Build a Personalized Online Course Recommender System with Machine Learning

N Hartley
20 April 2024

Outline

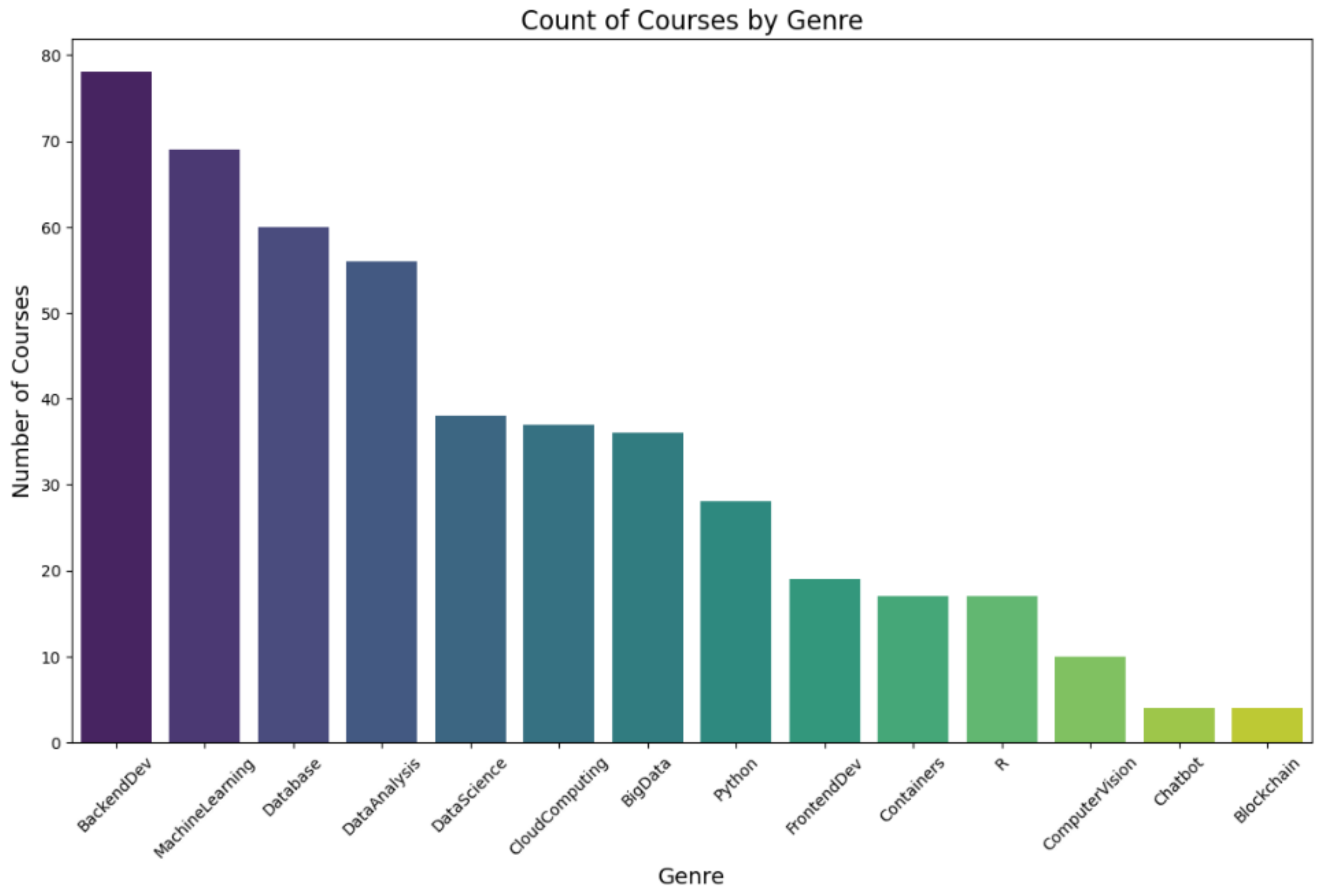
Introduction

- The goal of this project is to develop a variety of recommender systems which apply different approaches to identifying similarities amongst students and academic courses to make useful recommendations to the student about other courses which are relevant when compared to the courses they are currently enrolled in
- Student information and course information is in several different data frames which need to be cleaned, explored and (in some cases) merged to derive meaningful suggestion to students
- The analysis was conducted using the Python language in Jupyter notebooks which were run in Google Colab to enhance processing efficiency and minimize the risk of kernel faults when using the course lab architecture. The corresponding notebooks have been saved on GitHub and can be accessed via links in the appendix in this document

Exploratory Data Analysis

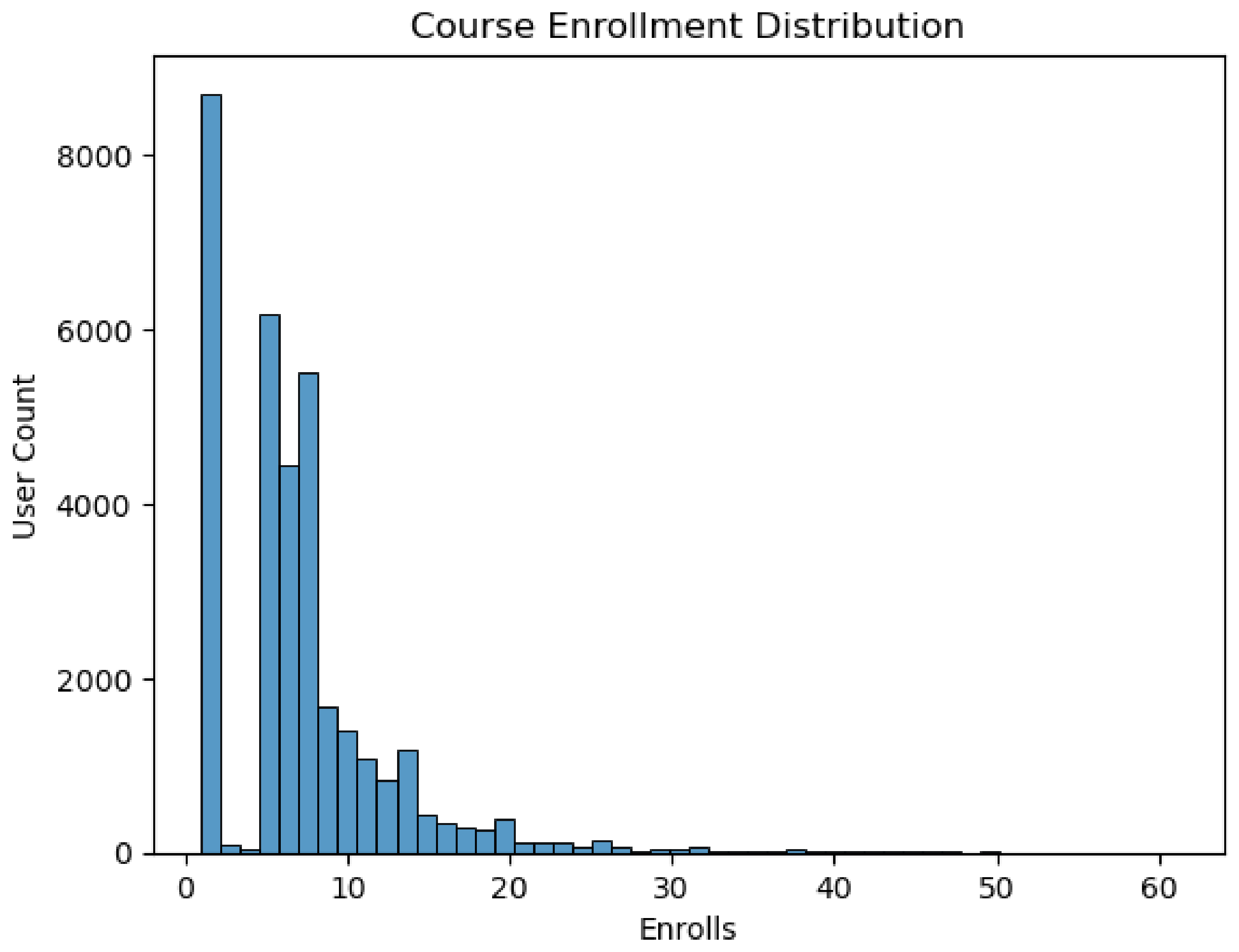
Course counts per genre

- Course information can be found in several different csv files as provided over the course of the final project
- My analysis indicates that back-end development and machine learning were the most popular areas
- The least popular genres represent only a small fraction of the most popular courses, so intuitively I would expect to see these being suggested less frequently when the recommender systems are built



