indicate that **neural network-based approaches outperform traditional matrix factorization and nearest-neighbor-based models**. These models effectively reduce prediction error, making them a more reliable choice for personalized recommendations.

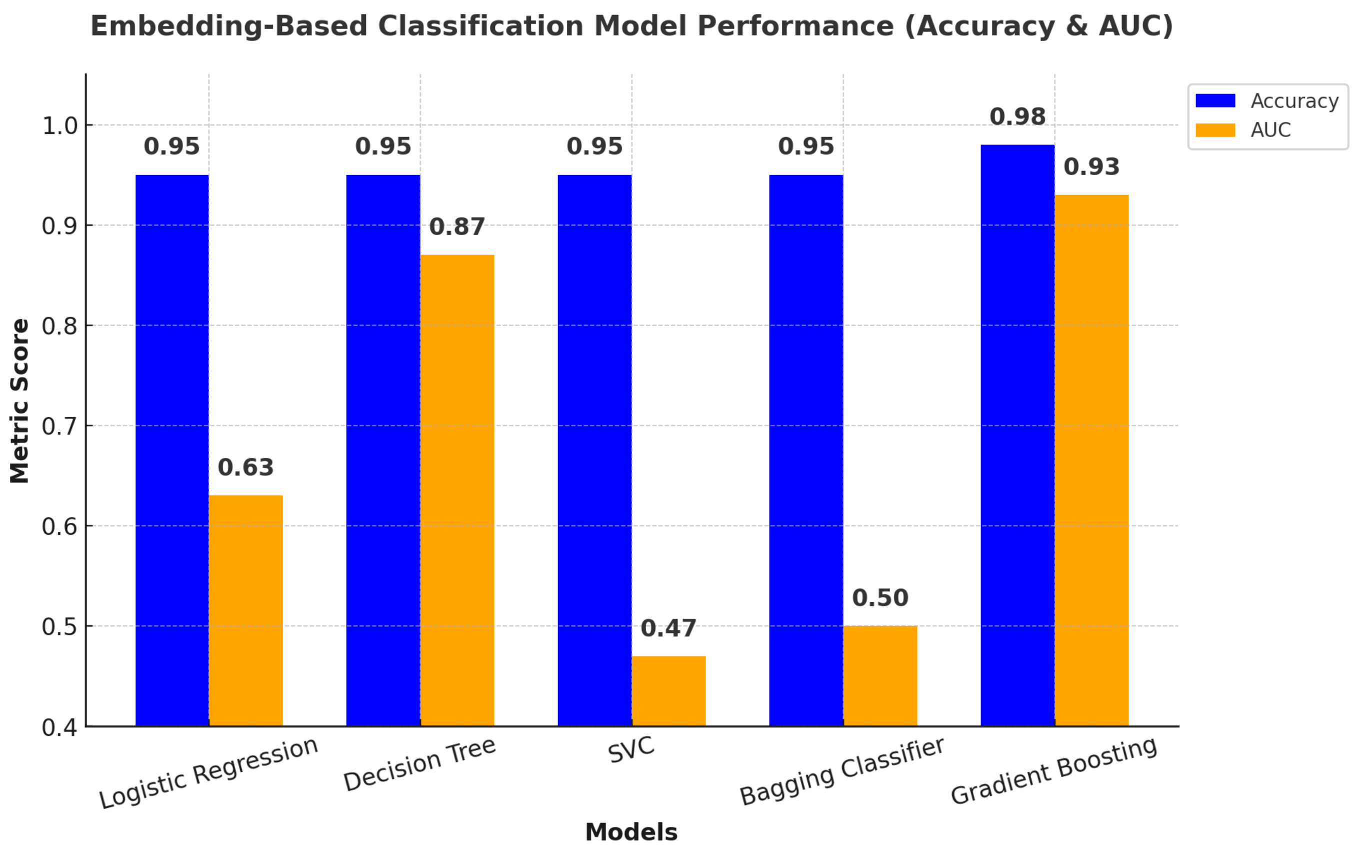


Performance Comparison of embedding based classification models

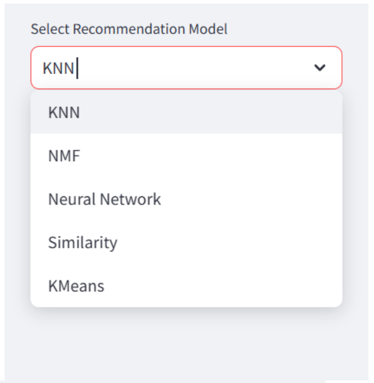
This bar chart presents a comparative analysis of different classification models utilizing embedding-based features, evaluated using Accuracy and AUC (Area Under the Curve). The results highlight the varying classification performance across models:

- Logistic Regression and Decision Tree classifiers achieve high accuracy (95.45% and 95.43%, respectively), but their AUC scores differ significantly. While Decision Tree attains an AUC of 0.8656, Logistic Regression exhibits a notably lower AUC of 0.6289, indicating a weaker ability to differentiate between classes.
- Support Vector Machine (SVC) achieves the same accuracy as Logistic Regression but has the lowest AUC (0.4685), suggesting that, despite correct classifications, its decision boundary may not be well optimized for class separation. The Bagging classifier, which ensembles SVC models, marginally improves robustness but maintains an AUC of 0.50, showing limited performance gains.
- The Gradient Boosting classifier outperforms all other models, achieving the highest accuracy (97.88%) and the highest AUC (0.9309). This suggests that boosting techniques effectively leverage embedding-based features, enhancing the model's ability to capture complex patterns in the data.

