CS6003 – BIG DATA ANALYTICS

<u>Courses Recommendation System</u>

TEAM 2:

Murali B	-	2021103545
Monica T	-	2021103543
Rajasekar S K	-	2021103562
J Ganesh	-	2021103602

SYNOPSIS:

OBJECTIVES:

The project aims to develop an application that leverages a collaborative and content-based recommendation system to suggest relevant courses to users. The primary objectives include enhancing user engagement by providing personalized recommendations aligned with individual interests and learning goals, diversifying course selections to encourage exploration of new topics, improving user retention through consistent and valuable suggestions.

PROBLEM BRIEFING:

The project addresses several key challenges faced by learners in today's digital education landscape. Firstly, with the abundance of online courses available, users often struggle to discover relevant content tailored to their specific interests and learning objectives. This leads to frustration and disengagement, hindering their overall learning experience. Additionally, existing recommendation systems often lack personalization and fail to consider the diverse preferences and learning styles of individual users. This results in generic recommendations that do not effectively cater to the unique needs of each learner. Furthermore, as online learning platforms continue to grow in popularity, scalability becomes a critical issue, requiring robust systems capable of handling large volumes of user data and course information efficiently. By tackling these challenges, the project aims to create an innovative solution that revolutionizes the way users discover and engage with online learning resources, ultimately fostering a more enriching and personalized learning experience for all.

CHALLENGES:

1. **Data Sparsity and Cold Start Problem:**

   One of the primary challenges is dealing with data sparsity, especially for new users who have limited interaction history on the platform. This leads to the cold start problem, where traditional recommendation algorithms struggle to provide accurate suggestions due to insufficient user data.

2. **Content Representation and Feature Engineering:**

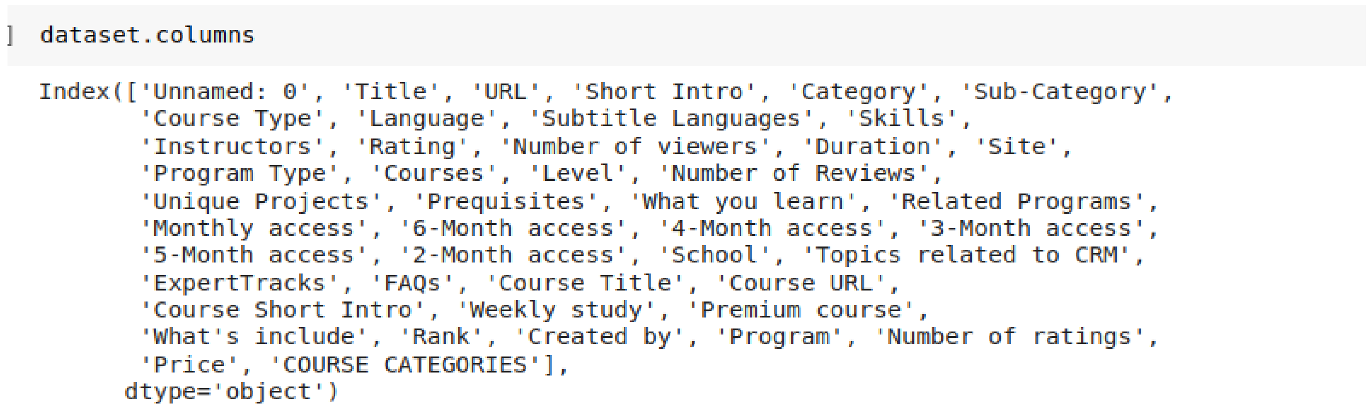
   Effectively representing course content and extracting relevant features for recommendation poses a significant challenge. Courses span diverse topics and formats, requiring sophisticated techniques to accurately capture their nuances and similarities.

3. **Scalability and Performance:**

   As the user base and course catalog grows, ensuring the scalability and performance of the recommendation system becomes increasingly challenging. Efficient algorithms and infrastructure are needed to handle large volumes of data and deliver real-time recommendations without sacrificing accuracy

Data Pre-processing.

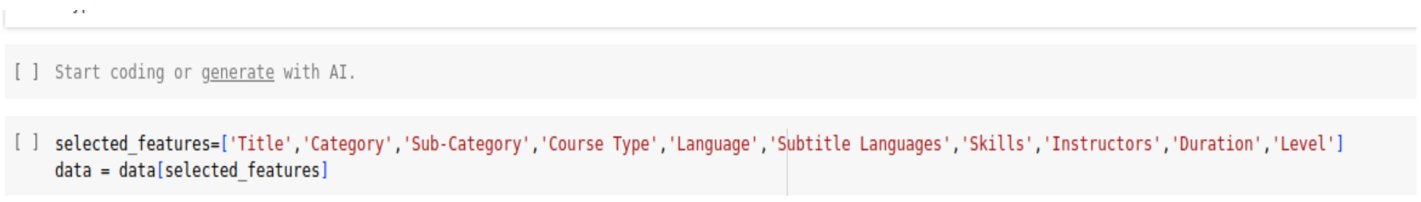
1. We have obtained the dataset from Kaggle which contains the description of 10,000 courses from different learning platforms like Coursera, Udacity, and Udemy. The complete list of features is shown in Figure 1.



```
]  dataset.columns

 Index(['Unnamed: 0', 'Title', 'URL', 'Short Intro', 'Category', 'Sub-Category',
        'Course Type', 'Language', 'Subtitle Languages', 'Skills',
        'Instructors', 'Rating', 'Number of viewers', 'Duration', 'Site',
        'Program Type', 'Courses', 'Level', 'Number of Reviews',
        'Unique Projects', 'Prequisites', 'What you learn', 'Related Programs',
        'Monthly access', '6-Month access', '4-Month access', '3-Month access',
        '5-Month access', '2-Month access', 'School', 'Topics related to CRM',
        'ExpertTracks', 'FAQs', 'Course Title', 'Course URL',
        'Course Short Intro', 'Weekly study', 'Premium course',
        'What's include', 'Rank', 'Created by', 'Program', 'Number of ratings',
        'Price', 'COURSE CATEGORIES'],
       dtype='object')
```

Figure 1: Features of the Dataset

2. Feature extraction is done and the most significant features describing the courses have been taken for further steps.

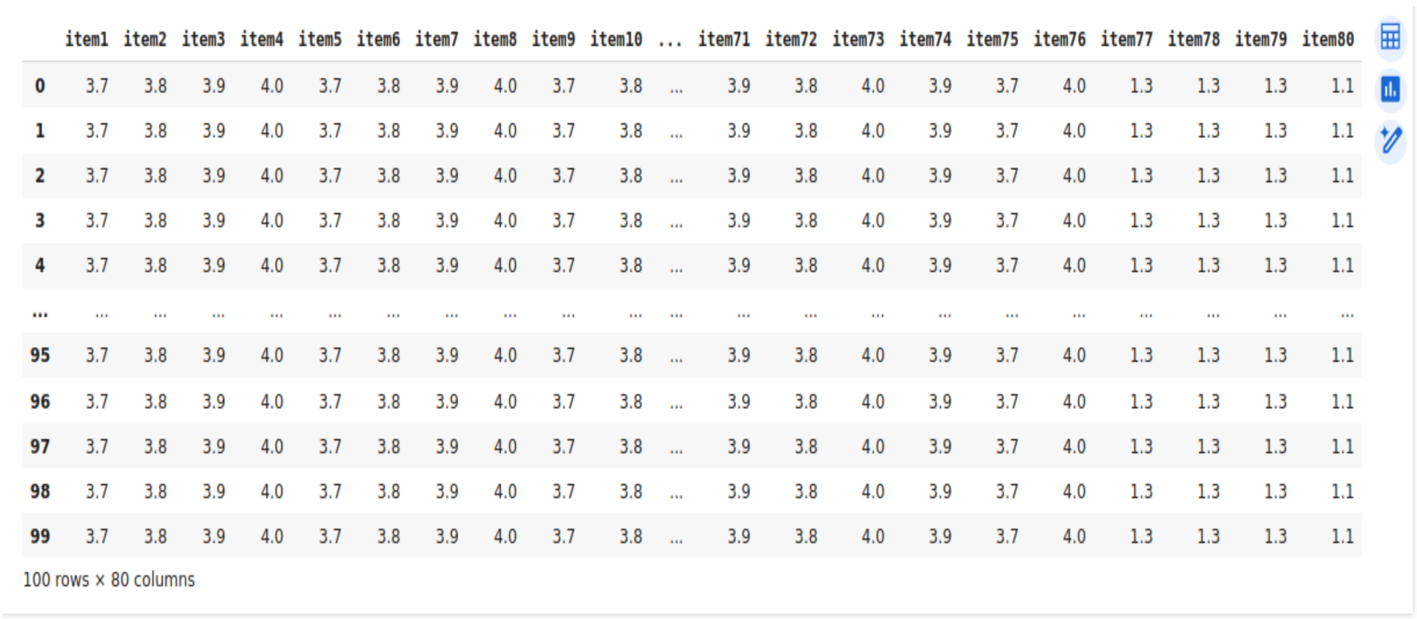


```
[ ] Start coding or generate with AI.

[ ] selected_features=['Title','Category','Sub-Category','Course Type','Language','Subtitle Languages','Skills','Instructors','Duration','Level']
    data = data[selected_features]
```

Figure 2: Selected Features

3. The NULL values in the data are filled by using the imputation technique with KNN which effectively encodes the categorical columns using label encoding and the value of the current null data point is determined by its neighbours.
4. For implementing collaborative filtering, we need user_id, and user_rating columns, we manually augmented them under the following assumptions.
   - User clustering in which a cluster of users gives a rating to a particular set of courses.
   - Randomness is added by selecting a few users from one cluster and assigning ratings for the courses from different clusters.
5. The final dataset used for constructing the rating matrix has 100 unique users who had rated 80 courses from the dataset.