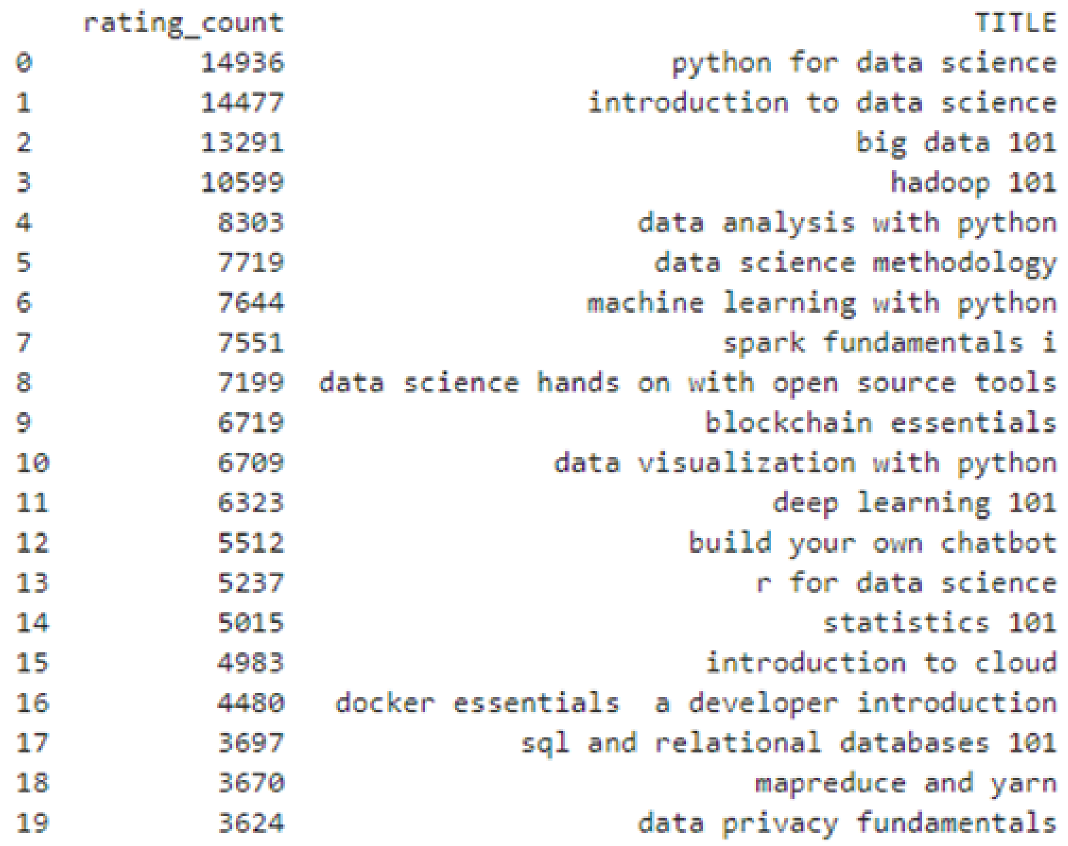
Course enrollment distribution

- A histogram of the enrollment distribution indicates a similar theme, with a small number of classes having really high enrollment
- The distribution is skewed right, indicating a long tail of courses which have relevance amongst the students
- Although outside of the scope of this project it could be interesting to consider the causality of the distribution



20 most popular courses

- The top 20 courses are listed in the table to the right
- Four classes have ratings counts of 10,000 or more
- We may want to consider the merits of the weightings for courses which have a lower ratings count in comparison with more frequently rated ones, as the smaller sample sizes may be influenced by a handful of outliers
- We should consider how we treat ratings of “0”. Are these rated lowly or simply without a rating?

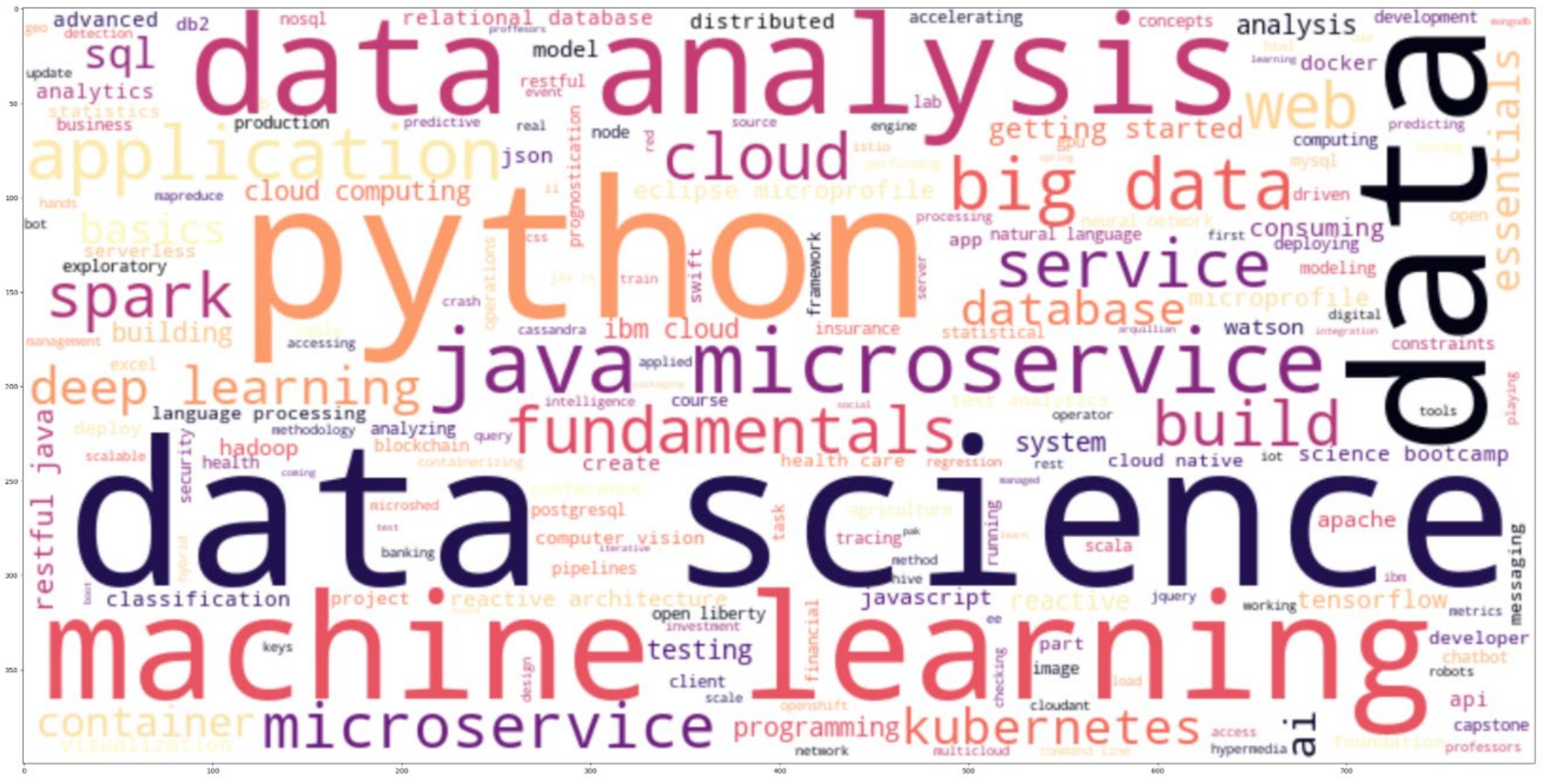


```
    rating_count                                   TITLE
0          14936                 python for data science
1          14477            introduction to data science
2          13291                            big data 101
3          10599                              hadoop 101
4           8303               data analysis with python
5           7719               data science methodology
6           7644              machine learning with python
7           7551                      spark fundamentals i
8           7199  data science hands on with open source tools
9           6719                    blockchain essentials
10          6709            data visualization with python
11          6323                        deep learning 101
12          5512                     build your own chatbot
13          5237                       r for data science
14          5015                           statistics 101
15          4983                    introduction to cloud
16          4480   docker essentials  a developer introduction
17          3697           sql and relational databases 101
18          3670                       mapreduce and yarn
19          3624                 data privacy fundamentals
```

Word cloud of course titles

- The data cloud derived from the source data supports more quantitative assessments but makes things easier to visualize