General observation: The findings indicate that tree-based ensemble methods, particularly Gradient Boosting, provide superior classification performance compared to linear models and support vector machines. The use of embedding features enhances predictive capability, and models with higher AUC scores demonstrate stronger class differentiation, making them more reliable for real-world applications.



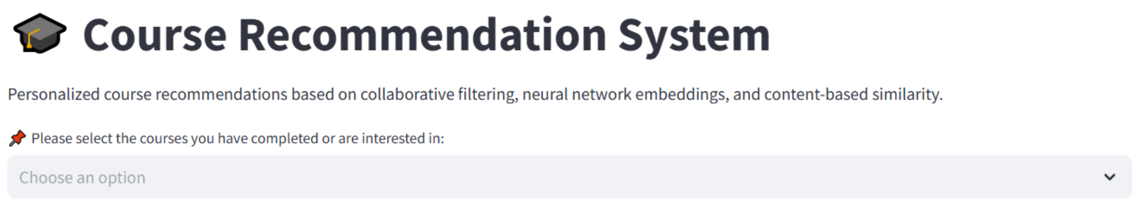
Course recommender system app with Streamlit



Select Recommendation Model

KNN

- KNN
- NMF
- Neural Network
- Similarity
- KMeans

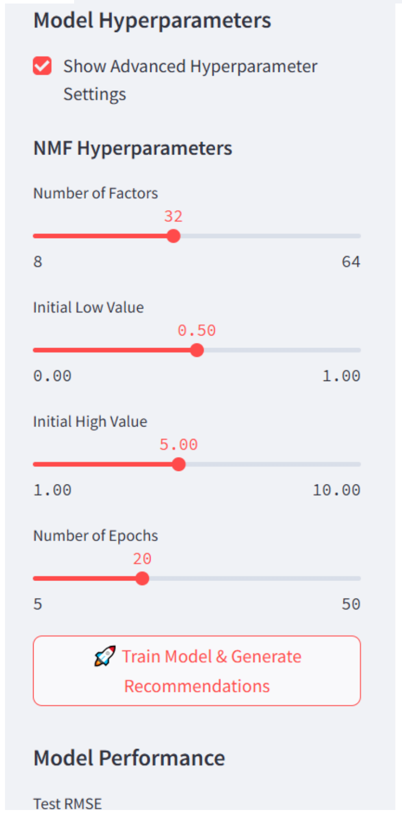


🎓 Course Recommendation System

Personalized course recommendations based on collaborative filtering, neural network embeddings, and content-based similarity.

📌 Please select the courses you have completed or are interested in:

Choose an option



Model Hyperparameters

✓ Show Advanced Hyperparameter Settings

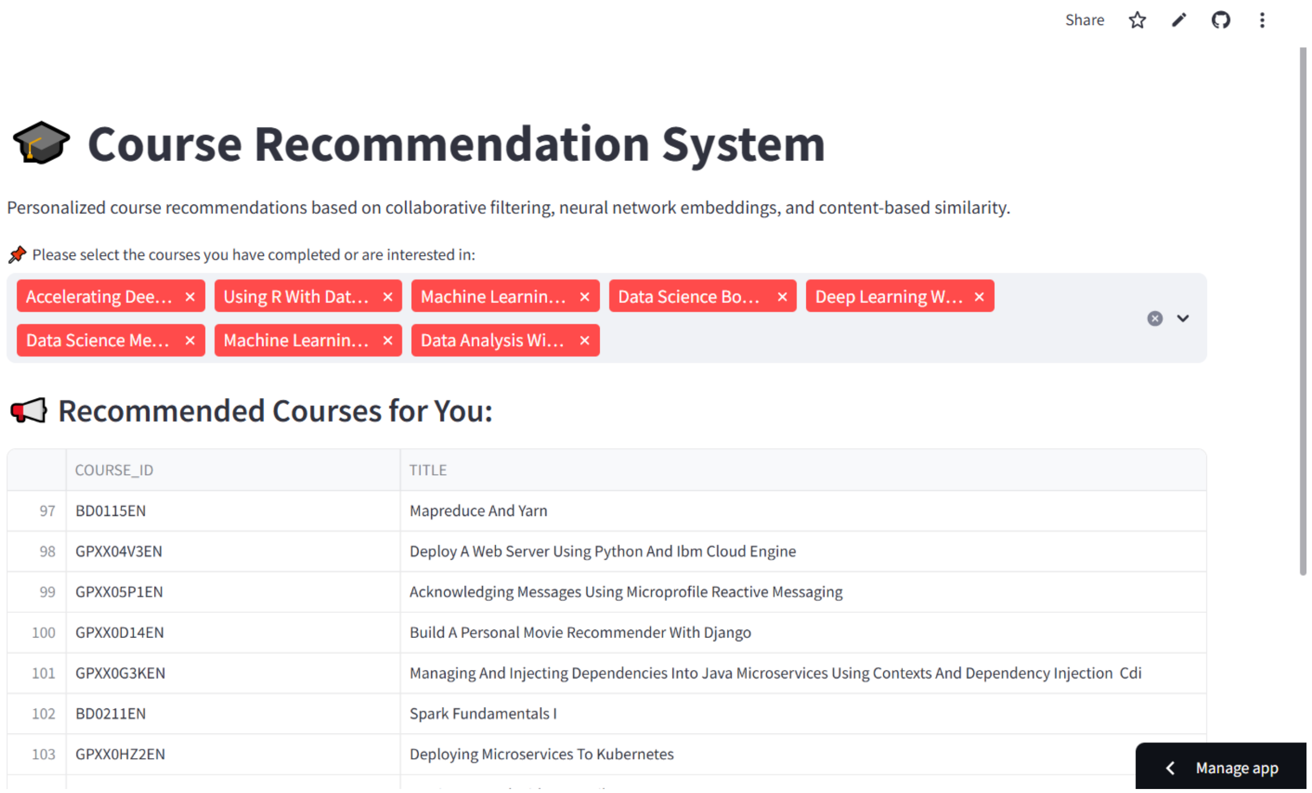
NMF Hyperparameters

Number of Factors	32	8 – 64
Initial Low Value	0.50	0.00 – 1.00
Initial High Value	5.00	1.00 – 10.00
Number of Epochs	20	5 – 50

🚀 Train Model & Generate Recommendations

Model Performance

Test RMSE



🎓 Course Recommendation System

Personalized course recommendations based on collaborative filtering, neural network embeddings, and content-based similarity.

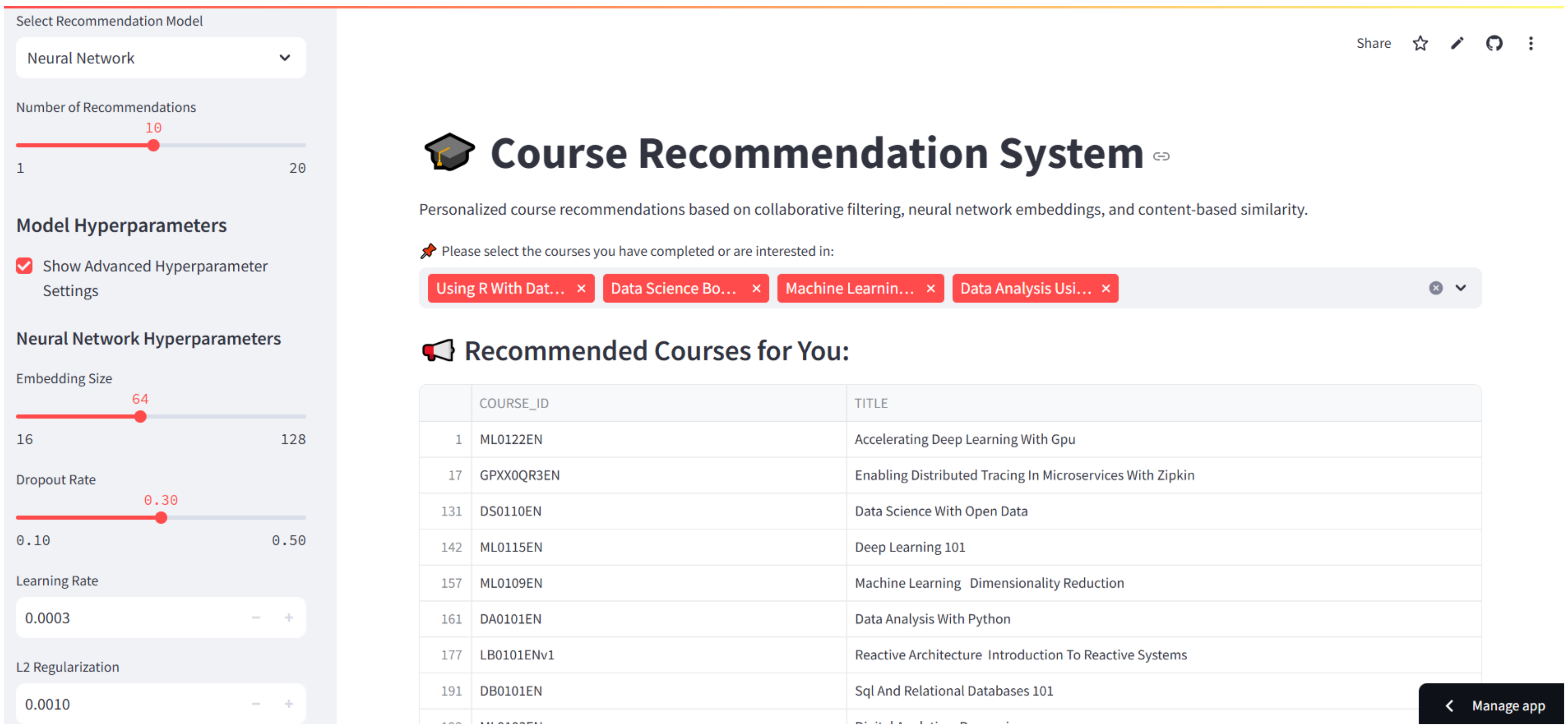
📌 Please select the courses you have completed or are interested in:

Accelerating Dee… × Using R With Dat… × Machine Learnin… × Data Science Bo… × Deep Learning W… × Data Science Me… × Machine Learnin… × Data Analysis Wi… ×

📢 Recommended Courses for You:

	COURSE_ID	TITLE
97	BD0115EN	Mapreduce And Yarn
98	GPXX04V3EN	Deploy A Web Server Using Python And Ibm Cloud Engine
99	GPXX05P1EN	Acknowledging Messages Using Microprofile Reactive Messaging
100	GPXX0D14EN	Build A Personal Movie Recommender With Django
101	GPXX0G3KEN	Managing And Injecting Dependencies Into Java Microservices Using Contexts And Dependency Injection Cdi
102	BD0211EN	Spark Fundamentals I
103	GPXX0HZ2EN	Deploying Microservices To Kubernetes

Course recommender system app with Streamlit



Select Recommendation Model

Neural Network

Number of Recommendations

10

1 20

Model Hyperparameters

☑ Show Advanced Hyperparameter Settings

Neural Network Hyperparameters

Embedding Size

64

16 128

Dropout Rate

0.30

0.10 0.50

Learning Rate

0.0003

L2 Regularization

0.0010

Course Recommendation System

Personalized course recommendations based on collaborative filtering, neural network embeddings, and content-based similarity.

📌 Please select the courses you have completed or are interested in:

Using R With Dat… × Data Science Bo… × Machine Learnin… × Data Analysis Usi… ×

📢 Recommended Courses for You:

	COURSE_ID	TITLE
1	ML0122EN	Accelerating Deep Learning With Gpu
17	GPXX0QR3EN	Enabling Distributed Tracing In Microservices With Zipkin
131	DS0110EN	Data Science With Open Data
142	ML0115EN	Deep Learning 101
157	ML0109EN	Machine Learning Dimensionality Reduction
161	DA0101EN	Data Analysis With Python
177	LB0101ENv1	Reactive Architecture Introduction To Reactive Systems
191	DB0101EN	Sql And Relational Databases 101

Personalized Course Recommendation System: A Multi-Model Approach

This web application provides personalized course recommendations by integrating multiple recommendation paradigms, including collaborative filtering, neural network embeddings, clustering, and content-based similarity analysis. The system processes course-user interaction data to generate customized learning pathways.

Technical Implementation

- **Algorithmic Diversity**: Implements KNN, Non-Negative Matrix Factorization (NMF), Neural Networks with embedding layers, K-Means clustering, and cosine similarity-based content matching.
- **Model Fine-Tuning**: Configurable hyperparameters for optimizing performance (e.g., k-values, embedding dimensions, learning rates, regularization).
- **Performance Evaluation**: Predictive models assessed via RMSE, clustering quality evaluated using silhouette coefficients.
- **Interactive Recommendation Generation**: Real-time course recommendations based on user selections with dynamic feedback.

User Interaction Workflow

1. Select completed courses to indicate learning preferences.
2. Choose a recommendation algorithm suited to dataset characteristics.
3. Optionally configure hyperparameters for advanced tuning.
4. Receive ranked course suggestions with relevance indicators.

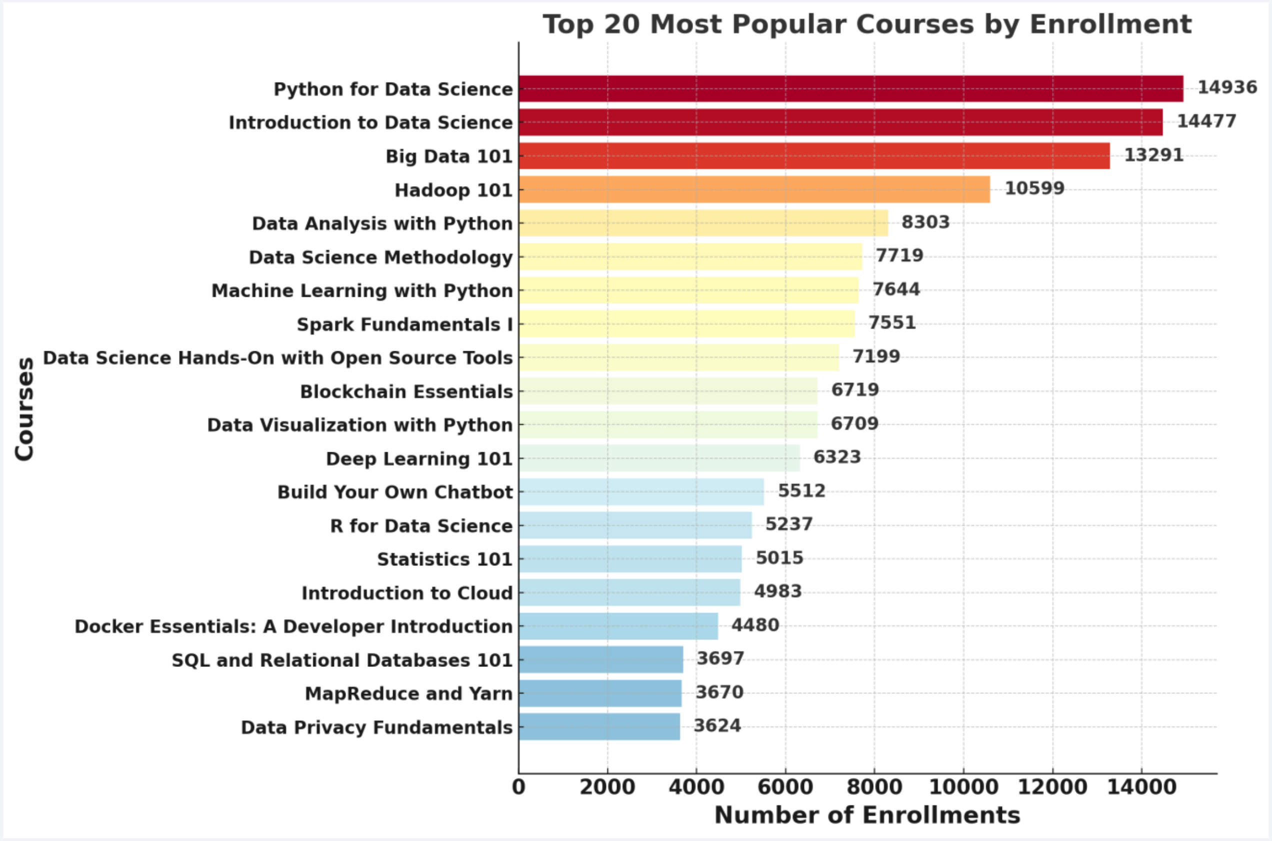
By integrating multiple recommendation strategies, the system enhances personalized learning by providing insights into both course relationships and user preferences, facilitating a data-driven approach to education.