6. The sparsity in the matrix is handled in two ways separately.
   - Substituting each unrated value by the mean of the rating given by the user.
   - Using kernel density estimation, the user-rated points have been extrapolated to find the underlying distribution. Here 'x' is the random variable which takes values from 1 to 5. The rating with maximum probability is substituted for each user individually.



	item1	item2	item3	item4	item5	item6	item7	item8	item9	item10	...	item71	item72	item73	item74	item75	item76	item77	item78	item79	item80
0	3.7	3.8	3.9	4.0	3.7	3.8	3.9	4.0	3.7	3.8	...	3.9	3.8	4.0	3.9	3.7	4.0	1.3	1.3	1.3	1.1
1	3.7	3.8	3.9	4.0	3.7	3.8	3.9	4.0	3.7	3.8	...	3.9	3.8	4.0	3.9	3.7	4.0	1.3	1.3	1.3	1.1
2	3.7	3.8	3.9	4.0	3.7	3.8	3.9	4.0	3.7	3.8	...	3.9	3.8	4.0	3.9	3.7	4.0	1.3	1.3	1.3	1.1
3	3.7	3.8	3.9	4.0	3.7	3.8	3.9	4.0	3.7	3.8	...	3.9	3.8	4.0	3.9	3.7	4.0	1.3	1.3	1.3	1.1
4	3.7	3.8	3.9	4.0	3.7	3.8	3.9	4.0	3.7	3.8	...	3.9	3.8	4.0	3.9	3.7	4.0	1.3	1.3	1.3	1.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	3.7	3.8	3.9	4.0	3.7	3.8	3.9	4.0	3.7	3.8	...	3.9	3.8	4.0	3.9	3.7	4.0	1.3	1.3	1.3	1.1
96	3.7	3.8	3.9	4.0	3.7	3.8	3.9	4.0	3.7	3.8	...	3.9	3.8	4.0	3.9	3.7	4.0	1.3	1.3	1.3	1.1
97	3.7	3.8	3.9	4.0	3.7	3.8	3.9	4.0	3.7	3.8	...	3.9	3.8	4.0	3.9	3.7	4.0	1.3	1.3	1.3	1.1
98	3.7	3.8	3.9	4.0	3.7	3.8	3.9	4.0	3.7	3.8	...	3.9	3.8	4.0	3.9	3.7	4.0	1.3	1.3	1.3	1.1
99	3.7	3.8	3.9	4.0	3.7	3.8	3.9	4.0	3.7	3.8	...	3.9	3.8	4.0	3.9	3.7	4.0	1.3	1.3	1.3	1.1

100 rows × 80 columns

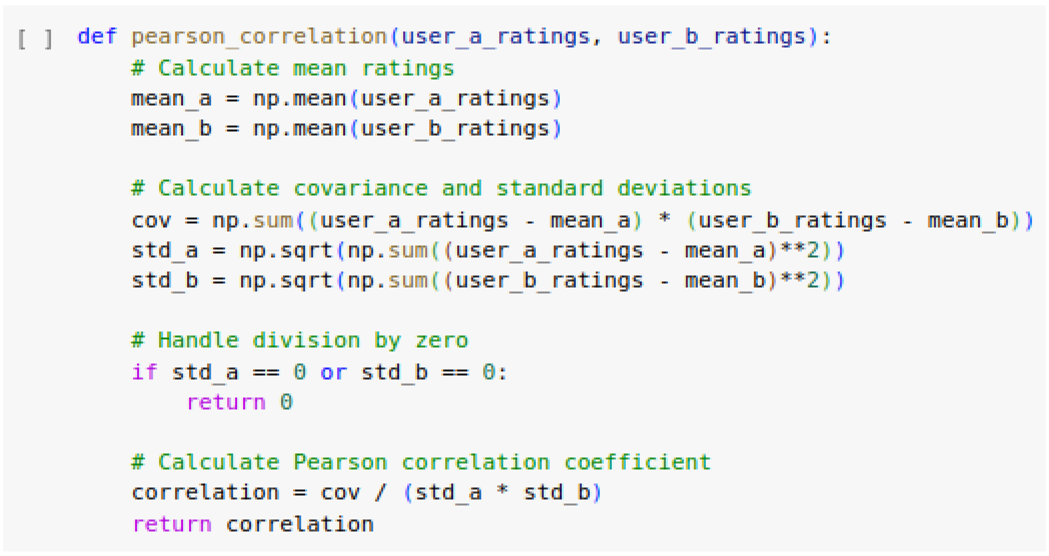
Figure 3: Rating Matrix after handling sparsity

Algorithmic Implementation:

1. User-based Collaborative Filtering

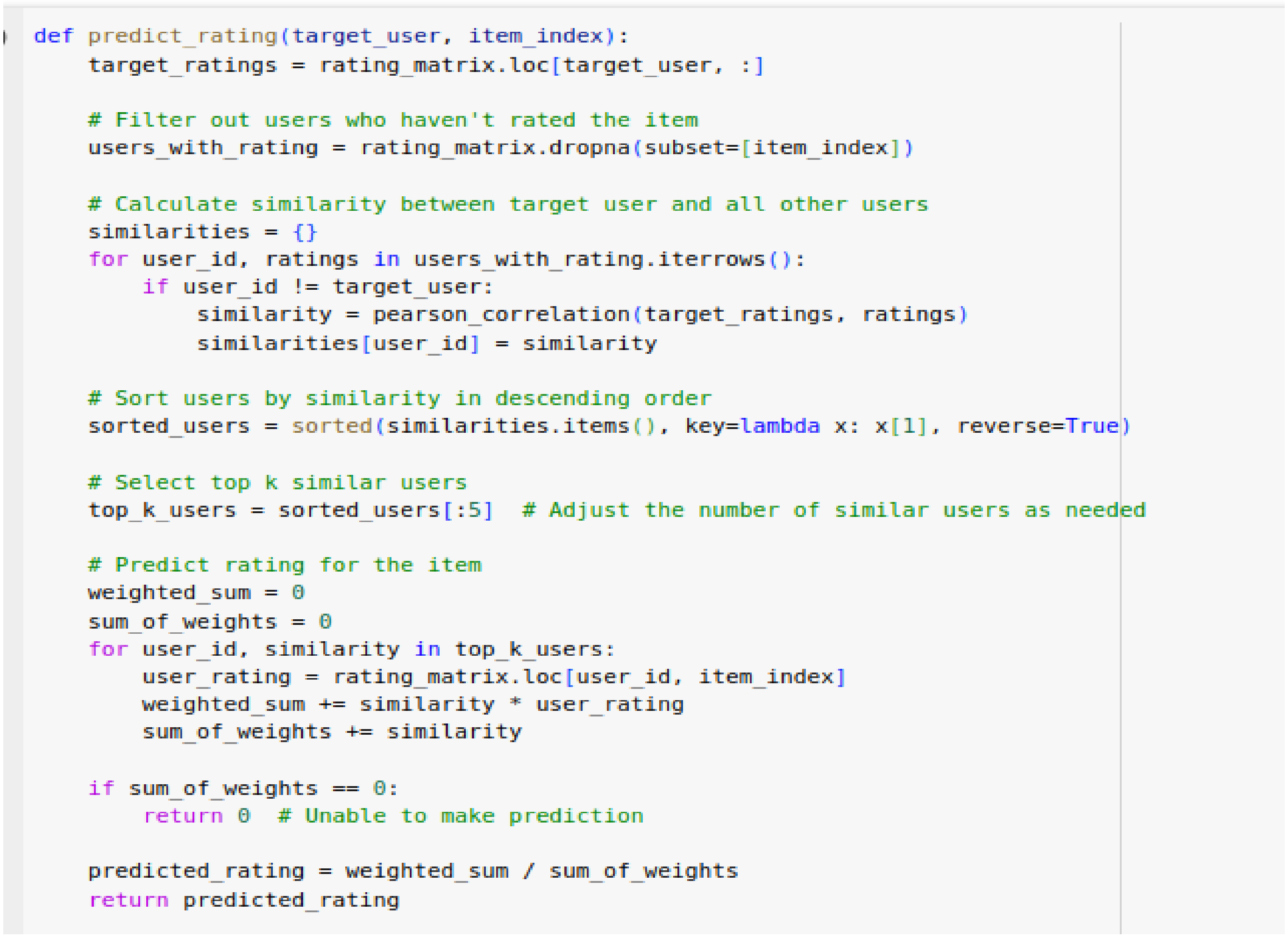
User-based collaborative filtering predicts what a user might like by finding users with similar interests based on past interactions, then recommending items those similar users reviewed but the target user hasn't experienced yet.

