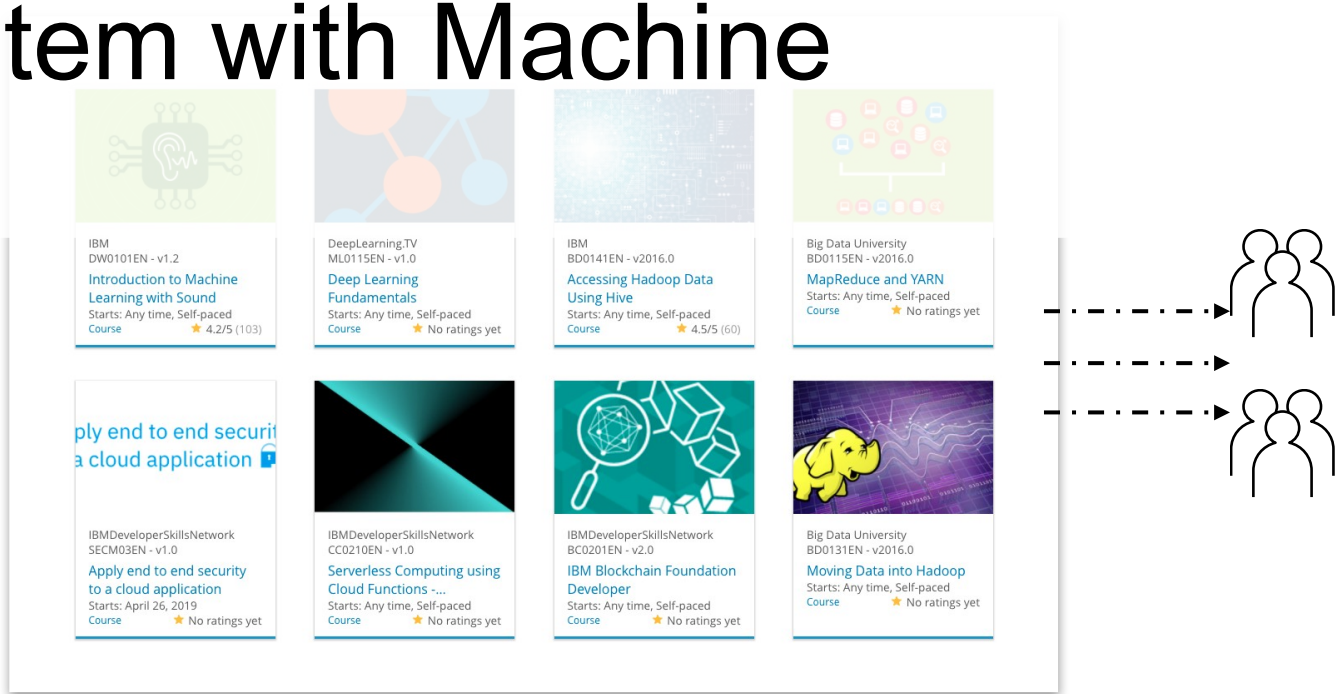
Sachin Rathi
31/8/2024

Build a Personalized Online Course Recommender System with Machine Learning



Outline

- Introduction and Background
- Exploratory Data Analysis
- Content-based Recommender System using Unsupervised Learning
- Collaborative-filtering based Recommender System using Supervised learning
- Conclusion
- Appendix

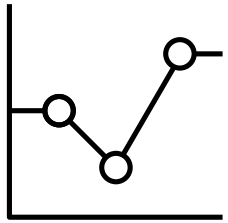
Introduction

- Project background and context
- Problem states and hypotheses

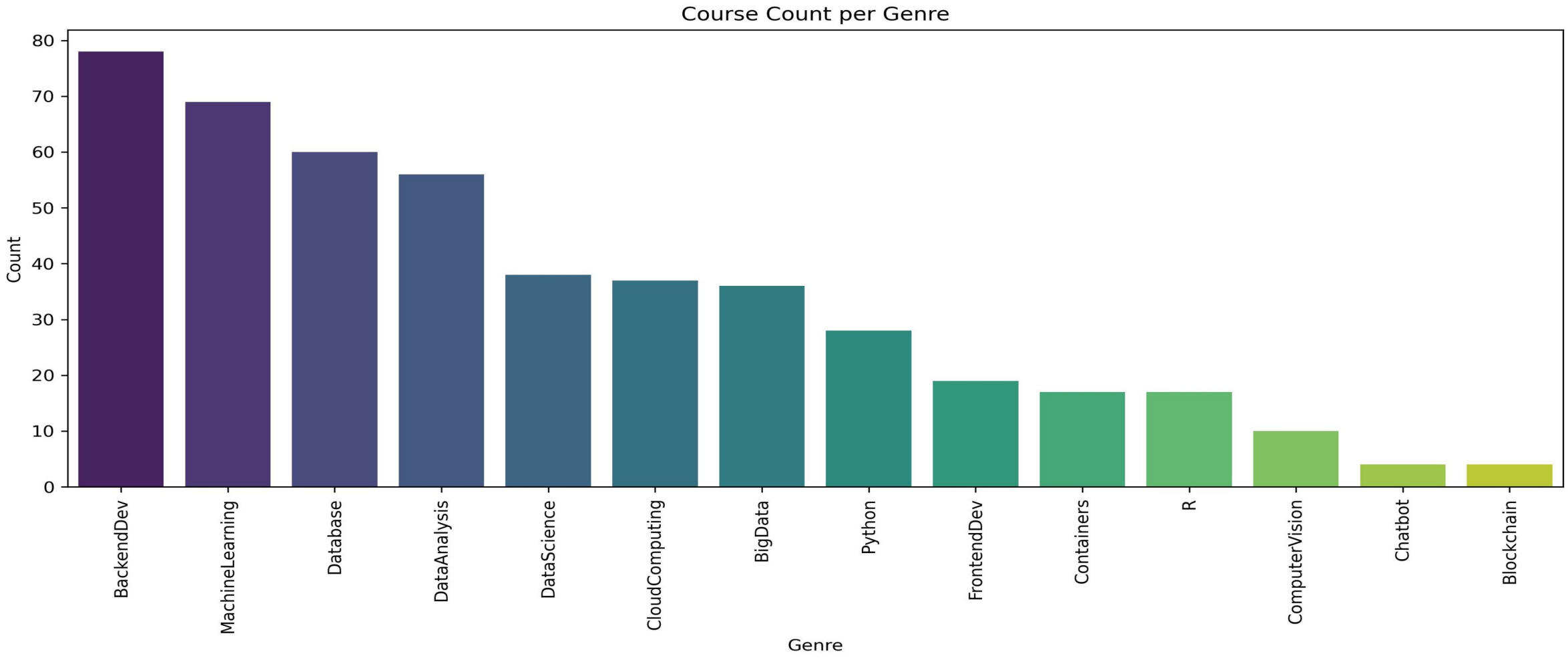
Instructions for learner:

- Describe the project background and context
- Describe the problem states and write the hypotheses

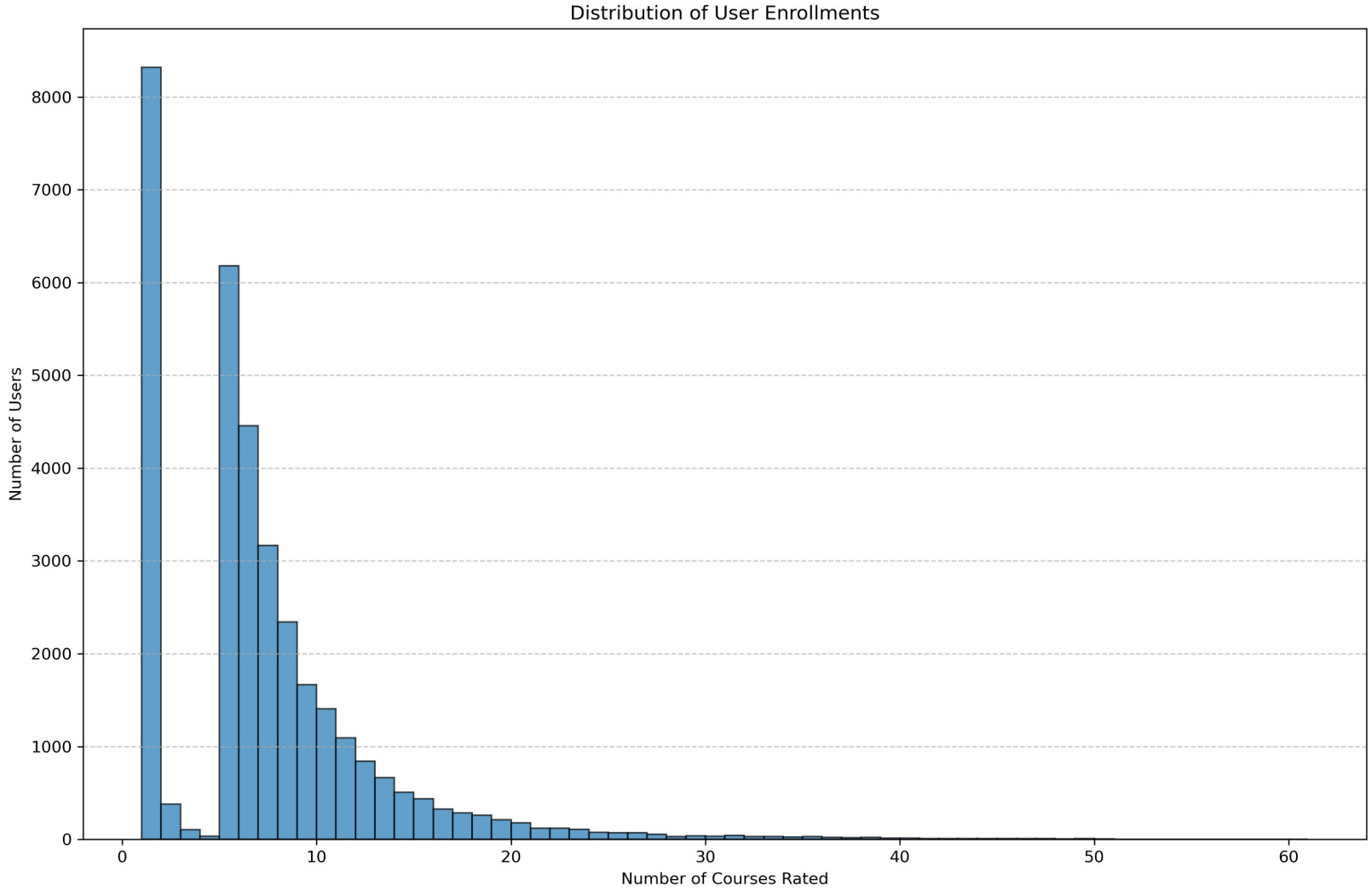
Exploratory Data Analysis



Course counts per genre



Course enrollment distribution

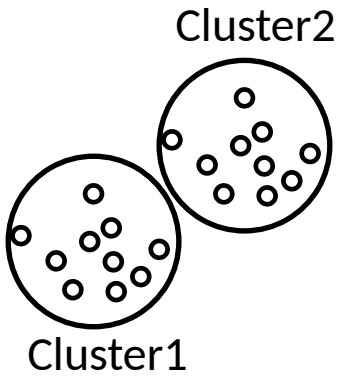


20 most popular courses

	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