📌 Access the application: https://personalized-online-course-recommender-system.streamlit.app/

Conclusion of Exploratory Data Analysis

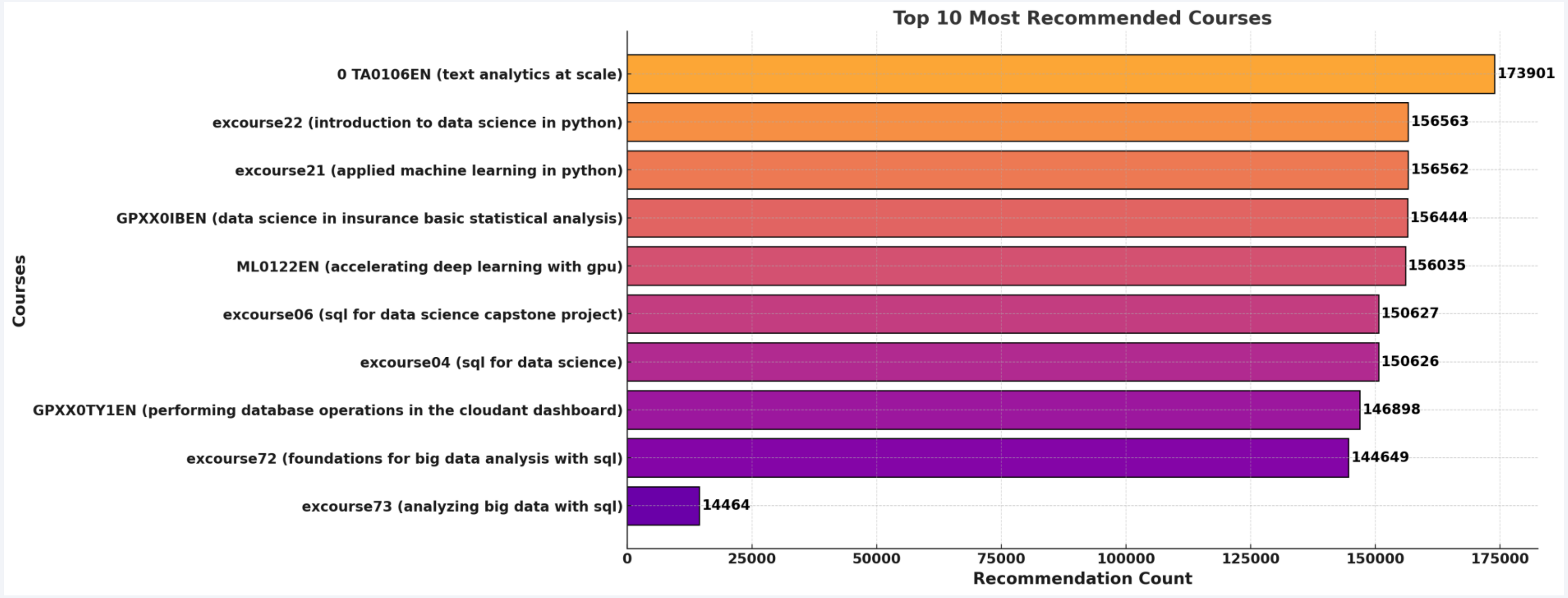
The exploratory data analysis (EDA) provided critical insights into user behavior, course popularity, and content distribution, forming the foundation for an effective recommendation system.