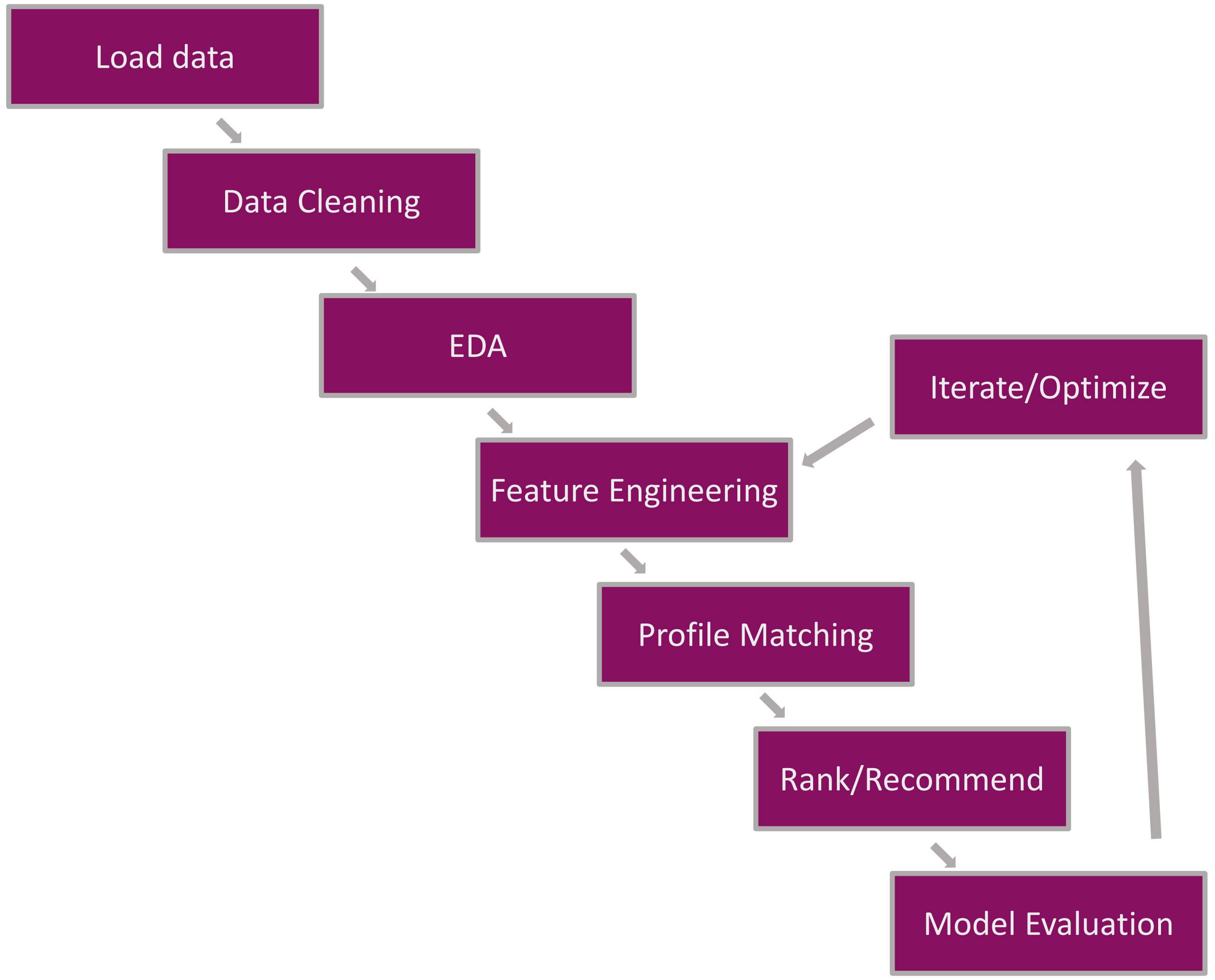
- Data Science, Data Analysis, Python and Machine Learning seem to have substantially more interest than other topics



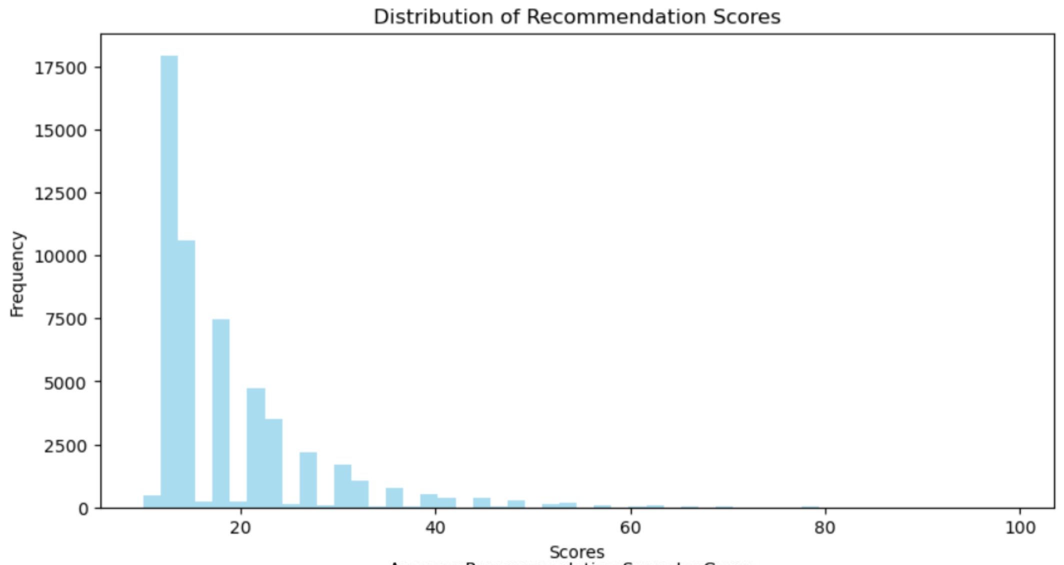
Content-based Recommender System using Unsupervised Learning

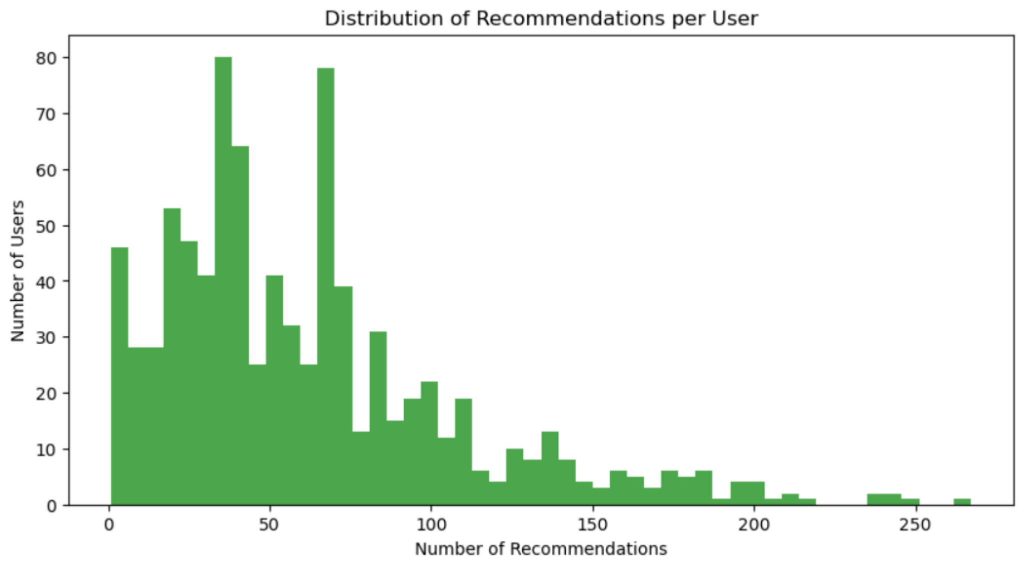
Flowchart of content-based recommender system using user profile and course genres