- User's identity is represented by the user_id which is then used to find the 'n' similar users using Pearson correlation.
- The prediction of each unrated item has been calculated using the weighted average method.
- The ratings above the threshold are shown to the user.



```
[ ]  def pearson_correlation(user_a_ratings, user_b_ratings):
         # Calculate mean ratings
         mean_a = np.mean(user_a_ratings)
         mean_b = np.mean(user_b_ratings)

         # Calculate covariance and standard deviations
         cov = np.sum((user_a_ratings - mean_a) * (user_b_ratings - mean_b))
         std_a = np.sqrt(np.sum((user_a_ratings - mean_a)**2))
         std_b = np.sqrt(np.sum((user_b_ratings - mean_b)**2))

         # Handle division by zero
         if std_a == 0 or std_b == 0:
             return 0

         # Calculate Pearson correlation coefficient
         correlation = cov / (std_a * std_b)
         return correlation
```

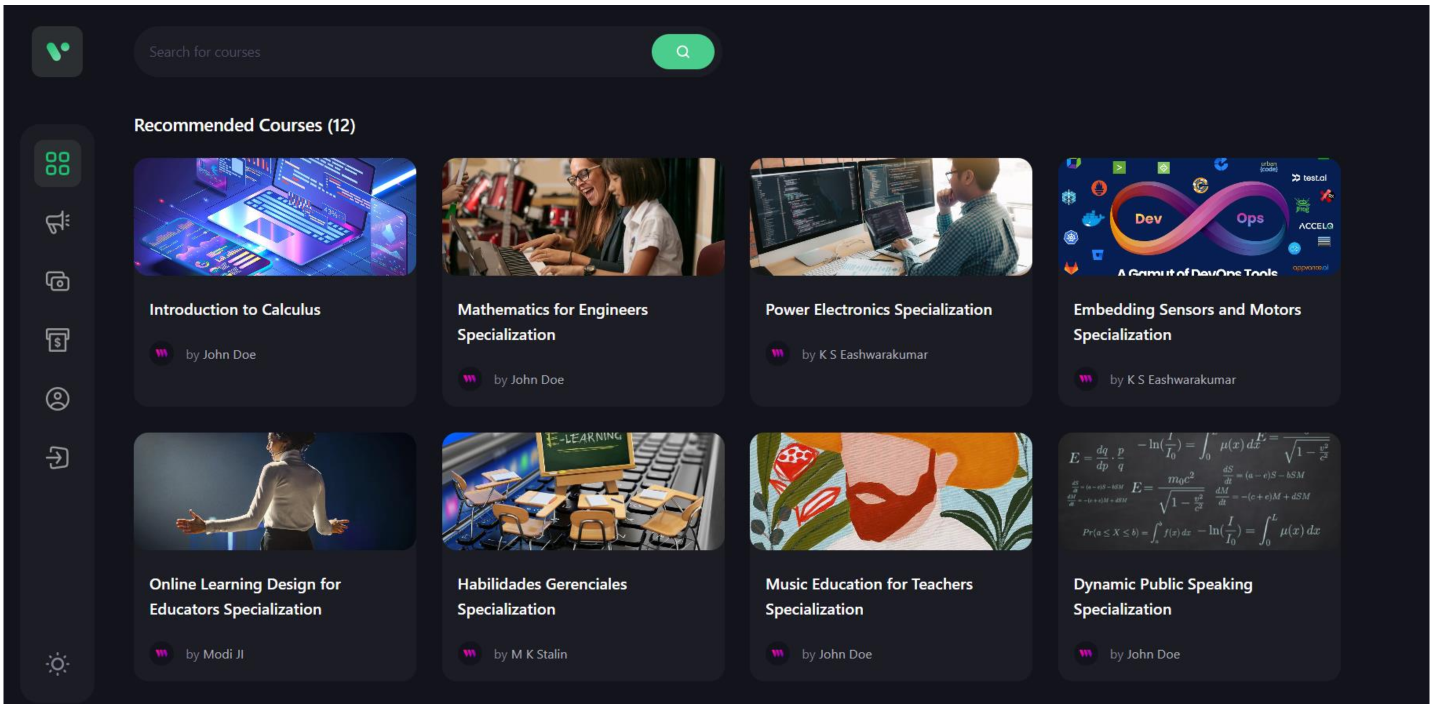
Figure 4: Pearson Correlation



```
def predict_rating(target_user, item_index):
    target_ratings = rating_matrix.loc[target_user, :]

    # Filter out users who haven't rated the item
    users_with_rating = rating_matrix.dropna(subset=[item_index])

    # Calculate similarity between target user and all other users
    similarities = {}
    for user_id, ratings in users_with_rating.iterrows():
        if user_id != target_user:
            similarity = pearson_correlation(target_ratings, ratings)
            similarities[user_id] = similarity

    # Sort users by similarity in descending order
    sorted_users = sorted(similarities.items(), key=lambda x: x[1], reverse=True)

    # Select top k similar users
    top_k_users = sorted_users[:5]  # Adjust the number of similar users as needed

    # Predict rating for the item
    weighted_sum = 0
    sum_of_weights = 0
    for user_id, similarity in top_k_users:
        user_rating = rating_matrix.loc[user_id, item_index]
        weighted_sum += similarity * user_rating
        sum_of_weights += similarity

    if sum_of_weights == 0:
        return 0  # Unable to make prediction

    predicted_rating = weighted_sum / sum_of_weights
    return predicted_rating
```

Figure 5: Prediction function for unrated items

In our application user-based collaborative system is used for suggesting courses on the home page as shown in Figure 6.



Recommended Courses (12)