- **Course Distribution:** The dataset consists of **307 unique courses** across diverse topics, with **Backend Development, Machine Learning, and Databases** being the most frequently represented genres. In contrast, niche topics such as **Chatbots and Blockchain** have limited content, which may pose challenges for recommendations.
- **User Engagement Patterns:** A total of **233,306 enrollments** from **33,901 unique users** were analyzed. Most users enroll in **a limited number of courses**, while a small fraction exhibit significantly higher engagement, leading to a highly skewed distribution. The **top 20 most popular courses account for 63.3% of all enrollments**, revealing a strong **popularity bias** in user preferences.
- **Keyword Analysis:** A **word cloud of course titles** showed a strong emphasis on **AI, Data Science, and Cloud Computing**, reinforcing the dominance of technology-driven learning content.
- These findings highlight the need for a **balanced recommendation strategy**, ensuring that highly popular courses do not dominate suggestions while also addressing the **cold-start problem for niche topics**. The insights gained from EDA will be leveraged in feature engineering and model selection to optimize course recommendations.



Conclusion of Content-Based Recommender System Using User Profile and Course Genres

A **content-based recommender system** was implemented to generate personalized course recommendations based on **user profiles and course genres**. The system constructs **user feature vectors** by analyzing the genres of previously enrolled courses and user ratings.



• **User Profile Generation:** Each user's interests were represented as a weighted feature vector based on their past enrollments and ratings. This profile was then matched against the genre vectors of available courses.

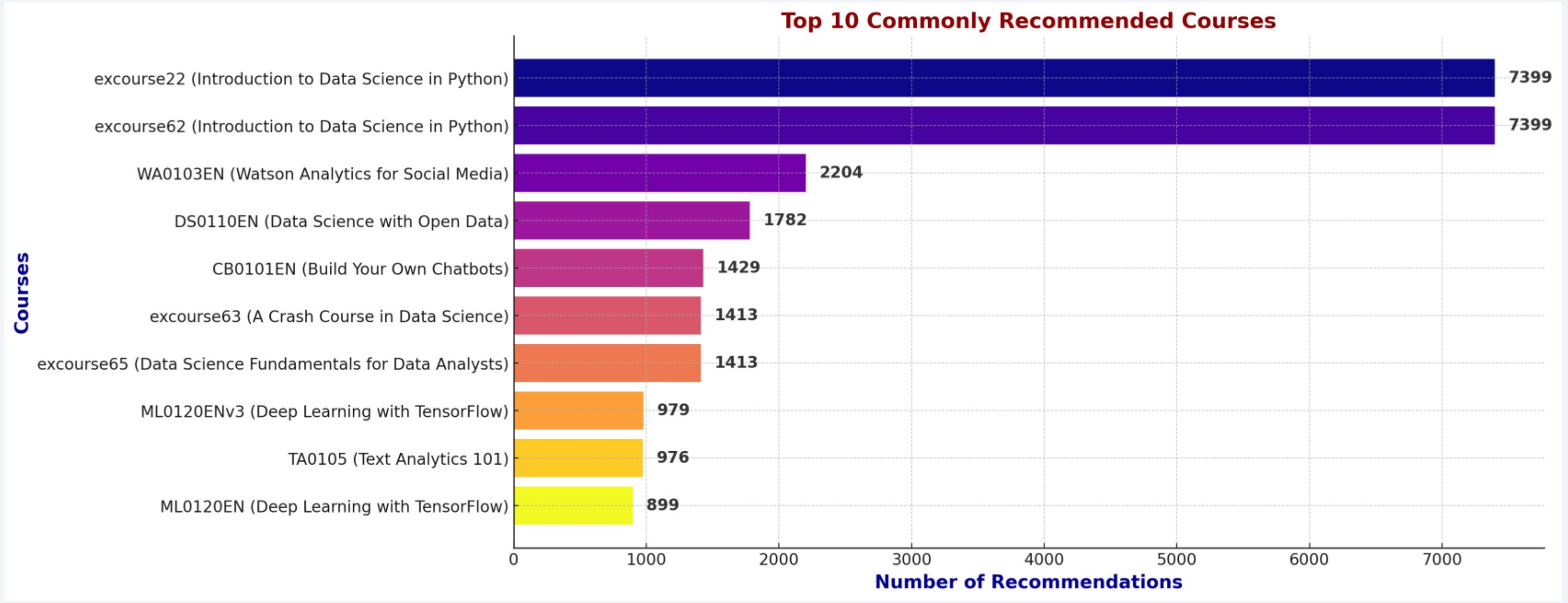
• **Recommendation Score Calculation:** A **dot product** was applied between user profile vectors and course genre vectors to compute recommendation scores. Higher scores indicate a stronger alignment between the user's learning preferences and the course content.

• **Findings & Insights:** The system successfully recommended relevant courses, with highly engaged users receiving more diverse suggestions. The **cold-start problem** remains a challenge for new users with limited historical data, indicating the potential need for hybrid models.

These results highlight the effectiveness of content-based filtering in personalized learning but also suggest that **combining it with collaborative filtering** could enhance recommendation diversity and accuracy.

Conclusion of Content-Based Recommender System Using Course Similarity

This analysis applied **course similarity metrics** to generate recommendations for users based on their previously enrolled courses. A **course similarity matrix**, computed using **Bag of Words (BoW) features**, was leveraged to identify the most relevant courses for each user.