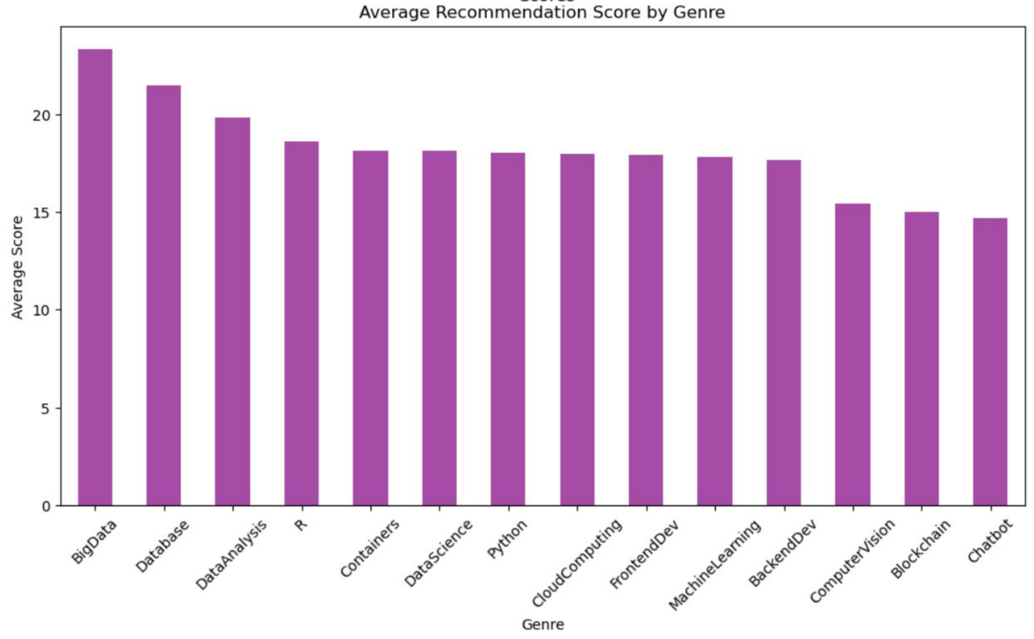
1. **Load Data:** Import user profiles and course genre data, ensuring proper parsing of user identifiers, course identifiers, and genres
2. **Data Cleaning:** Check for and handle missing or inconsistent data entries within both user profiles and course genres
3. **Exploratory Data Analysis (EDA):** Analyze the distribution of user profile features. Examine the variety and frequency of course genres. Identify any notable patterns or correlations
4. **Feature Engineering:** Convert course genres into a machine-readable format (e.g., one-hot encoding). Normalize or scale user profile features as necessary
5. **Profile and Genre Matching:** Compute similarity scores between user profiles and course attributes. Use techniques like cosine similarity for multi-dimensional feature matching
6. **Ranking and Recommendations:** Rank courses for each user based on similarity scores. Select top-N courses to recommend to each user based on their profile
7. **Model Evaluation:** Split the data into training and testing sets to validate the effectiveness of the recommendations. Use metrics like precision, recall, or F1 score to evaluate performance
8. **Iteration and Optimization:** Iterate on the model by tuning hyperparameters, feature selection, and similarity thresholds. Optimize the system based on feedback and performance metrics



Evaluation results of user profile-based recommender system



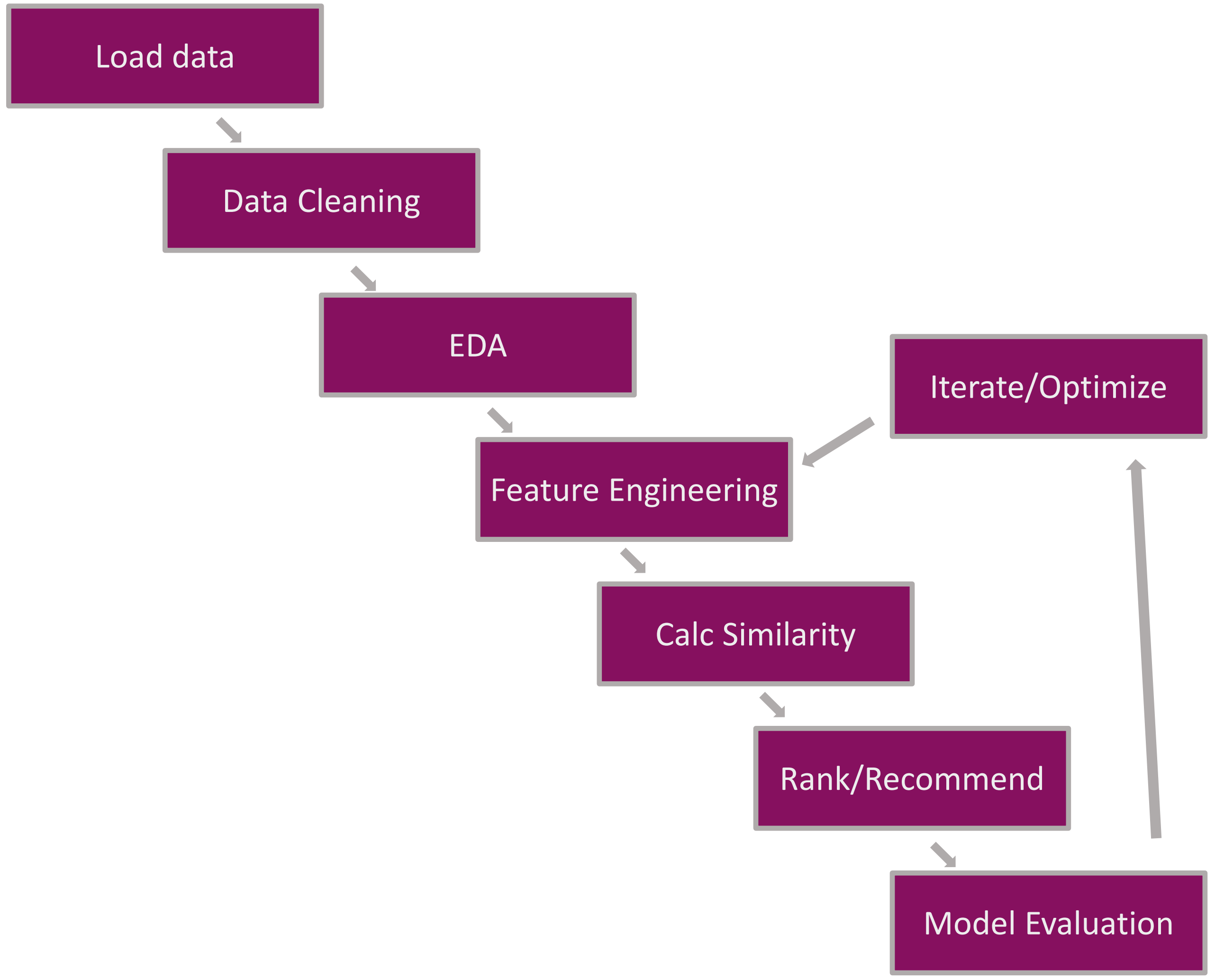




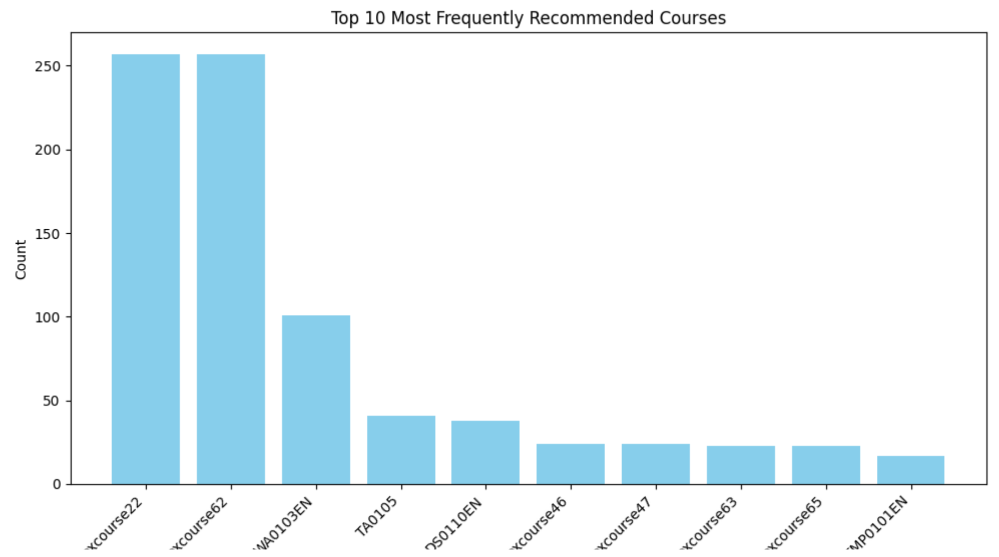
```
Top 10 Most Frequently Recommended Courses:
                                          TITLE  Recommendation_Count
0                         text analytics at scale                   608
1  data science in insurance  basic statistical a...                548
2             introduction to data science in python                547
3                 applied machine learning in python                547
4                  accelerating deep learning with gpu              544
5  performing database operations in the cloudant...               533
6                             sql for data science                  533
7            sql for data science capstone project                  533
8  cloud computing applications  part 2  big data...                524
9                         analyzing big data with sql               516
```