Introduction to Calculus
by John Doe

Mathematics for Engineers Specialization
by John Doe

Power Electronics Specialization
by K S Eashwarakumar

Embedding Sensors and Motors Specialization
by K S Eashwarakumar

Online Learning Design for Educators Specialization
by Modi JI

Habilidades Gerenciales Specialization
by M K Stalin

Music Education for Teachers Specialization
by John Doe

Dynamic Public Speaking Specialization
by John Doe

Figure 6: Home Page Recommendations Using User-Based Collaborative System

2. Item-based Collaborative Filtering:

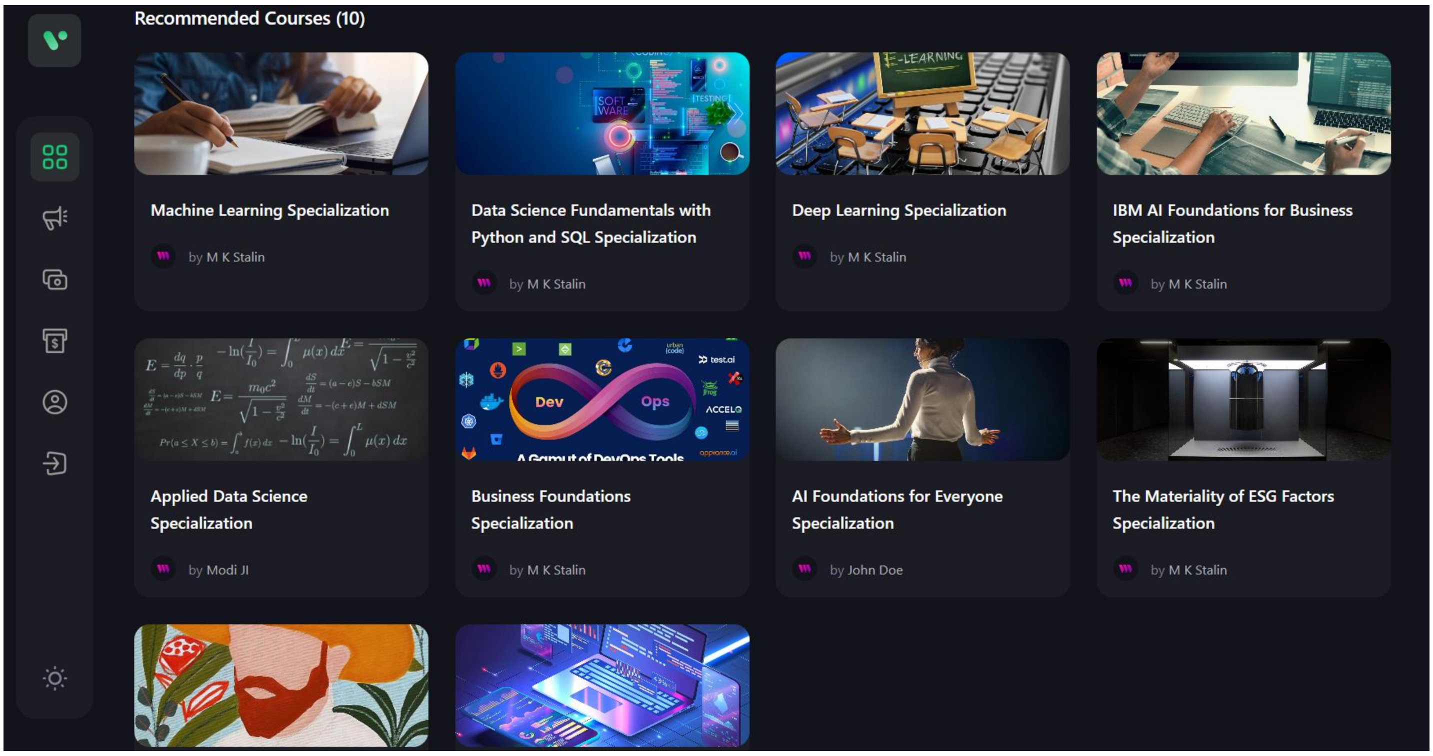
Unlike user-based filtering which focuses on similar users, item-based collaborative filtering recommends items similar to what a user liked by analysing past user interactions.

- The column vector in the rating matrix represents each item.
- K-nearest neighbours is used to finding similar courses with cosine similarity as the metric which is shown in Figure 7.
- The top n courses with high similarity have been recommended.
- To ease the delay, the similarity matrix has been constructed beforehand which has similarity for all item pairs.

```
sim_options = {
    "name": "cosine",
    "user_based": False, # for item based - false
}
algo = KNNWithMeans(sim_options=sim_options)
algo.fit(trainset)
```

Figure 7: KNN Instantiation

In our application, item-based collaborative filtering suggests similar courses for the user to pursue once a user has completed a course.


Recommended Courses (10)

Machine Learning Specialization
by M K Stalin

Data Science Fundamentals with Python and SQL Specialization
by M K Stalin

Deep Learning Specialization
by M K Stalin

IBM AI Foundations for Business Specialization
by M K Stalin

Applied Data Science Specialization
by Modi JI

Business Foundations Specialization
by M K Stalin

AI Foundations for Everyone Specialization
by John Doe

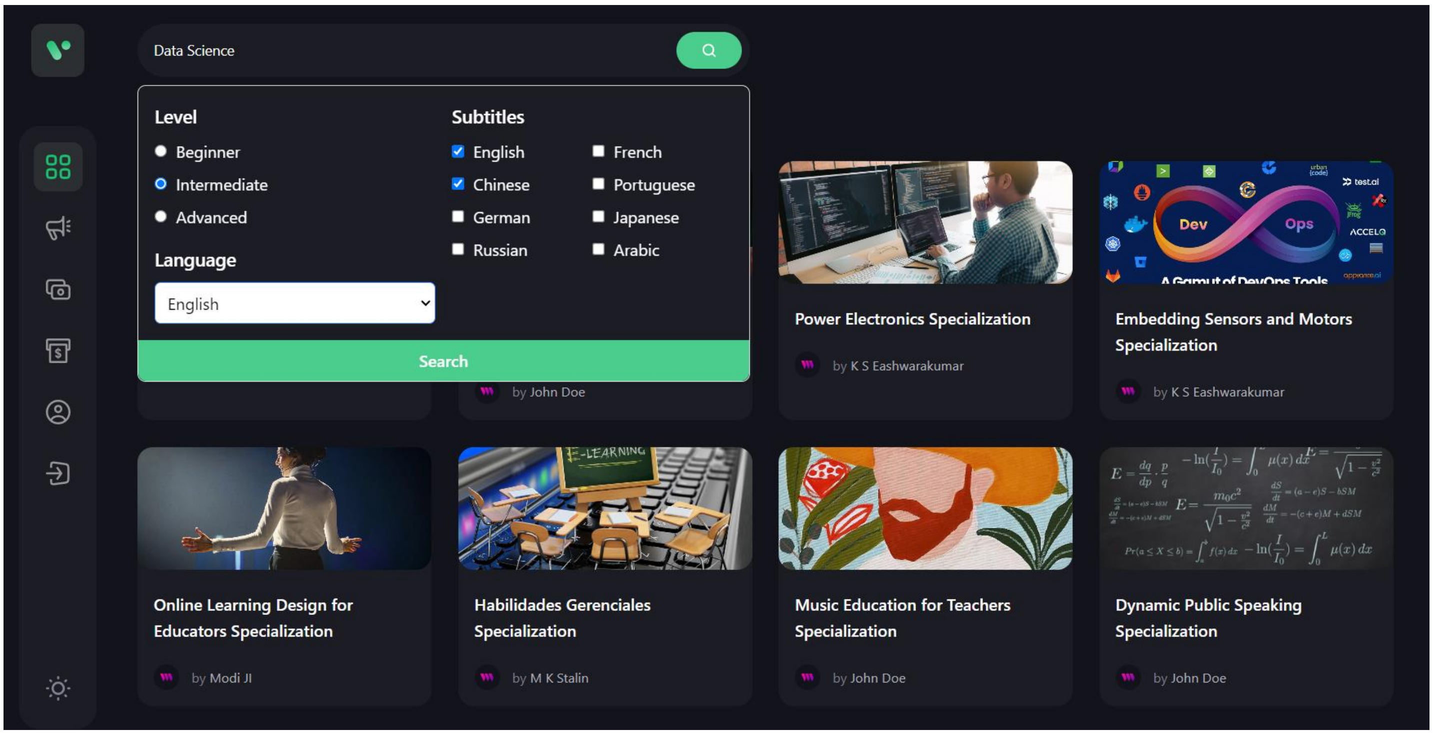
The Materiality of ESG Factors Specialization
by M K Stalin

Figure 8: Item-based Recommendations after completing a course

3. Content-based Filtering System:

To handle the cold start problem, we have implemented a content-based system in parallel with a collaborative approach.

1. Label encoding has been done for categorical columns such as course category, level, language, and subtitles preferred.
2. Using the K-nearest neighbors algorithm courses have been matched with an input vector that contains the encoded value of course category, level, and language.
3. The above results were then checked with the presence of subtitles in the user-preferred languages.
4. Final selected courses with high similarity are recommended to the user.


Data Science

Level
- Beginner
- [✓ Intermediate]
- Advanced

Language: English

Subtitles
- [✓ English]
- French
- [✓ Chinese]
- Portuguese
- German
- Japanese
- Russian
- Arabic