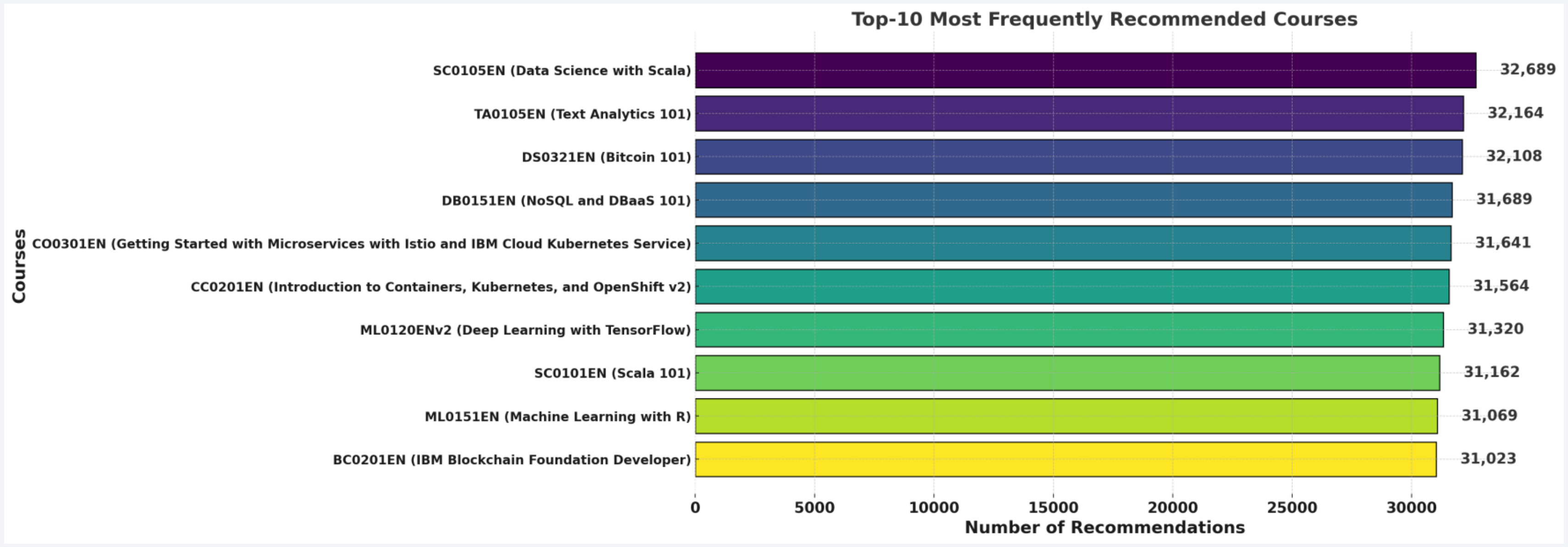
• **Similarity-Based Course Recommendations:** Courses were recommended by calculating similarity scores between a user's enrolled courses and all available courses. A **threshold-based approach** was used to filter out low-relevance suggestions.

• **Findings & Insights:** The system identified notable associations between similar courses based on textual similarity analysis, demonstrating that content-based filtering can help personalize learning pathways. However, **cold-start limitations may** persist for users with minimal course history, suggesting the potential benefits of **hybrid recommendation approaches. Performance & Popularity Trends:** Analysis of the most frequently recommended courses revealed patterns in content similarity, which may influence recommendation trends rather than directly reflecting user preferences.

These results highlight the effectiveness of **content similarity-based recommendations** while also pointing towards the potential benefits of **combining content-based and collaborative filtering** for more diverse and accurate suggestions.

Conclusion of Clustering Based Recommender System

The clustering-based recommender system effectively segmented approximately 34,000 users into 8 distinct learning communities using K-Means clustering on PCA-transformed user profiles. Each community represents users with strongly aligned interests, significantly enhancing recommendation accuracy. On average, approximately **54 new courses** were recommended to each user, tailored precisely to their learning preferences within their clusters. Among recommended courses, "SC0105EN," "TA0105EN," and "DS0321EN" emerged as top choices, reflecting widespread user interest and relevance across clusters.



Comparative testing with different clustering algorithms, such as DBSCAN and hierarchical clustering, showed that cluster cohesion could be different, but K-Means provided clearer interpretability and practical applicability. Future work could explore ensemble clustering strategies and adaptive thresholds for recommendation quantity to further optimize user experience.

Conclusion of Collaborative Filtering Models

The comparative evaluation of collaborative filtering algorithms revealed clear differences in predictive performance and practical applicability. The K-Nearest Neighbors (KNN) and Non-negative Matrix Factorization (NMF) algorithms implemented via the Surprise library yielded similar RMSE scores (approximately 1.29), indicating moderate prediction accuracy but limited in capturing nuanced user-item interactions. Notably, implementing these methods using Scikit-learn resulted in significantly higher RMSE values, reflecting decreased predictive effectiveness due to sparse data handling limitations.

In contrast, the Neural Network Embedding-based recommender significantly outperformed both KNN and NMF, achieving a notably lower RMSE (optimized model: ~0.41). This improvement can be attributed to the neural network's ability to extract deeper latent features that effectively represent user-course interactions, enhancing recommendation precision. Despite higher computational complexity, neural embedding models offered superior predictive performance and scalability, making them particularly suitable for large-scale, sparse datasets.