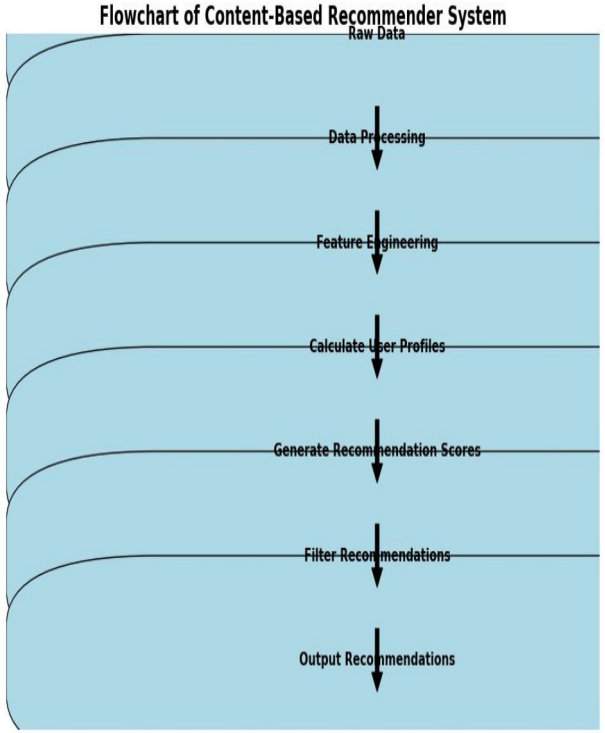
Word cloud of course titles

Content-based Recommender System using Unsupervised Learning

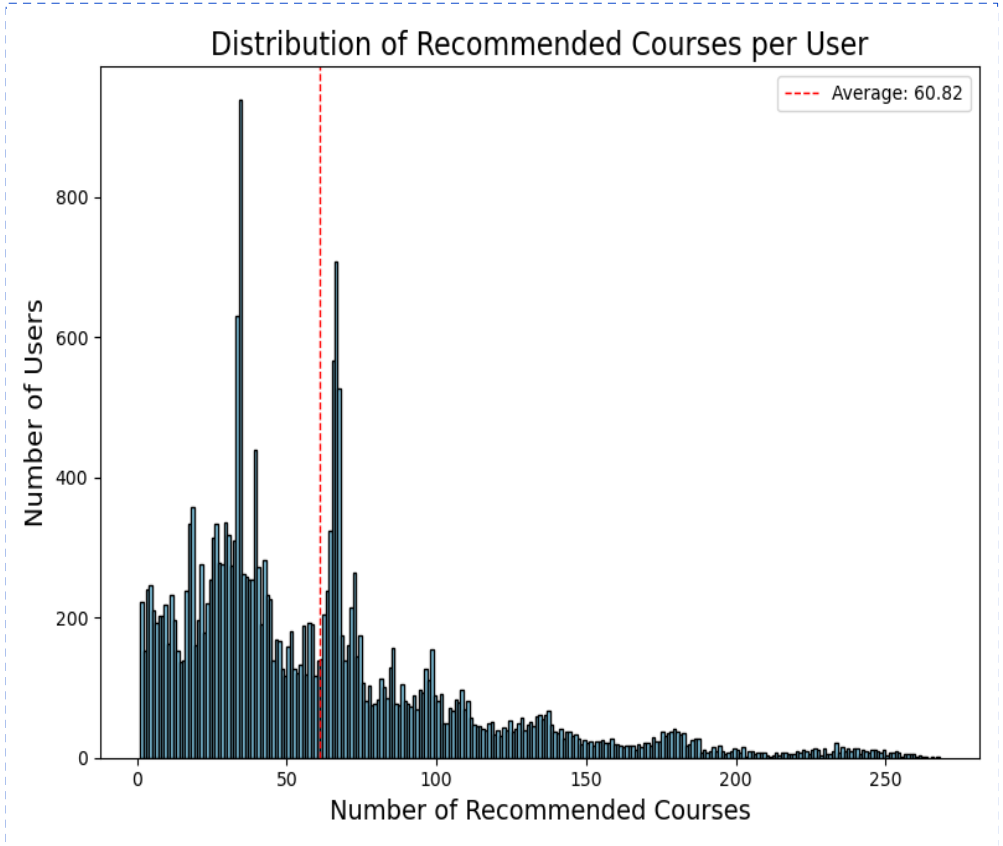


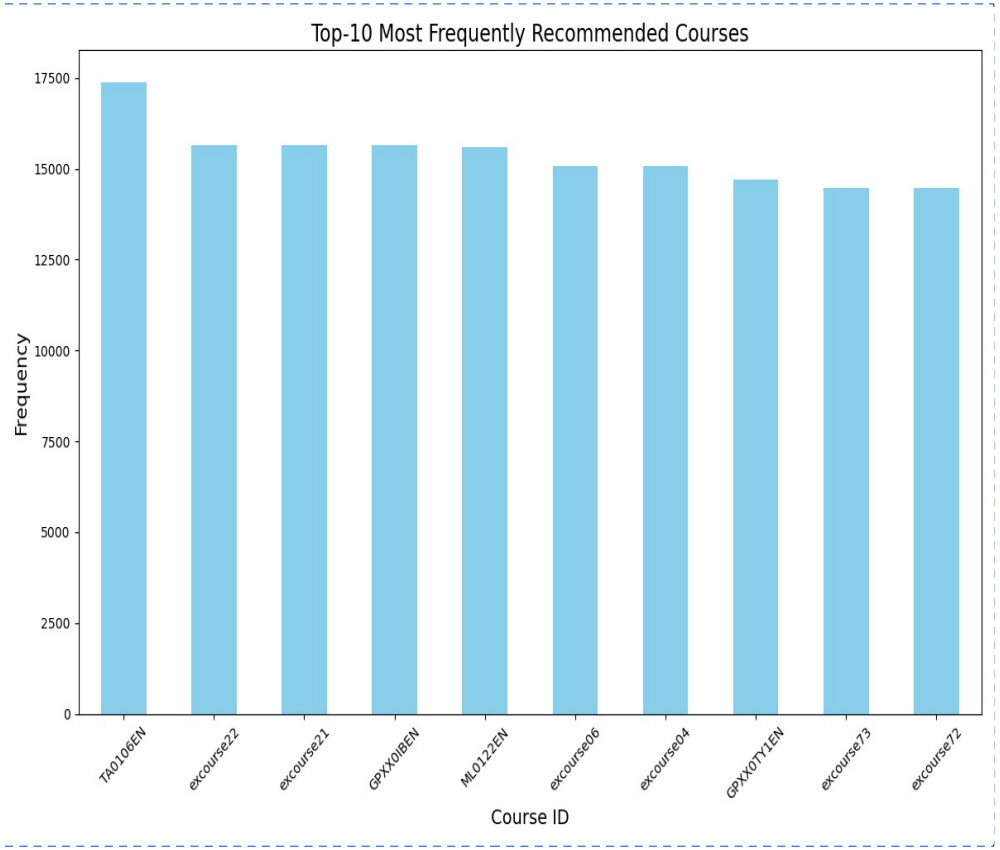
Flowchart of content-based recommender system using user profile and course genres