Search

by John Doe

Power Electronics Specialization — by K S Eashwarakumar

Embedding Sensors and Motors Specialization — by K S Eashwarakumar

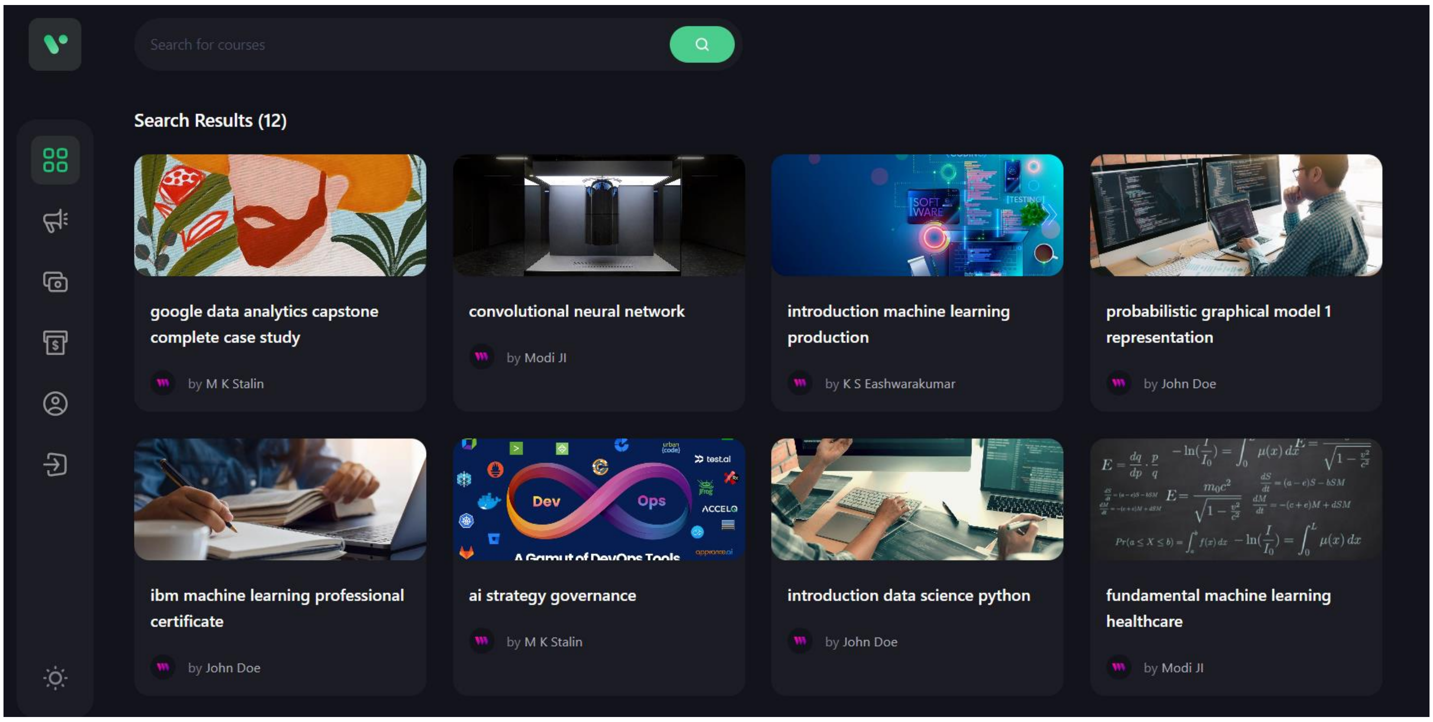
Online Learning Design for Educators Specialization — by Modi JI

Habilidades Gerenciales Specialization — by M K Stalin

Music Education for Teachers Specialization — by John Doe

Dynamic Public Speaking Specialization — by John Doe

Figure 9: Search Component with filters


Search for courses

Search Results (12)

google data analytics capstone complete case study — by M K Stalin

convolutional neural network — by Modi JI

introduction machine learning production — by K S Eashwarakumar

probabilistic graphical model 1 representation — by John Doe

ibm machine learning professional certificate — by John Doe

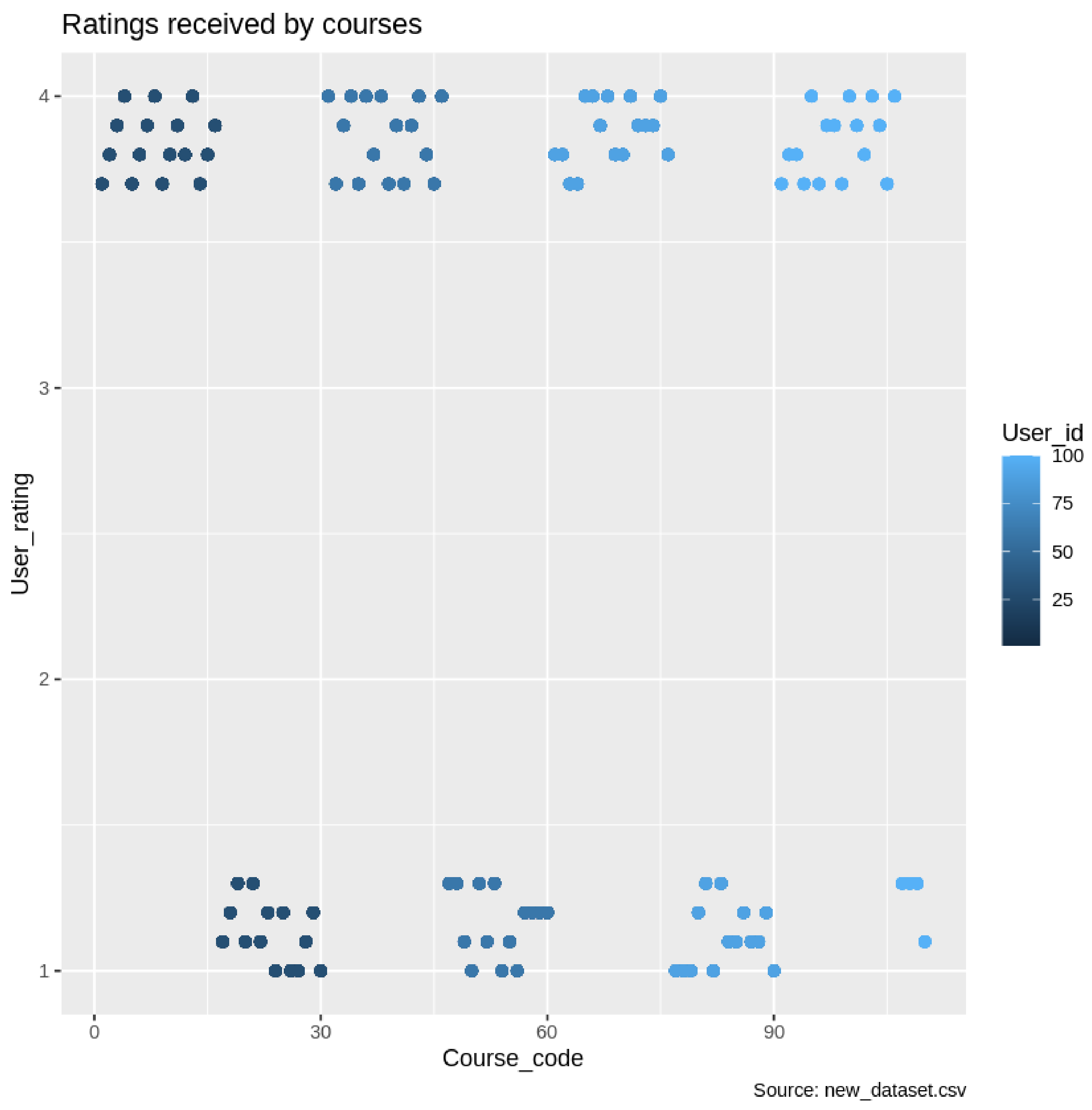
ai strategy governance — by M K Stalin

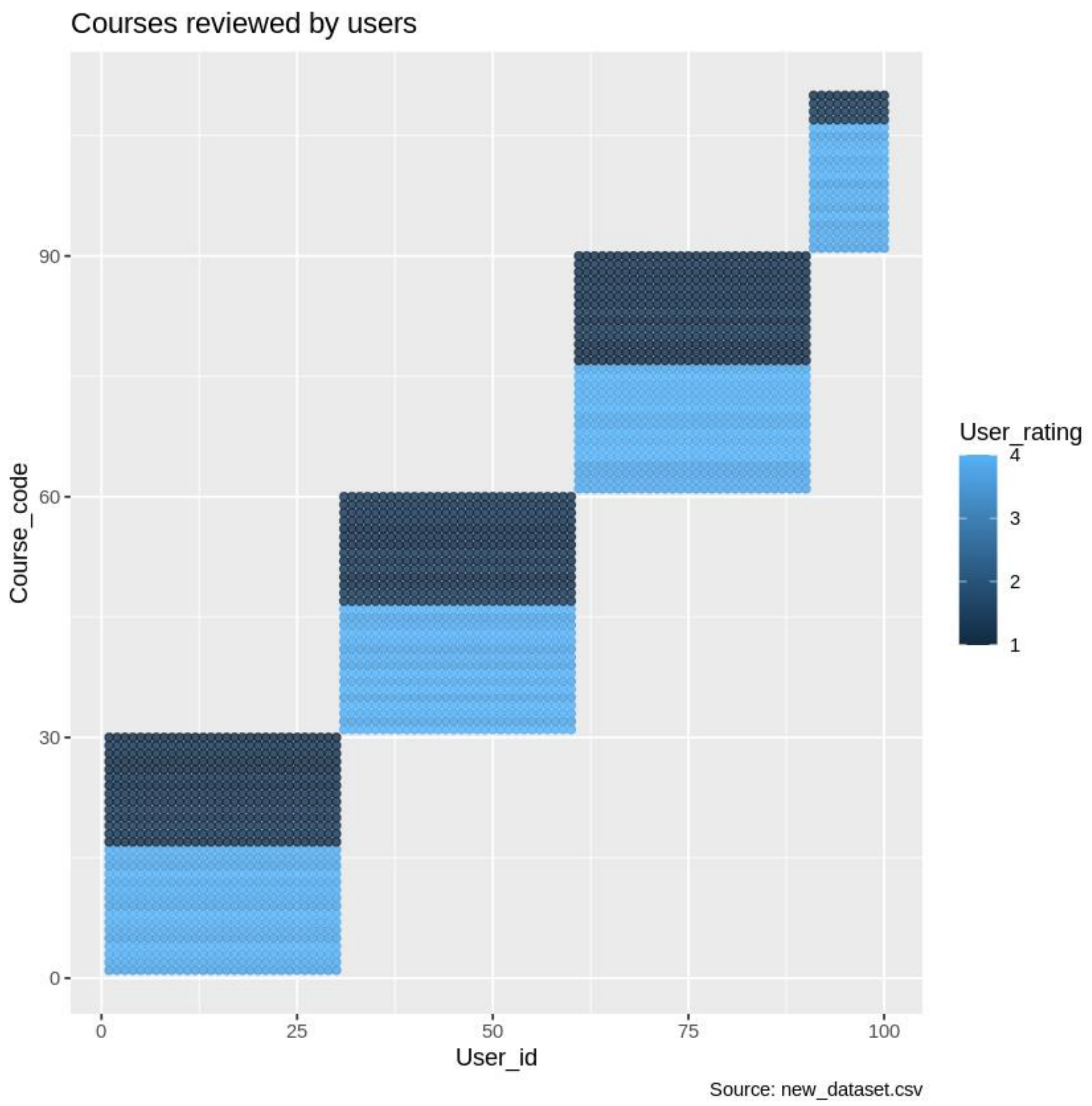
introduction data science python — by John Doe