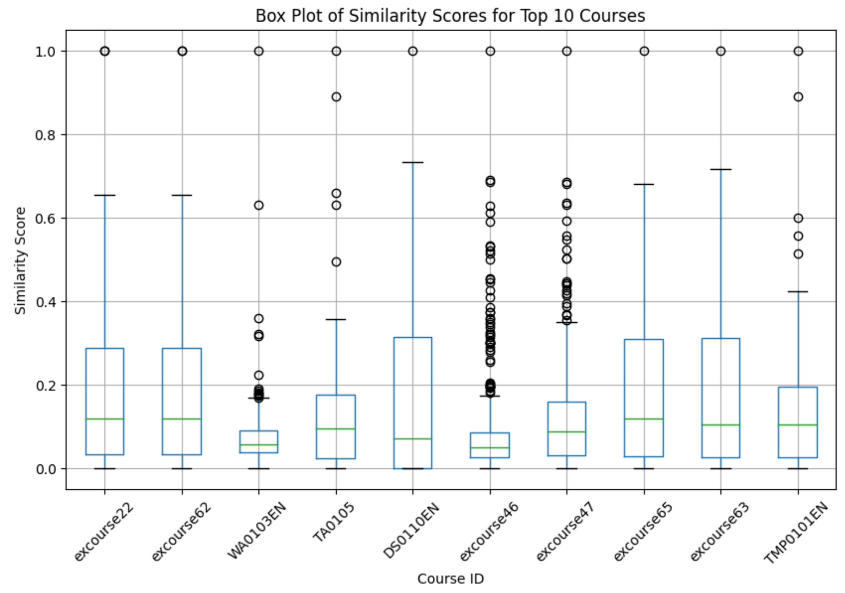
Flowchart of content-based recommender system using course similarity

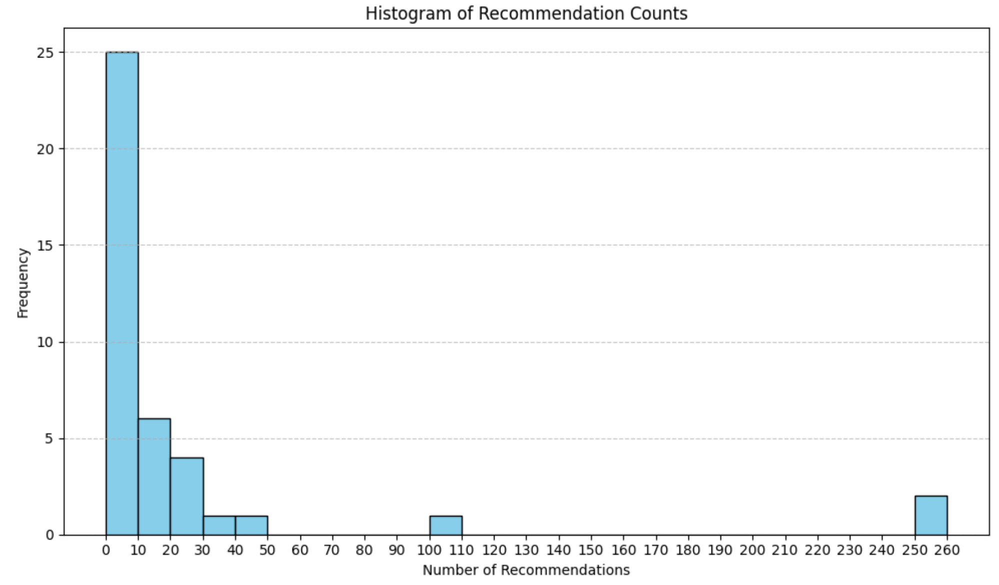
1. **Load Data:** Import course data, ensuring course identifiers, titles, descriptions, and genre information etc
2. **Data Cleaning:** Check for inconsistencies or missing values in the course descriptions and genres
3. **Exploratory Data Analysis (EDA):** Assess the distribution of courses across different genres. Analyze course descriptions to understand the depth and diversity of content
4. **Feature Extraction:** Extract features from course titles and descriptions using techniques such as TF-IDF or word embeddings. Optionally categorize courses by genres and use this categorization as an additional feature
5. **Calculate Similarity:** Calculate similarity scores between courses based on their extracted features using cosine similarity or other relevant metrics
6. **Ranking & Recommendations**: For a given course, identify and rank other courses based on their similarity scores. Recommend courses with the highest similarity scores that the user has not yet engaged with
7. **Model Evaluation**: Use a metric like Mean Squared Error (MSE) or precision at k to evaluate the effectiveness of the recommendations. Validate the recommendations with user feedback or through A/B testing
8. **Iteration & Optimization:** Adjust feature extraction methods and similarity thresholds based on evaluation outcome. Continuously refine the recommendation engine to improve relevance and personalization