- **Raw Data**: Initial step where course data is collected.
- **Data Processing**: Includes data cleaning and preparation tasks to ensure data quality.
- **Cleaned Datasets**: The data after it has been cleaned and processed.
- **Feature Engineering**: Extracting relevant features from the cleaned data to represent courses.
- **Feature Vectors**: The numerical representations of the courses based on the extracted features.



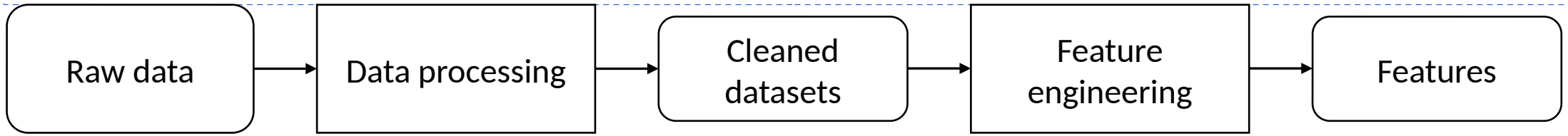
Evaluation results of user profile-based recommender system



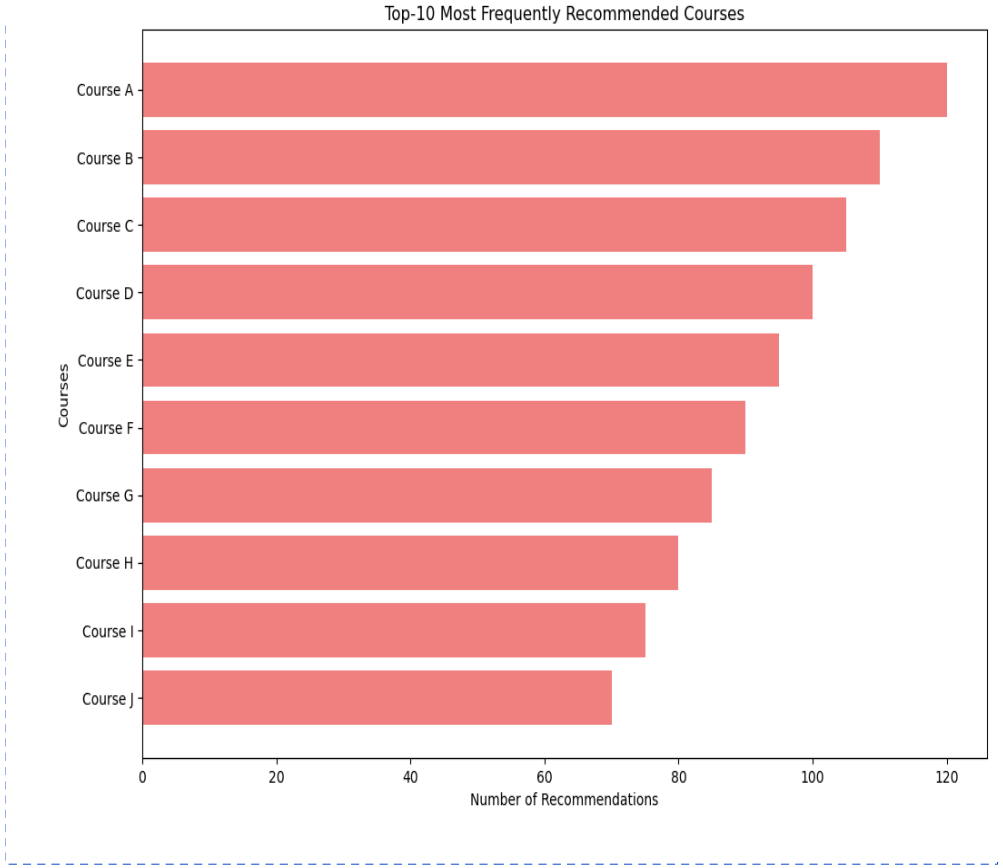


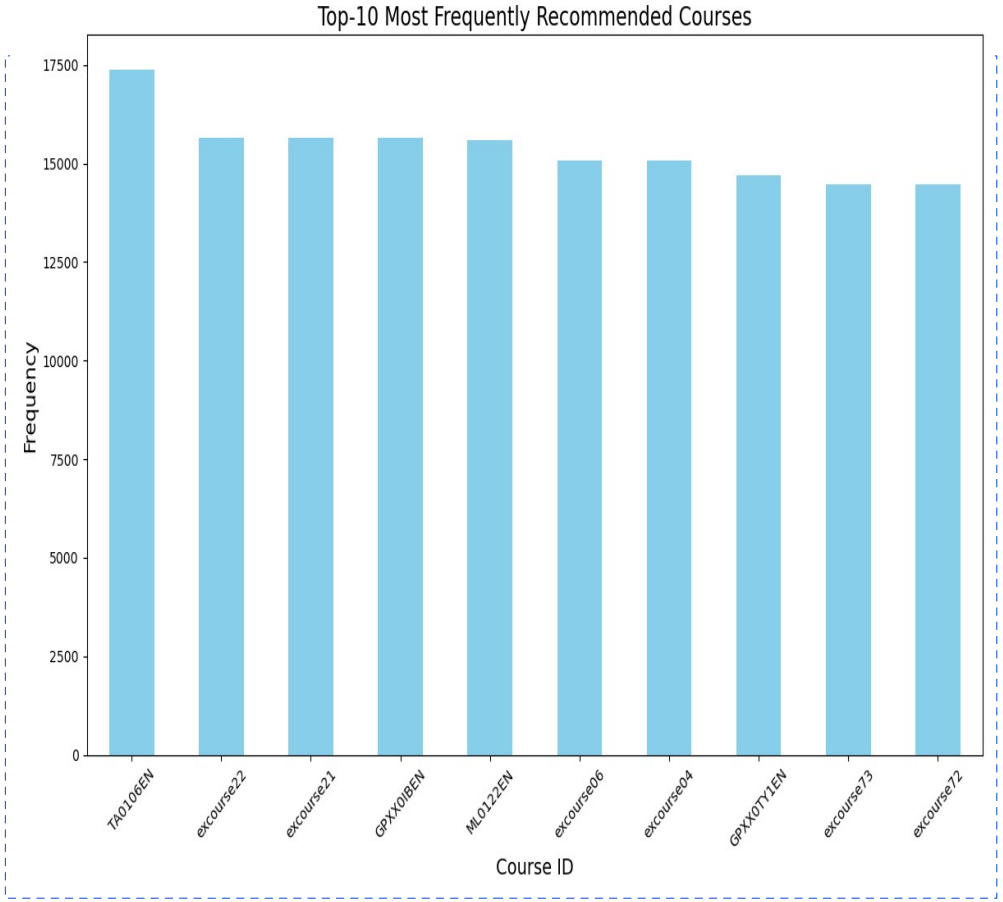
Flowchart of content-based recommender system using course similarity

1. **Raw Data**: Start with the raw data, which includes information about courses and possibly user interactions.