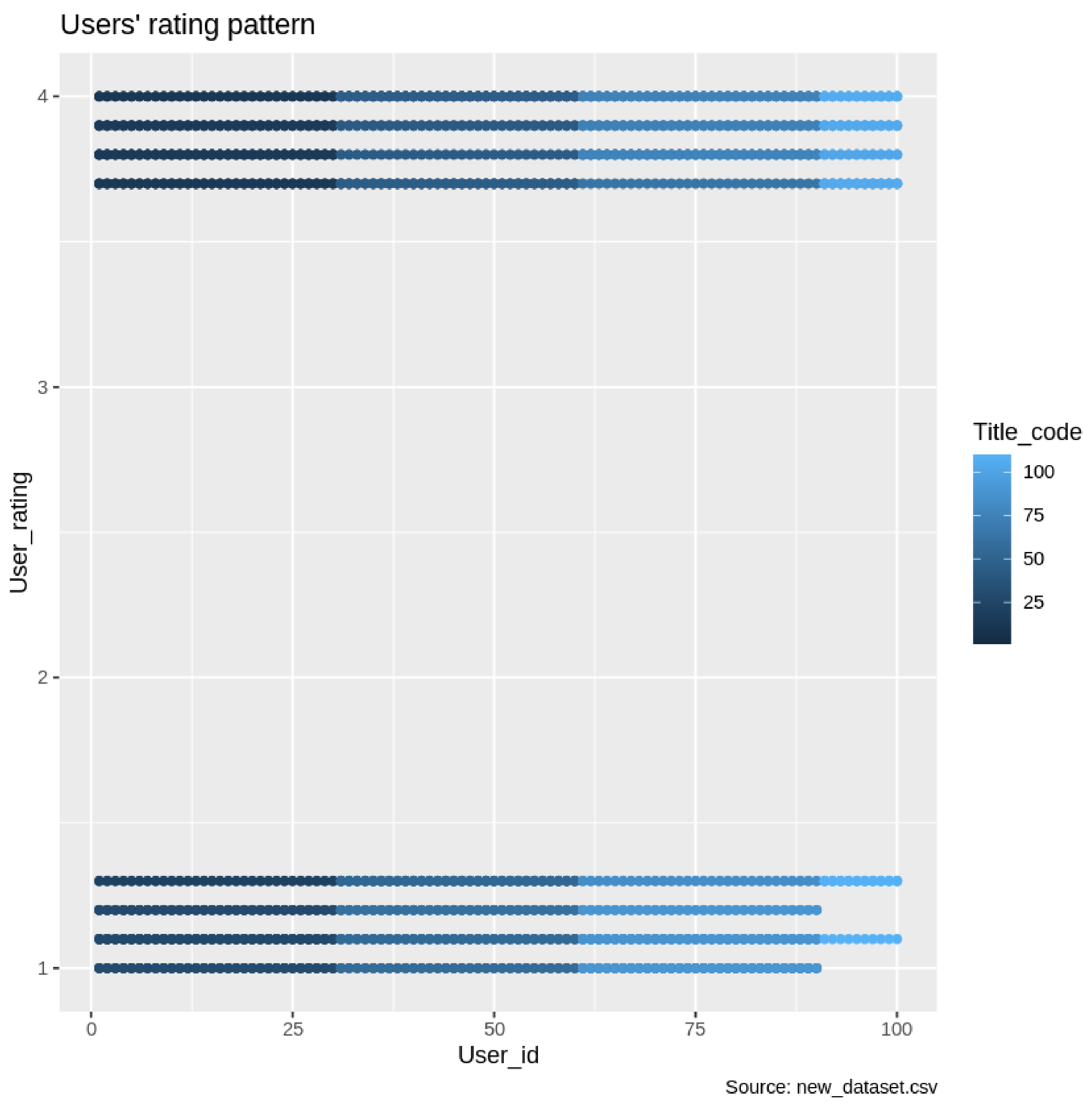
fundamental machine learning healthcare — by Modi JI

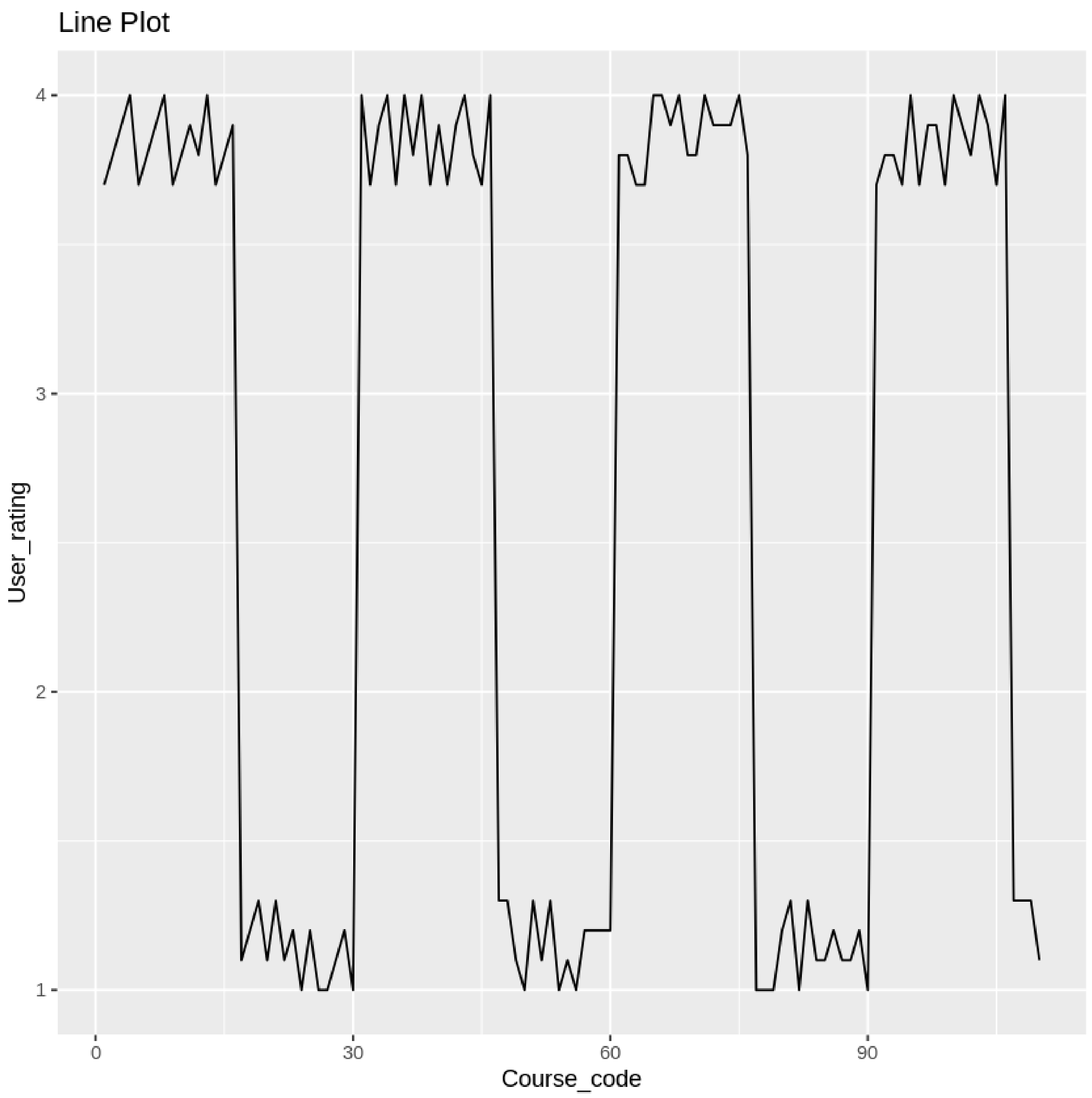
Figure 10: Search Engine Results using Content-based Filtering