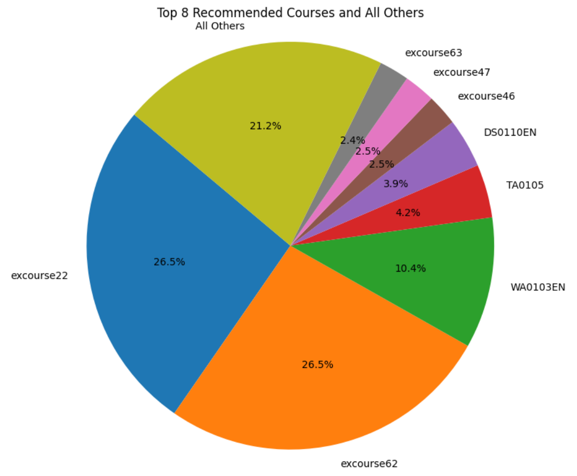


Evaluation results of similarity-based recommender system



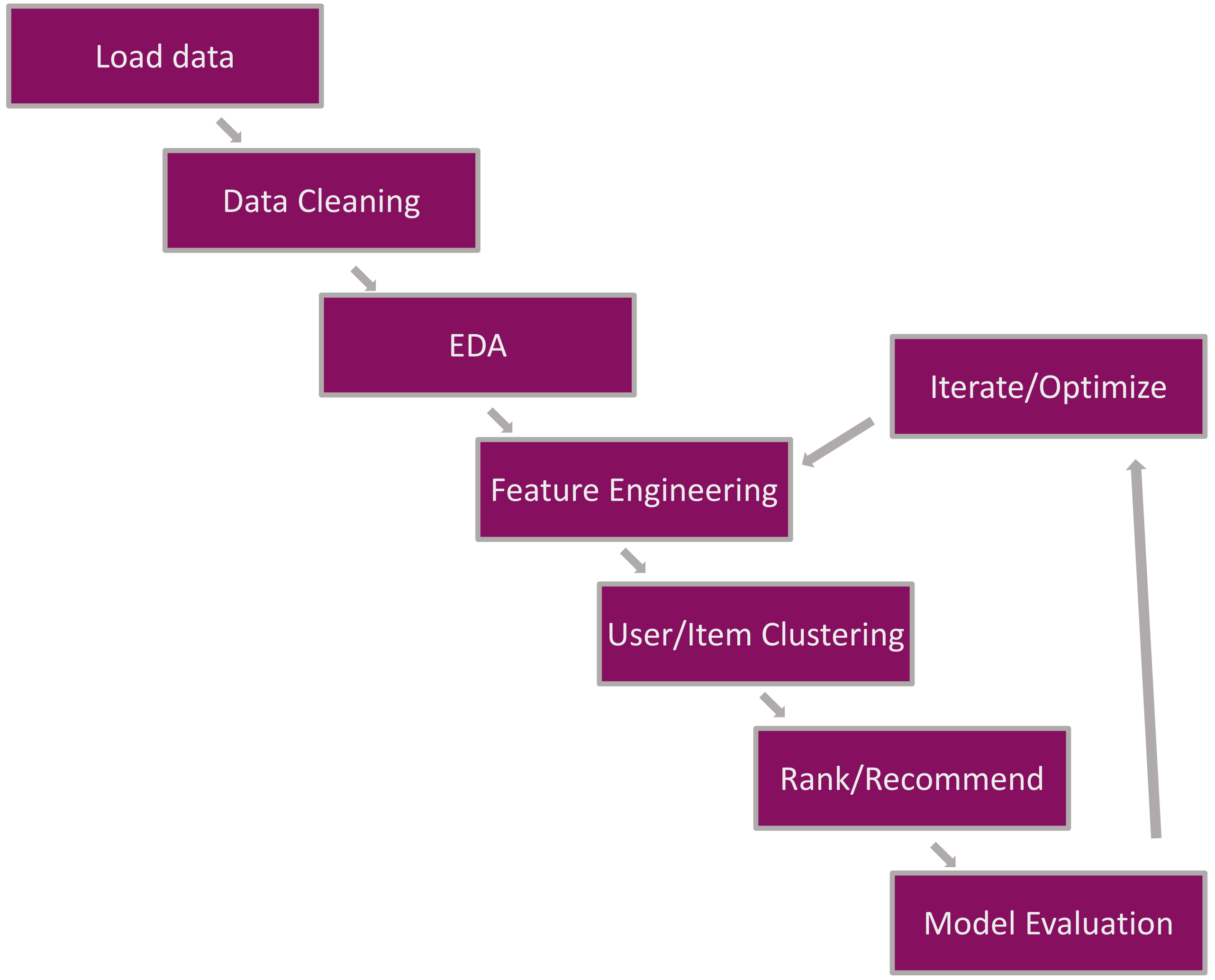




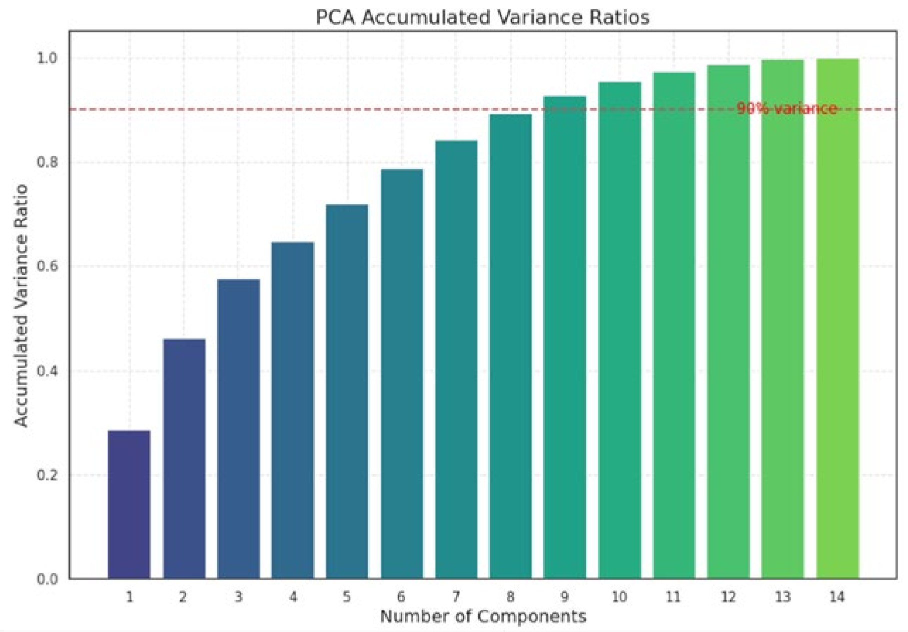


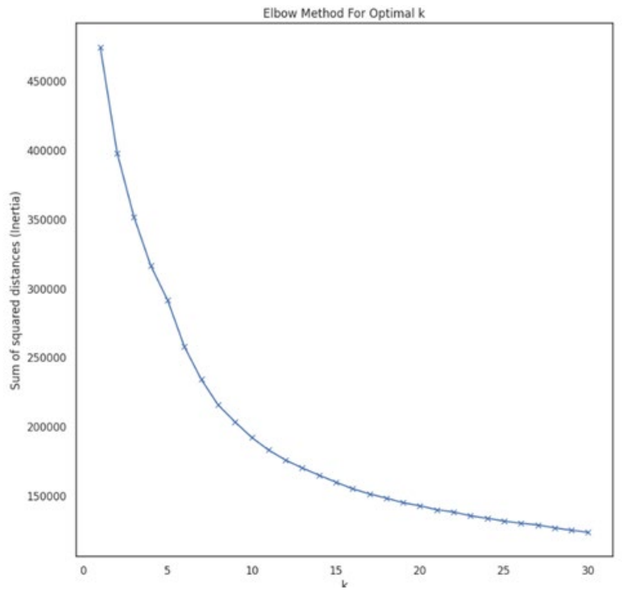
Flowchart of clustering-based recommender system

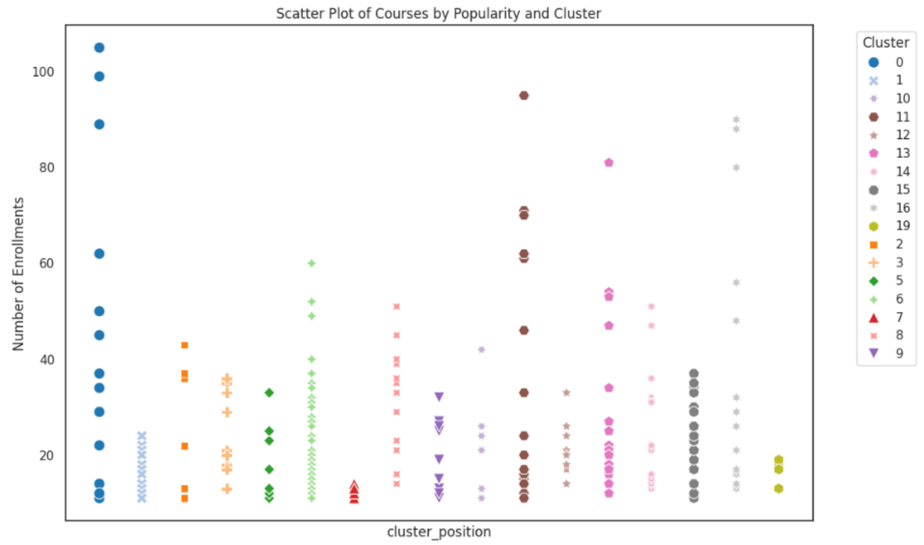
1. **Load Data:** Import course data, ensuring course identifiers, titles, descriptions, and genre information etc
2. **Data Cleaning:** Check for inconsistencies or missing values in the course descriptions and genres
3. **Exploratory Data Analysis (EDA):** Assess the distribution of courses across different genres. Analyze course descriptions to understand the depth and diversity of content
4. **Feature Extraction:** Extract features from course titles and descriptions using techniques such as TF-IDF or word embeddings. Optionally categorize courses by genres and use this categorization as an additional feature
5. **User/Item Clustering:** Apply a clustering algorithm such as K-Means, hierarchical clustering, or DBSCAN to group users or courses based on selected features. Determine the optimal number of clusters through methods such as the elbow method or silhouette analysis
6. **Ranking & Recommendations**: For each user, recommend courses from the user's cluster that the user has not interacted with yet. Alternatively, if clustering courses, recommend other courses within the same cluster as the courses a user likes or interacts with
7. **Model Evaluation**: Evaluate the quality of clusters and the recommendation system using metrics like cohesion, separation, and silhouette score. Validate the recommendations against a test set
8. **Iteration & Optimization:** Refine clustering by exploring different algorithms, number of clusters, or feature sets



Evaluation results of clustering-based recommender system





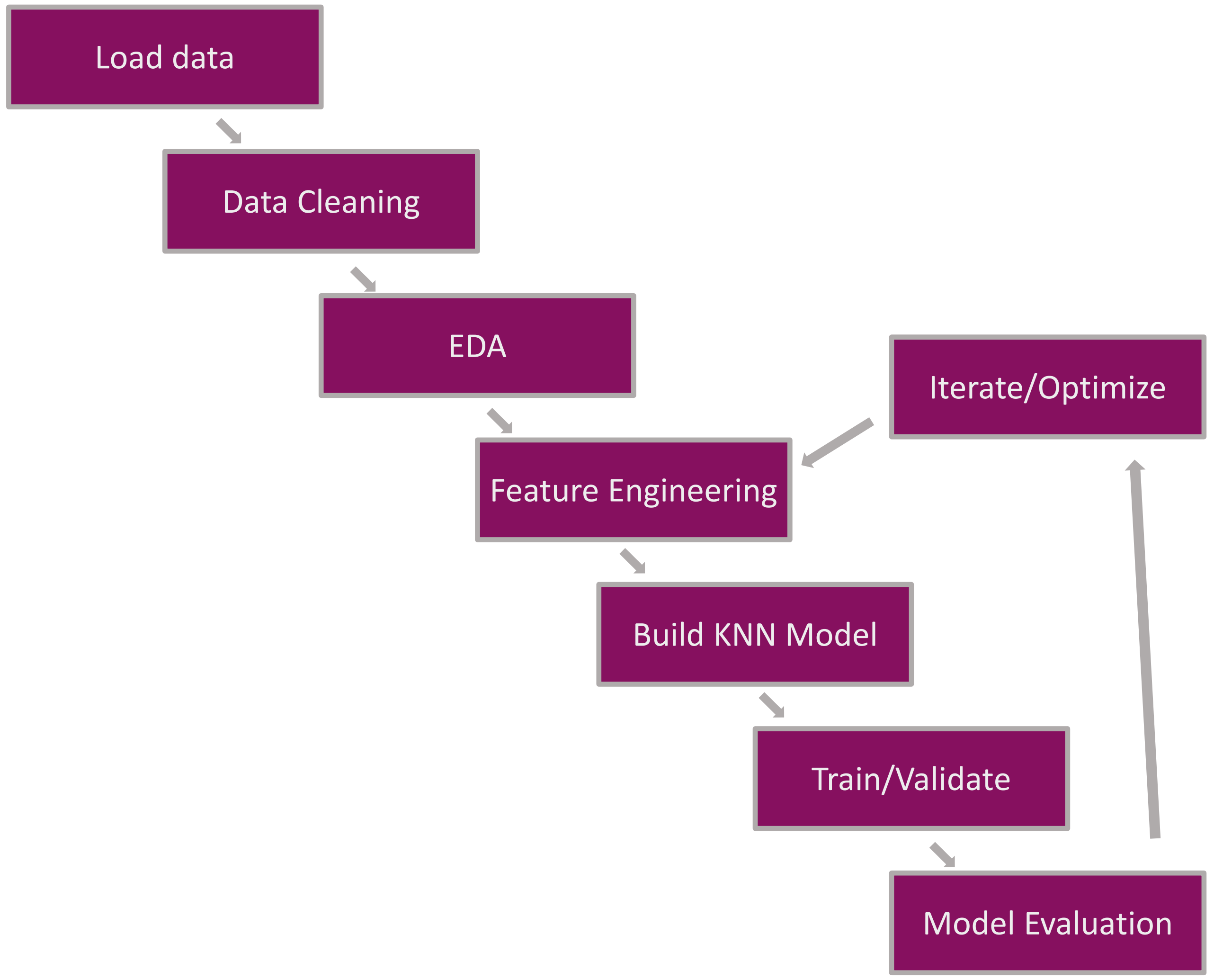


```
  variable   course
0    User1   course1
1    User1   course2
3    User2   course3
4    User2   course4
6    User3   course3
7    User3   course4
8    User3   course5
Average number of recommended courses per user: 2.33
Top-10 most frequently recommended courses:
course
course3    2
course4    2
course1    1
course2    1
course5    1
```