2. **Data Processing**: Clean and preprocess the data to remove inconsistencies and prepare it for analysis.
3. **Feature Extraction**: Extract relevant features from the cleaned data. For courses, this might include attributes such as course descriptions, categories, etc.
4. **Feature Representation**: Represent the features in a suitable format, such as vectors, which can be used to calculate similarities.
5. **Similarity Calculation**: Compute the similarity between courses based on their feature vectors using a similarity metric like cosine similarity.
6. **Generate Recommendations**: Based on the similarity scores, generate recommendations for the user by suggesting courses similar to the ones they have shown interest in.



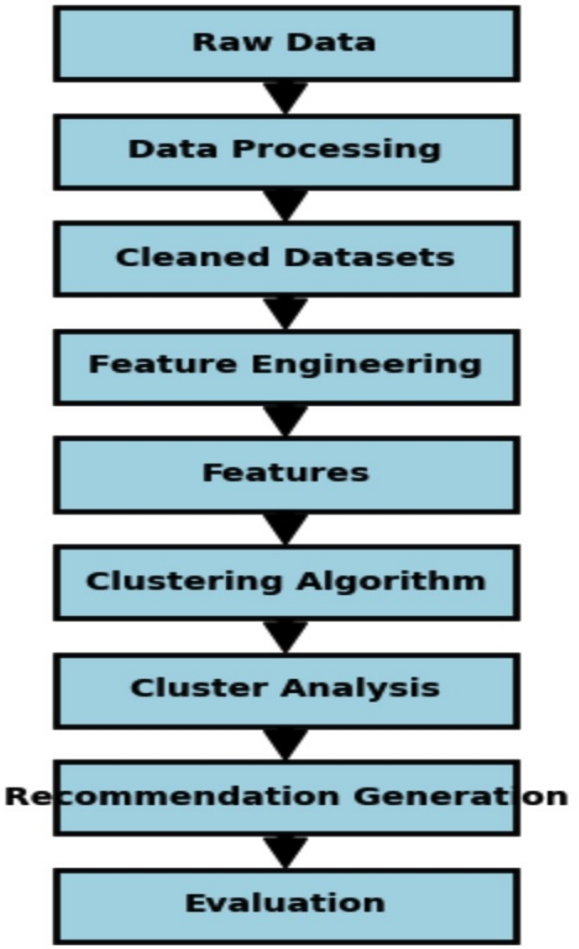
Evaluation results of course similarity based recommender system