Future directions may include further refinement of neural network architectures and exploring hybrid models to balance interpretability and accuracy.

Conclusion of Embedding-Based Classfication Models

In this analysis, the performance of five embedding-based classification algorithms, namely Logistic Regression, Decision Tree, Support Vector Classifier (SVC), Bagging Classifier, and Gradient Boosting Classifier (GBC), was comprehensively evaluated to predict categorical user-course interaction modes. The results revealed consistent accuracy across most algorithms (~0.95), while their discriminatory power, measured by the Area Under Curve (AUC), varied considerably. The Bagging classifier, which ensembles SVC models, shows similar performance with an AUC of 0.50, indicating no significant improvement in class separation.

Although Logistic Regression, SVC, and Bagging Classifier models achieved comparably high accuracy values (~0.95), their AUC scores were significantly lower, ranging between 0.47 and 0.63. This discrepancy highlights a critical limitation: despite correct classification rates being high, these models struggled to reliably differentiate between interaction categories, likely due to imbalanced data distributions or limited capturing of subtle interaction patterns.

The Decision Tree Classifier demonstrated improved discriminative ability, with an AUC of 0.87, indicating moderate capability in distinguishing user-item interaction categories. The inherent interpretability of Decision Trees further adds practical value, enabling clearer insights into model decisions and facilitating easier communication of results.

Most significantly, the Gradient Boosting Classifier emerged as the top-performing model, with exceptional accuracy (0.98) and the highest discriminatory capability (AUC = 0.93). This outcome suggests that GBC may effectively utilize patterns present within the embedding features, potentially capturing intricate user-item interactions and enhancing predictive performance. Its robust performance validates embedding vectors' capability to encode meaningful interaction patterns, particularly benefiting predictive tasks involving complex latent relationships.

Conclusively, the Gradient Boosting Classifier demonstrated the best performance, combining high accuracy with strong discriminatory power. Nevertheless, the Decision Tree remains a valuable alternative, especially when interpretability and transparency are crucial. Future research directions might consider hybrid or ensemble models to integrate the interpretability strengths of simpler methods with the predictive power demonstrated by Gradient Boosting.

Future Directions and Recommendations

The analyses conducted throughout this study demonstrated substantial success in employing various machine learning methodologies, including content-based filtering, collaborative filtering, clustering techniques, and neural network-based embeddings, to generate personalized course recommendations. Although impressive predictive performances were achieved, multiple opportunities remain available to further refine and enhance the capabilities of the recommender system.

Initially, addressing the popularity bias identified during Exploratory Data Analysis (EDA) could significantly enhance the diversity of recommendations. Employing approaches such as re-ranking algorithms, personalized normalization techniques, or incorporating diversity-aware metrics into recommendation evaluations would help mitigate the dominance of highly enrolled courses. Additionally, fairness-aware recommendation methods can be integrated to promote a more balanced exposure of niche yet valuable courses.

In the context of content-based recommender systems, combining traditional TF-IDF vectorization with advanced semantic embedding approaches such as Word2Vec, GloVe, or Transformer-based embeddings (e.g., BERT, Sentence-BERT) would markedly improve the semantic accuracy and relevance of suggested courses. Implementing adaptive threshold tuning methodologies to dynamically adjust the balance between the number and quality of recommendations is also advisable to maintain continuous personalization aligned with individual user engagement levels.

For clustering-based recommendation methods, ensemble clustering strategies integrating multiple algorithms such as K-Means, hierarchical clustering, and DBSCAN would leverage the complementary strengths of these techniques, potentially leading to superior clustering quality, interpretability, and user segmentation accuracy. Furthermore, temporal clustering approaches could be explored to analyze changes in user preferences over time, thereby capturing evolving learning interests and adapting recommendations accordingly.

The collaborative filtering analyses clearly illustrated the advantage of neural network embedding models in capturing complex patterns within user-course interactions. Given these results, future work could focus on further refining neural network architectures by exploring advanced embedding methods, attention mechanisms, or leveraging transformer-based models and deep learning architectures optimized for large-scale sparse data. Additionally, adopting online learning methods capable of continuous model updates based on real-time user interactions would significantly enhance recommendation responsiveness and accuracy.

Finally, the classification analyses underscored the superior predictive performance of Gradient Boosting Classifiers. Nonetheless, to address practical concerns related to interpretability and transparency, hybrid modeling frameworks combining Gradient Boosting's robust predictive power with inherently interpretable models such as Decision Trees or Logistic Regression could be beneficial. Employing explainability frameworks, such as SHAP and LIME, would further strengthen the interpretability of these hybrid models, thus ensuring balanced predictive accuracy and transparency for end-users.

By pursuing these proposed enhancements, subsequent iterations of the recommender system can achieve more diverse, precise, interpretable, and contextually meaningful course recommendations, significantly improving learner experience and educational outcomes.

Appendix

- **GitHub Repository:** https://github.com/Cagla-Caglar/Personalized-Online-Course-Recommender-System-with-Machine-Learning
- **Streamlit App:** https://personalized-online-course-recommender-system.streamlit.app/