ANALYTICS AND VISUALISATION OF THE DATASET USING - 'R'

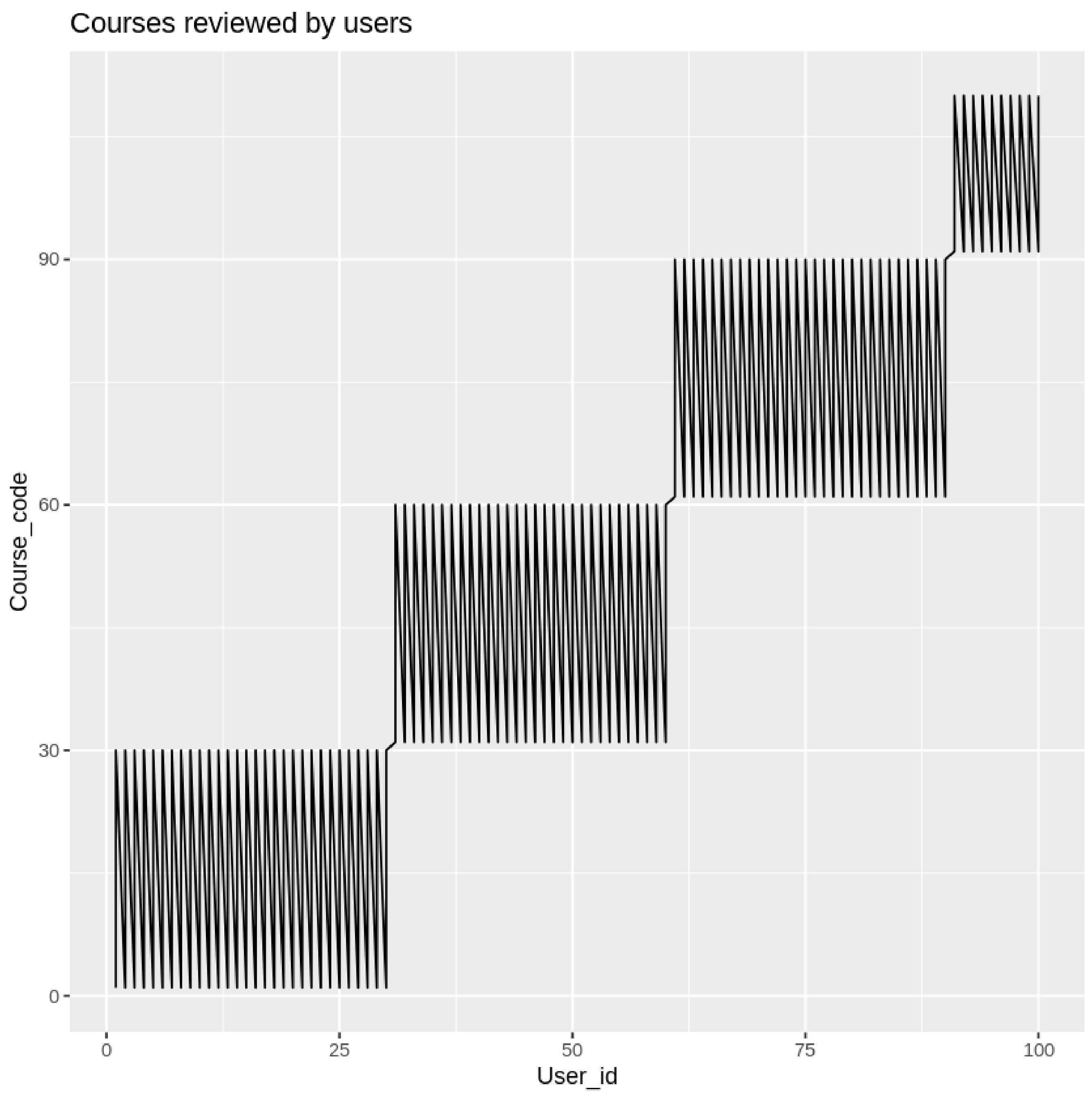


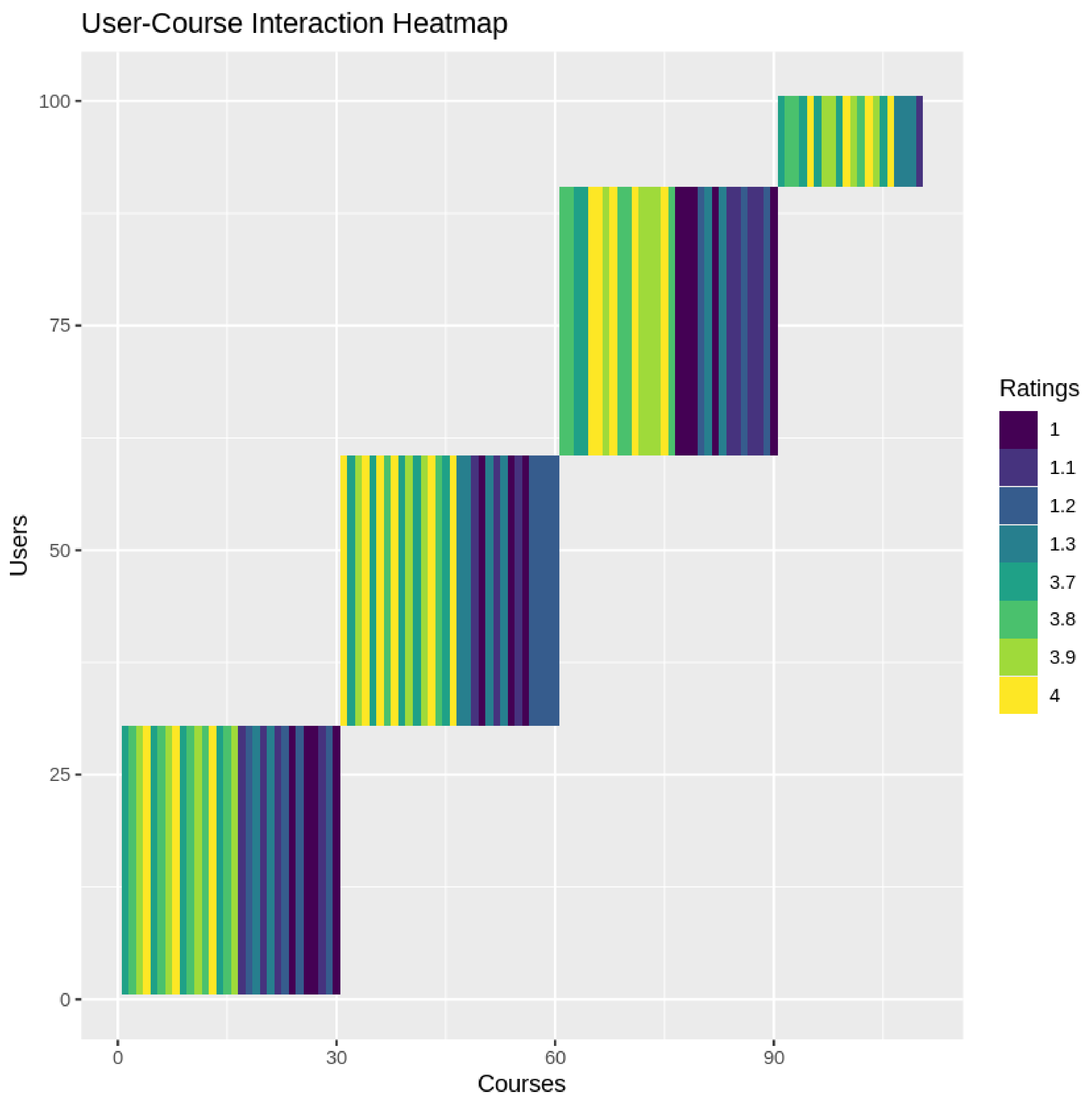

Courses reviewed by users
Course_code
90
60
30
0
0
25
50
75
100
User_id
User_rating
4
3
2
1
Source: new_dataset.csv


Users' rating pattern
User_rating
4
3
2
1
0
25
50
75
100
User_id
Title_code
100
75
50
25
Source: new_dataset.csv