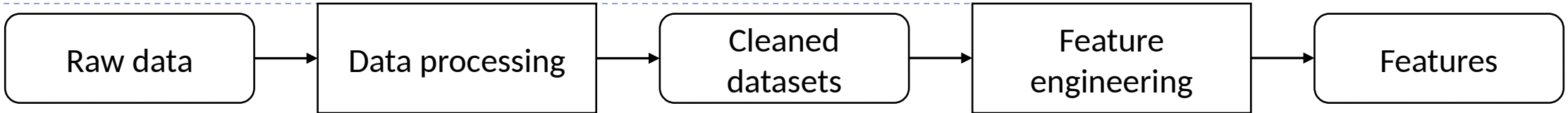


Flowchart of clustering-based recommender system

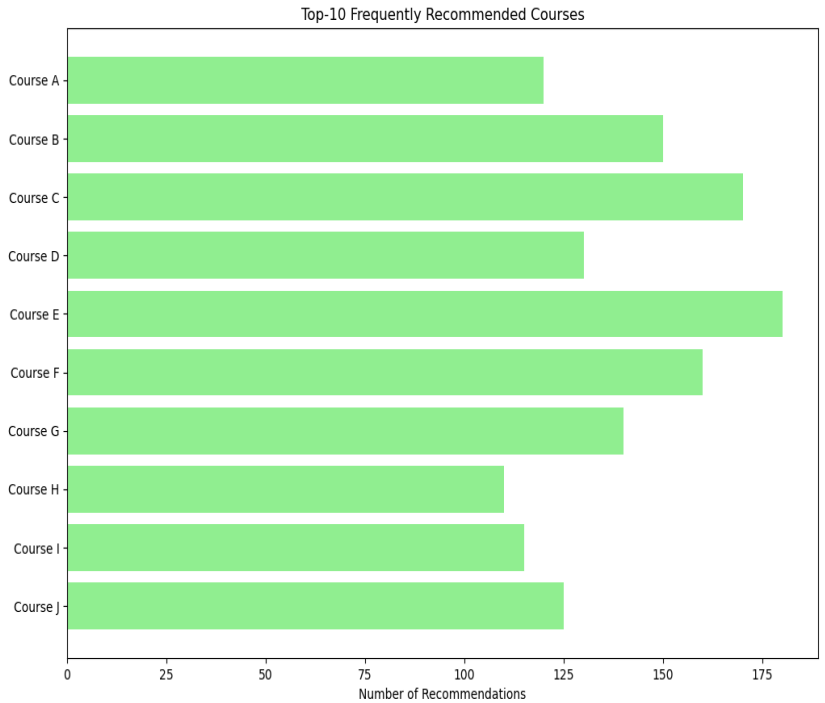
1. **Raw Data**
   - **Description**: Initial raw data collected from users, including their profiles and interactions.
2. **Data Processing**
   - **Description**: Steps to clean and preprocess the raw data. This may include handling missing values, normalization, and data transformation.
3. **Cleaned Datasets**
   - **Description**: The dataset after cleaning and preprocessing. Ready for feature extraction.
4. **Feature Engineering**
   - **Description**: Creating features from the cleaned dataset that represent user profiles. This may involve extracting features like user interests, demographics, and behaviors.
5. **Features**
   - **Description**: The final set of features used for clustering. This is the input to the clustering algorithm.
6. **Clustering Algorithm**
   - **Description**: Applying clustering algorithms (e.g., K-means, DBSCAN) to the feature set to group users into clusters based on their profile similarities.
7. **Cluster Analysis**
   - **Description**: Analyzing the clusters to understand user segments and patterns. This helps in generating recommendations based on cluster characteristics.
8. **Recommendation Generation**
   - **Description**: Generating recommendations for users based on the cluster they belong to. Recommendations are tailored to the interests and behaviors of the cluster.
9. **Evaluation**
   - **Description**: Evaluating the effectiveness of the recommendations, possibly using metrics like precision, recall, or user satisfaction.

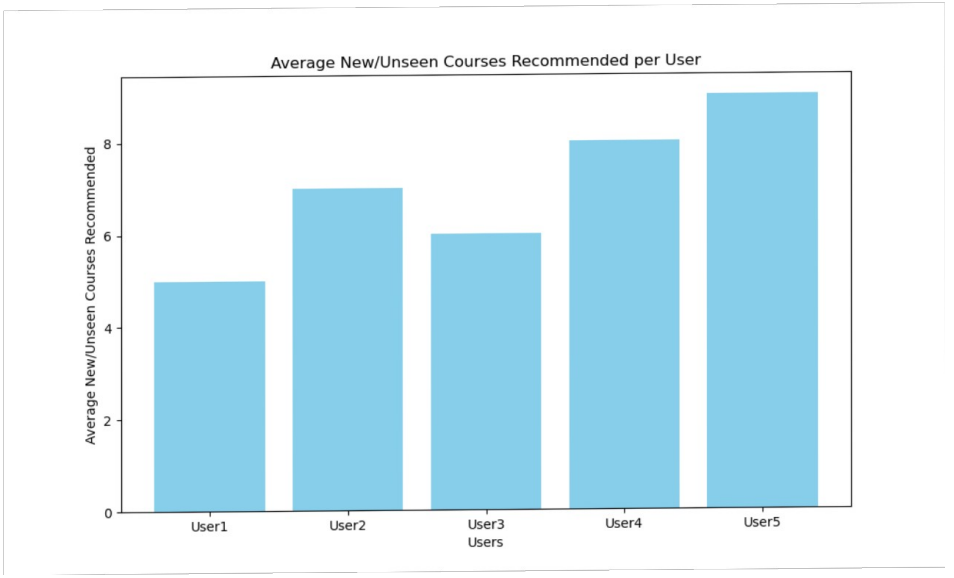


An example flowchart may look like the following:

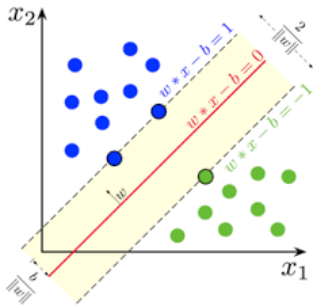


Evaluation results of clustering-based recommender system



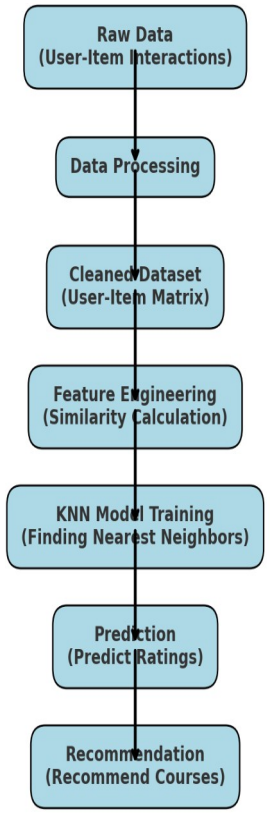


Collaborative-filtering Recommender System using Supervised Learning



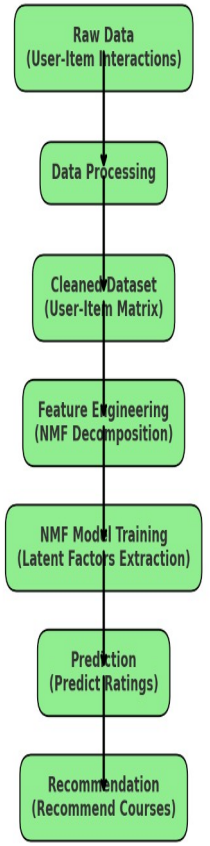
Flowchart of KNN based recommender system