Collaborative-filtering Recommender System using Supervised Learning

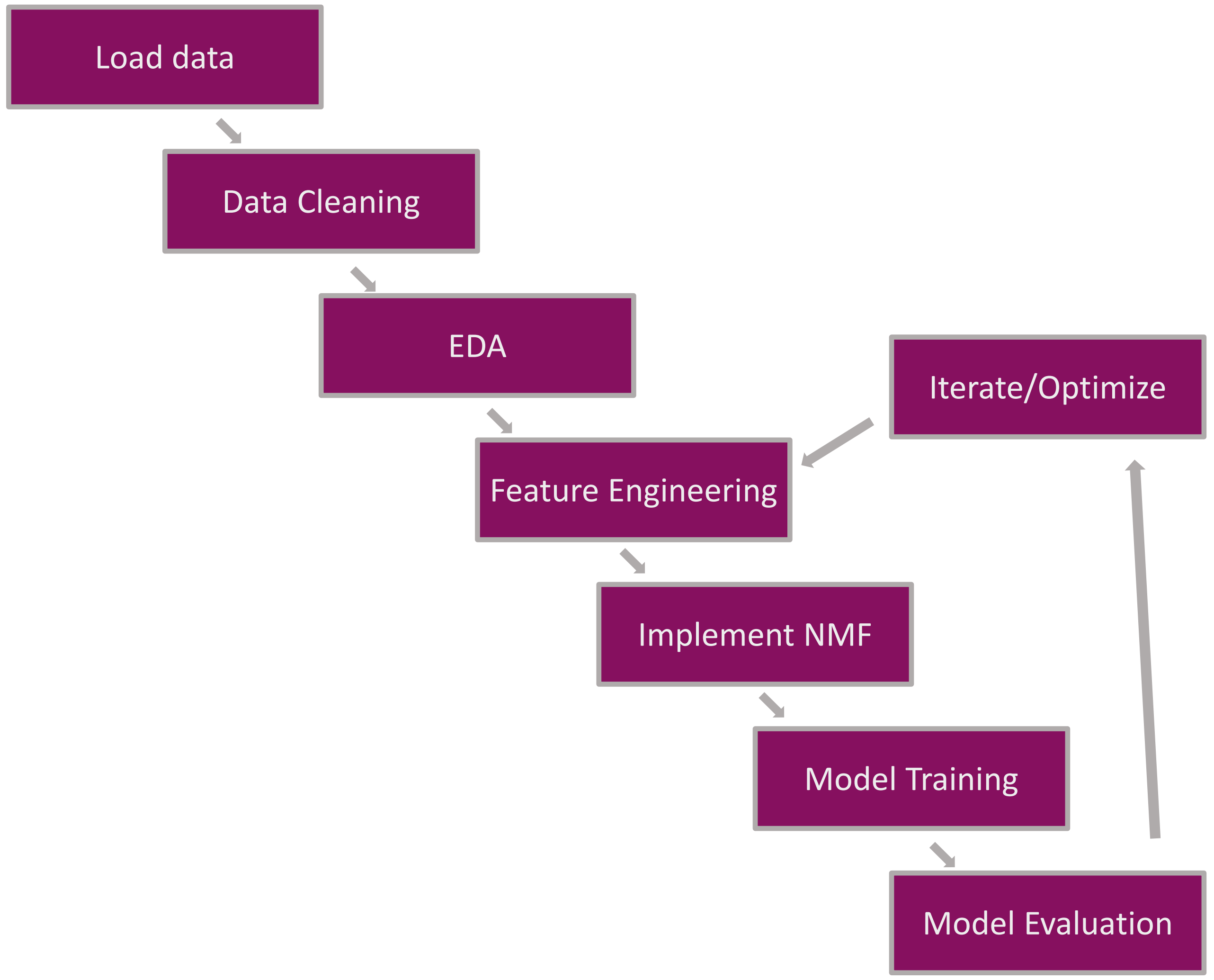
Flowchart of KNN-based recommender system

1. **Load Data:** Import the user-item interaction data, ensuring user and item identifiers are properly loaded along with the interaction ratings or binary interaction flags
2. **Data Cleaning:** Prepare the interaction data by filling missing values, normalizing ratings, and encoding categorical variables if necessary
3. **Exploratory Data Analysis (EDA):** Analyze the distribution of user interactions and item ratings. Identify any outliers or patterns that might affect the similarity calculations
4. **Feature Engineering:** Normalize or standardize features to ensure they're on the same scale
5. **Build the KNN Model:** Use a library like scikit-surprise to implement the KNN algorithm. Choose a similarity metric (cosine, Pearson, etc.) and define the number of neighbors k
6. **Train and Validate the Mode:** Split the data into training and testing sets. Train the KNN model on the training set and validate it on the test set
7. **Model Evaluation:** Use metrics such as RMSE (Root Mean Square Error) to evaluate the model's accuracy. Consider additional metrics like precision at k, recall, or F1 score, especially if the interaction data is binary
8. **Iteration and Optimization:** Fine-tune the model by adjusting k, trying different similarity metrics, and considering user or item biases



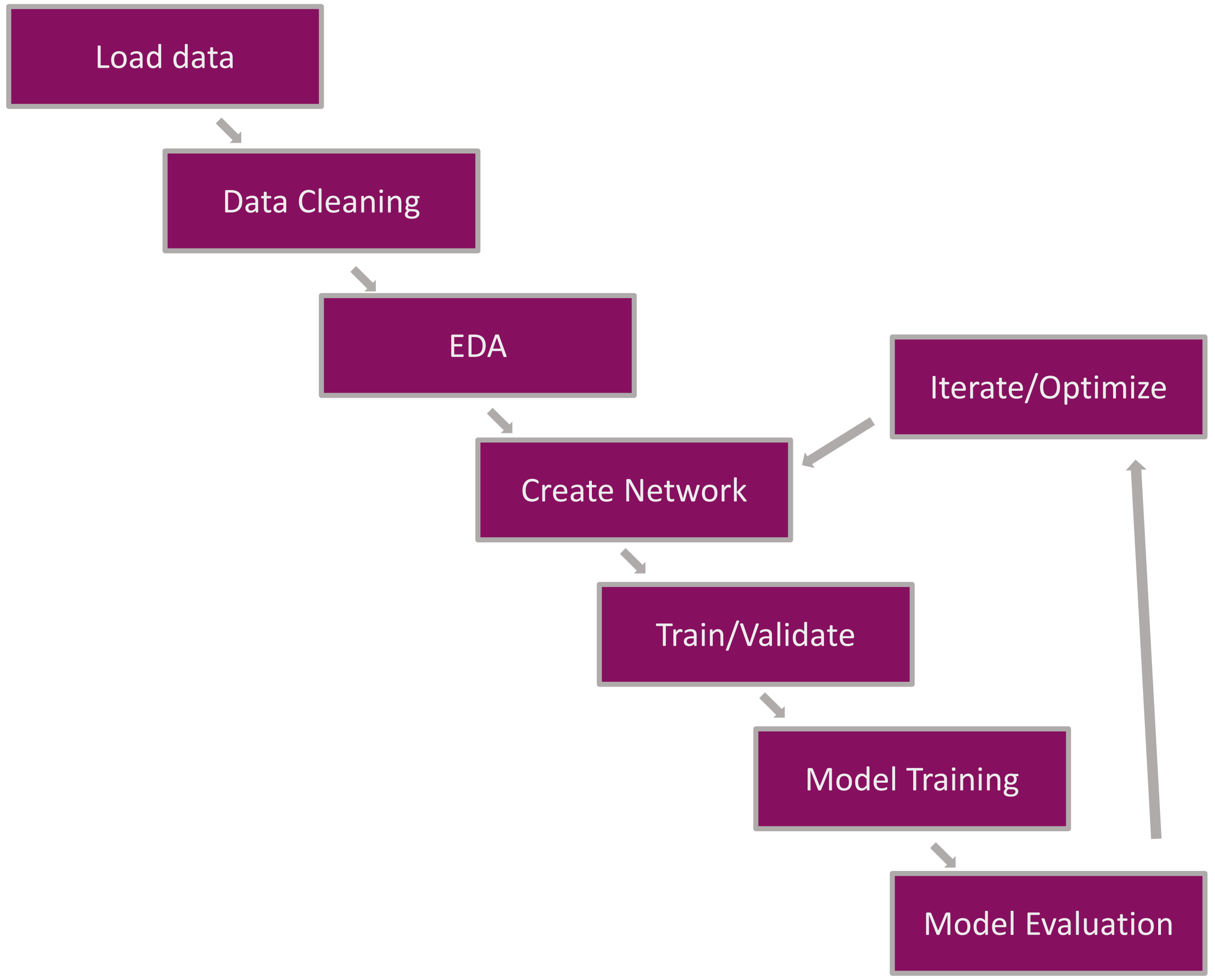
Flowchart of NMF-based recommender system