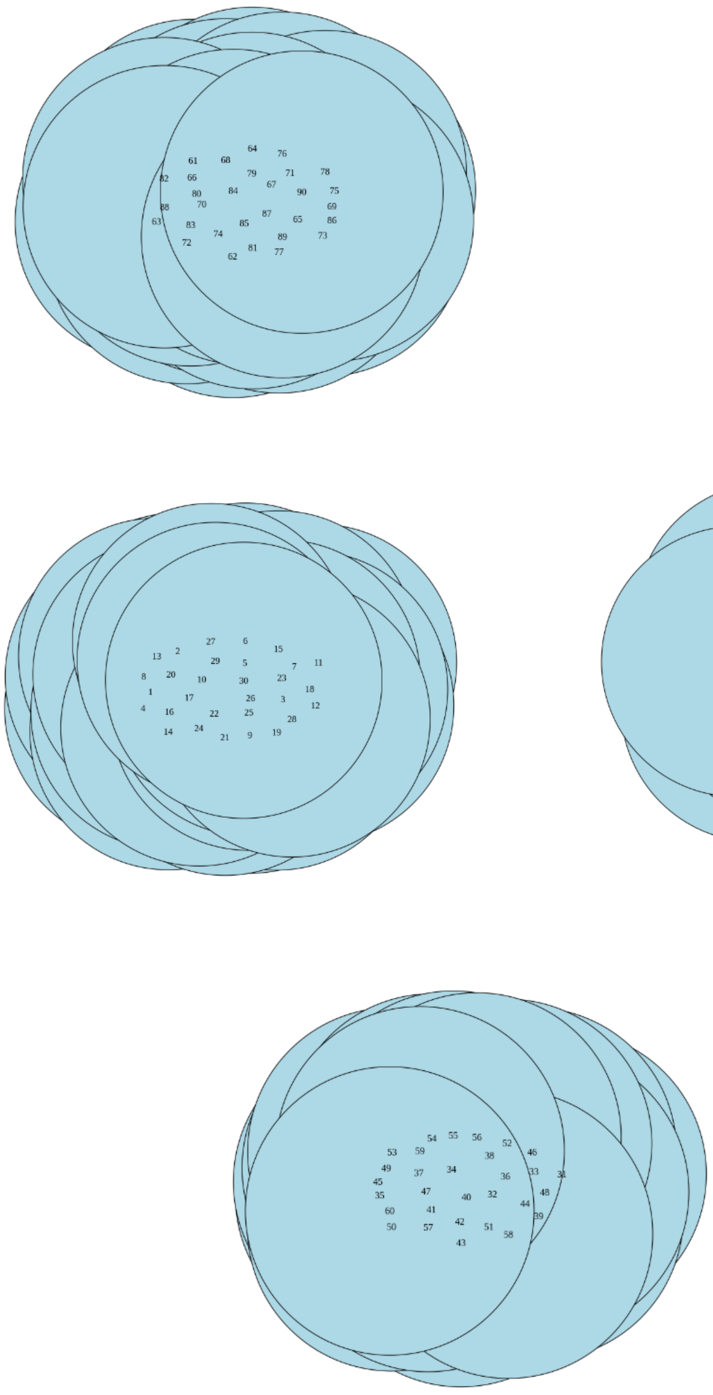


Line Plot
User_rating
1
2
3
4
Course_code
0
30
60
90


Courses reviewed by users
Course_code
User_id
0
30
60
90
0
25
50
75
100


User-Course Interaction Heatmap
Users
100
75
50
25
0
Courses
0
30
60
90
Ratings
1
1.1
1.2
1.3
3.7
3.8
3.9
4

Relation among Courses based on Users'rating



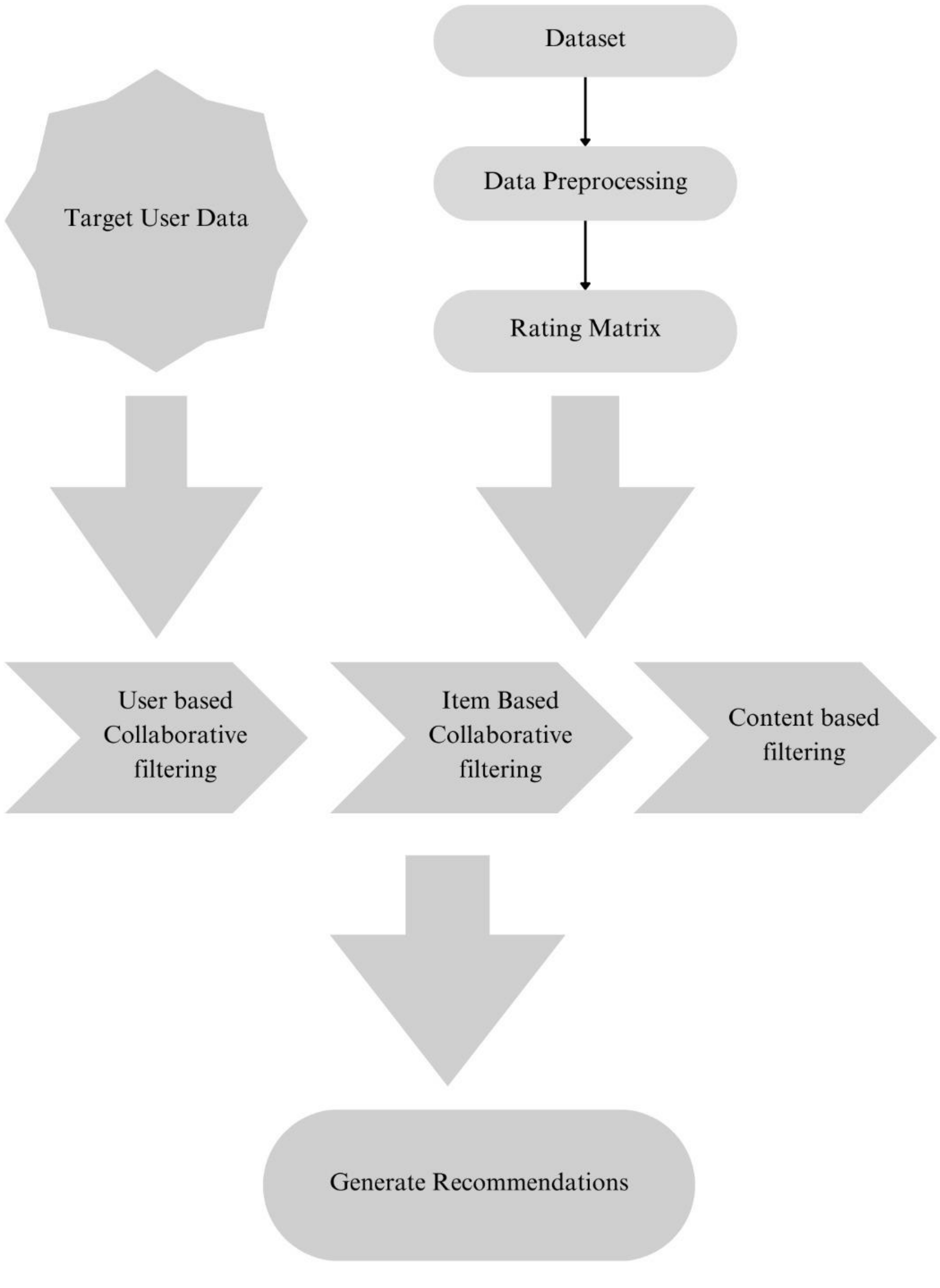
INSIGHTS FROM THE DATASET:

- The most popular course chosen is: AI For Business Specialization
- The course with the best average rating is: Applied Software Engineering Fundamentals Specialization with an average rating of 4.03
- The user-item clusters are preserved before and after pre-processing the dataset.
- It can be identified from the user_rating vs user_id plot that the ratings are only extreme values and there are close to no moderate ratings.
- In the network graph, since there is no relation(edge) between one subset of courses to another, we can interpret that the recommendations will be restricted within the clusters.

Tools and Frameworks Used:

1. **Front-end**
   - React JS (Vite)
   - HTML
   - CSS
2. **Back-end**
   - Flask
3. **Data handling**
   - NumPy
   - Pandas
   - Python
   - Scikit-Surprise
4. **Visualizations**
   - R
5. **Cloud environment**
   - Google Colab

BLOCK DIAGRAM:



RESULT:

We have implemented a system which envisions the usage of different algorithms for different components.

1. Home page leverages **user-based collaborative filtering**.
2. **Item-based collaborative filtering** suggests similar courses for the user to pursue once a user has completed a particular course.
3. **Content based system** matches the user’s requirements with courses features.

REFERENCES:

- E-Learning Course Recommender System Using Collaborative Filtering Models
  Kalyan Kumar Jena 1, Sourav Kumar Bhoi 1, Tushar Kanta Malik 1, Kshira Sagar Sahoo 2,3, N Z Jhanjhi 4,* ,Sajal Bhatia 5 and Fathi Amsaad 6.

The above paper has been referred as the base paper for implementing collaborative filtering.