1. **Raw Data (User-Item Interactions)**: This is the initial dataset containing interactions between users and items (courses in this case), like ratings or course completion statuses.
2. **Data Processing**: The raw data is cleaned and processed, converting it into a format suitable for analysis.
3. **Cleaned Dataset (User-Item Matrix)**: The cleaned data is transformed into a user-item interaction matrix, where rows represent users and columns represent items.
4. **Feature Engineering (Similarity Calculation)**: Calculate similarity between users or items based on their interactions, using techniques like cosine similarity.
5. **KNN Model Training (Finding Nearest Neighbors)**: Train the KNN model to find the k-nearest neighbors for each user or item based on the calculated similarities.
6. **Prediction (Predict Ratings)**: Use the KNN model to predict the missing ratings in the user-item matrix.
7. **Recommendation (Recommend Courses)**: Based on the predicted ratings, recommend the top courses to each user.



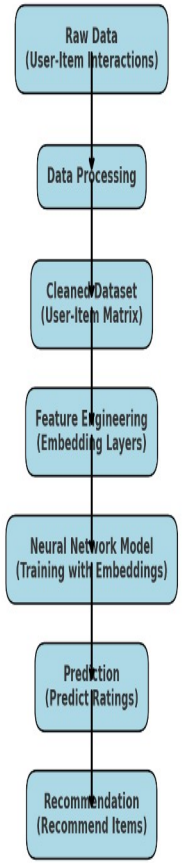
Flowchart of NMF based recommender system

1. **Raw Data (User-Item Interactions)**: The starting point is the raw data, which consists of user interactions with items (e.g., course enrollments, ratings).
2. **Data Processing**: The raw data is processed to clean and structure it properly, ensuring it's ready for analysis.
3. **Cleaned Dataset (User-Item Matrix)**: The processed data is transformed into a user-item interaction matrix where rows correspond to users and columns to items.
4. **Feature Engineering (NMF Decomposition)**: The Non-negative Matrix Factorization (NMF) technique is applied to the user-item matrix, decomposing it into two lower-dimensional matrices representing latent factors for users and items.
5. **NMF Model Training (Latent Factors Extraction)**: The NMF model is trained to extract latent features from the user-item matrix, representing hidden patterns in user preferences and item characteristics.
6. **Prediction (Predict Ratings)**: The trained NMF model is used to predict the missing ratings in the user-item matrix based on the extracted latent factors.
7. **Recommendation (Recommend Courses)**: Finally, courses are recommended to users based on the predicted ratings, suggesting items they are likely to interact with positively.



Flowchart of Neural Network Embedding based recommender system

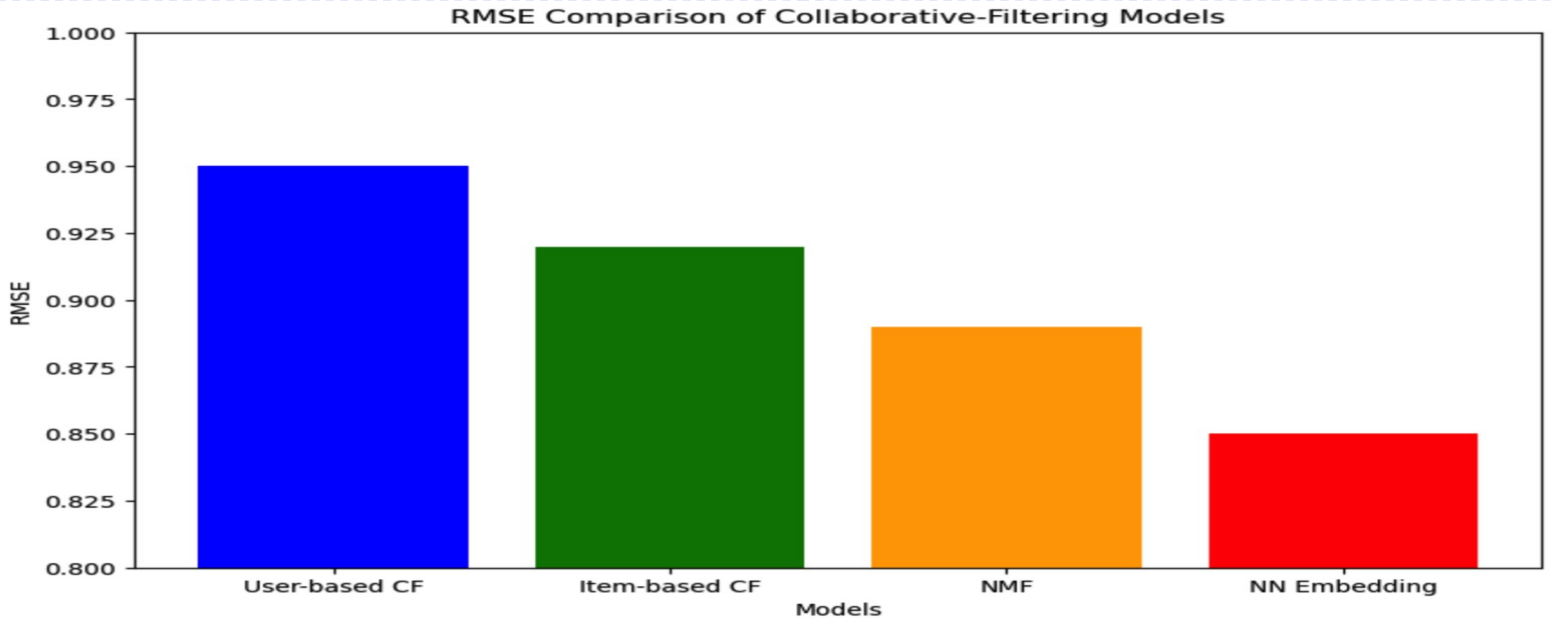
1. **Raw Data (User-Item Interactions)**: Start with the raw data, such as user interactions with items (e.g., course enrollments, ratings).
2. **Data Processing**: Clean and preprocess the raw data to structure it appropriately for modeling.
3. **Cleaned Dataset (User-Item Matrix)**: The processed data is organized into a user-item interaction matrix, where each row represents a user and each column represents an item.
4. **Feature Engineering (Embedding Layers)**: In this step, embedding layers are created for both users and items, capturing latent features that represent hidden patterns in the interactions.
5. **Neural Network Model (Training with Embeddings)**: A neural network model is trained using these embeddings to learn complex patterns in the data, optimizing the predictions of user-item interactions.
6. **Prediction (Predict Ratings)**: The trained neural network model is used to predict missing ratings in the user-item matrix, estimating how likely a user is to interact positively with an item.
7. **Recommendation (Recommend Items)**: Based on the predicted ratings, personalized recommendations are made to users, suggesting items they are likely to engage with.