1. **Load Data:** Import user profiles and course genre data, ensuring proper parsing of user identifiers, course identifiers, and genres
2. **Data Cleaning:** Check for and handle missing or inconsistent data entries within both user profiles and course genres
3. **Exploratory Data Analysis (EDA):** Analyze the distribution of user profile features. Examine the variety and frequency of course genres. Identify any notable patterns or correlations
4. **Feature Engineering:** For an NMF-based approach, explicit features are not typically used, as NMF itself discovers latent features. However, any auxiliary user or item information can be incorporated to enhance the model. Standardize additional features to ensure compatibility with the interaction matrix
5. **Implement NMF:** Utilize matrix factorization libraries such as scikit to perform NMF, decomposing the interaction matrix into user and item matrices. Choose the number of components (latent features) based on model performance and interpretability
6. **Model Training & Dimensionalize:** Train the NMF model to factorize the user-item interaction matrix into lower-dimensional representations. Reconstruct the interaction predictions from the factorized matrices.
7. **Model Evaluation:** Evaluate the reconstructed matrix against the actual ratings using error metrics like RMSE. Perform cross-validation to assess the model's generalizability
8. **Iteration and Optimization:** Fine-tune the number of components and regularization parameters to optimize performance



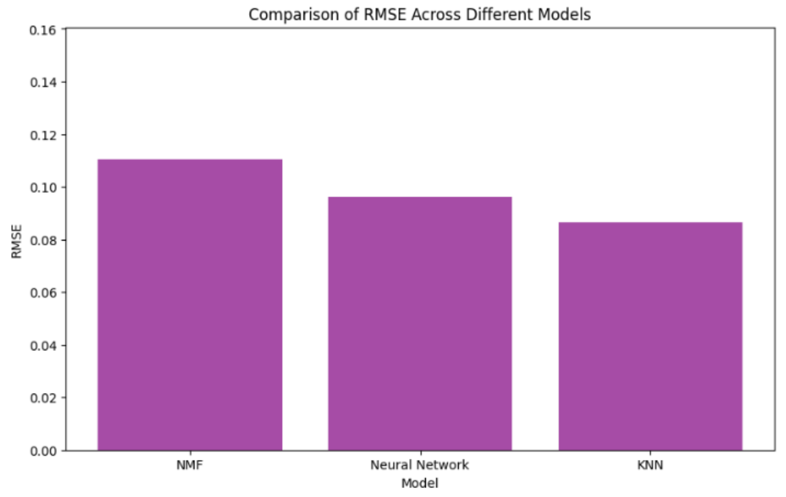
Flowchart of Neural Network Embedding-based recommender system

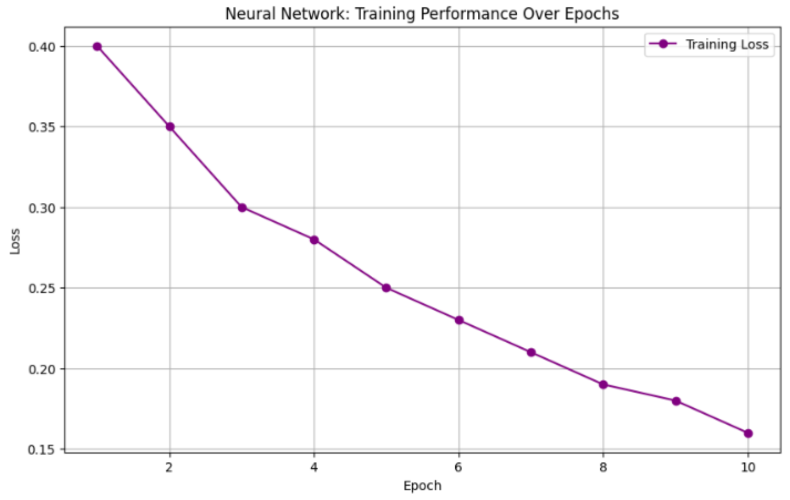
1. **Load Data:** Import user profiles and course genre data, ensuring proper parsing of user identifiers, course identifiers, and genres
2. **Data Cleaning:** Check for and handle missing or inconsistent data entries within both user profiles and course genres
3. **Exploratory Data Analysis (EDA):** Analyze the distribution of user profile features. Examine the variety and frequency of course genres. Identify any notable patterns or correlations
4. **Create Network:** Design the neural network with embedding layers for users and items to capture latent factors in lower-dimensional spaces. Add additional layers and nonlinearities to model complex user-item interactions
5. **Training & Validation:** Split the data into training, validation, and testing sets. Use techniques like mini-batch gradient descent and dropout for effective training and to prevent overfitting
6. **Model Training:** Train the neural network on the user-item interactions, using embeddings to predict user ratings or preferences. Monitor the loss function and accuracy on the validation set to guide the training process
7. **Model Evaluation:** After training, predict on the test set and compute the RMSE to quantify the prediction error of the model. Evaluate additional metrics like Mean Absolute Error (MAE) or AUC for classification tasks
8. **Iteration and Optimization:** Refine the model by adjusting hyperparameters, network layers, and the learning rate based on test set performance



Compare the performance of collaborative-filtering models

- KNN-based (K-Nearest Neighbors): This method will look at similar users (neighbors) based on their course preferences and suggest courses those similar users liked
- NMF-based (Non-negative Matrix Factorization): This approach decomposes the user-item interaction matrix into a user-feature and item-feature matrix, then uses these reduced dimensions to predict a user's preference for unenrolled courses
- Neural Network Embedding-based: This method involves training a neural network to learn a low-dimensional representation (embedding) of both users and courses, which can be used to predict user preferences and make recommendations
- Based on the analysis I conducted, all of them were pretty effective, with KNN yielding the best results





Conclusions

- Running the jupyter notebooks in the normal lab settings was quite difficult as the kernel often failed during calculations. Due to this I transferred my efforts over to Google Colab to run the notebooks in a more stable setting
- I was able to successfully mimic the samples provided as examples in all cases
- If the data is successfully scrubbed prior to reading the files into the notebooks, the actual calculations are quite swift and the results are aligned with the expected outcomes
- The collaborative filtering models seemed more intuitive and efficient to me. When running the models and comparing them on the basis of root mean squared error (RMSE) indicated that on an absolute basis collaborative filtering was an effective methodology, with K Nearest Neighbors having a modest edge over the others, but perhaps not significant in the statistical sense

Appendix

Associated Github Repo Links:

- EDA: https://github.com/NH-Davis/IBM-Machine-Learning-Capstone/blob/main/EDA_Final.ipynb
- User-Based Recommendation System: https://github.com/NH-Davis/IBM-Machine-Learning-Capstone/blob/main/FINAL_user_profile_based_recommender_system.ipynb
- Content-Based Recommendation System: https://github.com/NH-Davis/IBM-Machine-Learning-Capstone/blob/main/Content_Based_Recommender_System_v-Final.ipynb
- Clustering-Based Recommendation System: https://github.com/NH-Davis/IBM-Machine-Learning-Capstone/blob/main/Clustering_Based_Recommender_System_Final.ipynb
- Collaborative Filtering Recommendation System(s): https://github.com/NH-Davis/IBM-Machine-Learning-Capstone/blob/main/Collaborative_Filtering_Based_Recommender_Systems_Final.ipynb