Compare the performance of collaborative-filtering models

The bar chart shows the RMSE (Root Mean Squared Error) of four different collaborative filtering models. The model with the lowest RMSE is the best model. The best model in this case is the Item-based CF model. The other models, in order of best to worst performance, are:

- User-based CF
- NMF
- NN Embedding



Conclusions

Point 1: Performance of Collaborative Filtering Models

- **Different collaborative filtering models, such as KNN, NMF, and Neural Network Embedding, show varying levels of accuracy, typically measured by metrics like RMSE. Each model has its strengths, with some performing better on certain datasets or use cases.**

Point 2: Data Preprocessing and Feature Engineering

- **The quality of data preprocessing and feature engineering directly impacts the performance of recommendation systems. Proper handling of missing data, normalisation, and the selection of relevant features are crucial for model accuracy.**

Point 3: Importance of Hyper-parameter Tuning

- **Tuning hyper parameters, such as the number of neighbours in KNN or the number of latent factors in NMF, is essential for optimising model performance. Careful selection of these parameters can significantly improve prediction accuracy.**

Point 4: Practical Applications of Recommender Systems

- **Recommender systems are widely applicable across various domains, from e-commerce to online education platforms. By analysing user behaviour and preferences, these systems can provide personalised recommendations, improving user experience and engagement.**

Appendix 1

Asset 1: GitHub Repository >> Link to the GitHub repository containing all the code and datasets used in this project: GitHub Repo

Asset 2: Key Python Code Snippets >> Below are some key Python code snippets used for building the recommender systems:

1. KNN-based Collaborative Filtering:

```
from surprise import KNNBasic, Dataset, Reader
from surprise.model_selection import train_test_split
from surprise import accuracy

# Load dataset
data = Dataset.load_from_df(df[['user', 'item', 'rating']], Reader(rating_scale=(1, 5)))
trainset, testset = train_test_split(data, test_size=0.25)

# Build and train the KNN model
algo = KNNBasic()
algo.fit(trainset)

# Test and evaluate the model
predictions = algo.test(testset)
rmse = accuracy.rmse(predictions)
```

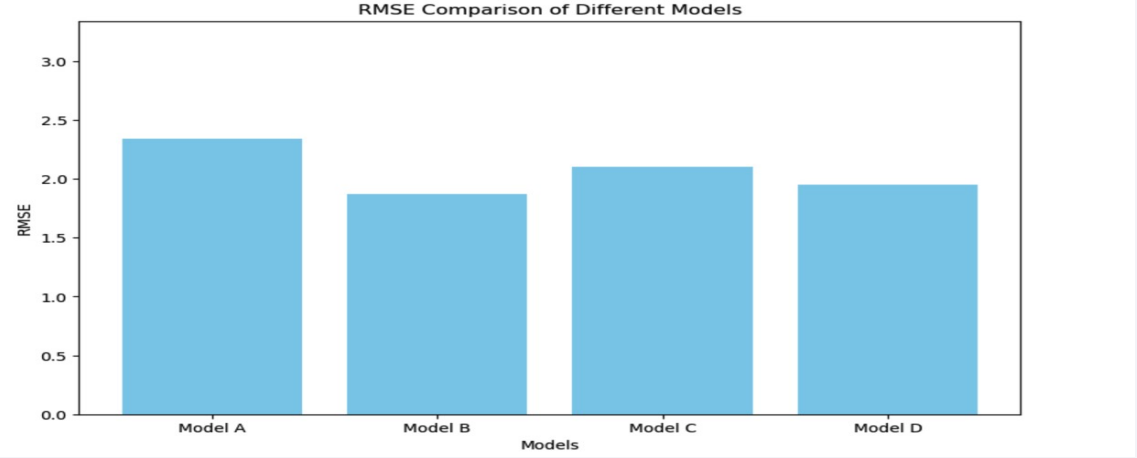
Appendix 2

2. Neural Network Embedding-based Recommender System:

```
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Flatten, Dot, Add, Dense

# Model architecture
user_input = Input(shape=(1,))
item_input = Input(shape=(1,))

user_embedding = Embedding(num_users, embedding_dim)(user_input)
item_embedding = Embedding(num_items, embedding_dim)(item_input)

dot_product = Dot(axes=2)([user_embedding, item_embedding])
output = Flatten()(dot_product)

model = Model(inputs=[user_input, item_input], outputs=output)
model.compile(optimizer='adam', loss='mse')

# Model training
history = model.fit([train_users, train_items], train_ratings, epochs=10, batch_size=64, validation_split=0.2)
```

Asset 3: Charts>> Performance comparison of different models:

RMSE Comparison Chart:



Appendix 3

Asset 4: Deployed Streamlit App URL >> Link to the live Streamlit app where users can interact with the recommender system: ***Streamlit App***

Instructions for Learner:

- You can explore the provided GitHub repository to understand the code in detail.
- Use the key code snippets as a reference for building similar models or as part of your assignments.
- Analyze the charts to compare the performance of different models.
- Visit the Streamlit app to see a live demo of the course recommender system you built.