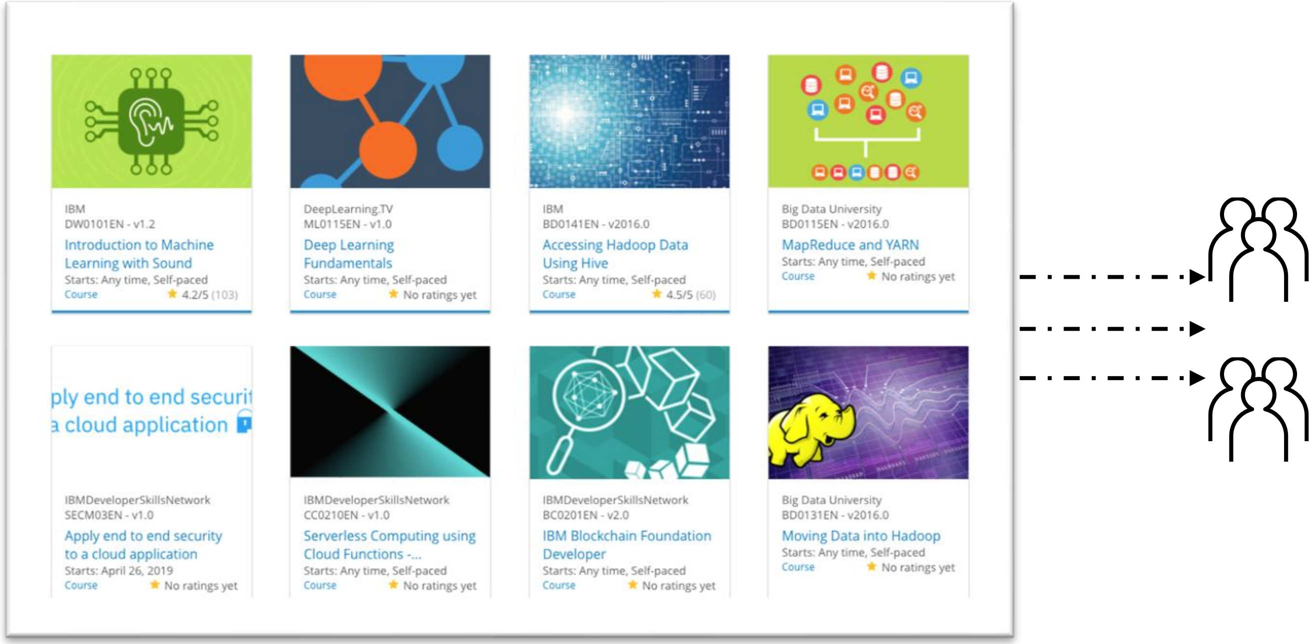
Build a Personalized Online Course Recommender System with Machine Learning



Vishal Purohit

13/07/2025

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

- Our company hosts courses on our online platform; however, we currently have no way of recommending new courses to our users.
- Our aim is to explore content and collaborative based recommender systems, how they work and to determine which of them performs best with our data.
- A secondary aim is to create a prototype app that will demonstrate the recommender system to the management team.
- Project hypothesis:
  - Our data contains 14 genres and all are computer related. As such we believe that course similarity will do better than user similarity since computer related courses generally contain transferable skills. It seems unlikely that the user base will break down into people who only are interested in one computer-based skill.

Exploratory Data Analysis

Course counts per genre

	COURSE_ID	TITLE	Database	Python	CloudComputing	DataAnalysis	Containers	MachineLearning	ComputerVision	DataScience	BigData	Chatbot	R	BackendDev	FrontendDev	Blockchain
0	ML0201EN	robots are coming build iot apps with watson ...	0	0	0	0	0	0	0	0	0	0	0	1	1	0
1	ML0122EN	accelerating deep learning with gpu	0	1	0	0	0	1	0	1	0	0	0	0	0	0
2	GPXX0ZG0EN	consuming restful services using the reactive ...	0	0	0	0	0	0	0	0	0	0	0	1	1	0
3	RP0105EN	analyzing big data in r using apache spark	1	0	0	1	0	0	0	0	1	0	1	0	0	0
4	GPXX0Z2PEN	containerizing packaging and running a sprin...	0	0	0	0	1	0	0	0	0	0	0	1	0	0

Fig 1: Table showing courses and one-hot-encoded genres

- Rather than looking at a table full of ones and zeros, we can plot a bar graph showing the number of courses belonging each of our 14 genres.
- We note that there appears to be a "staircase" pattern in the graph, where we can loosely group the data into 4 groups.
- This could tell us something about demand from employers or student choices at university.

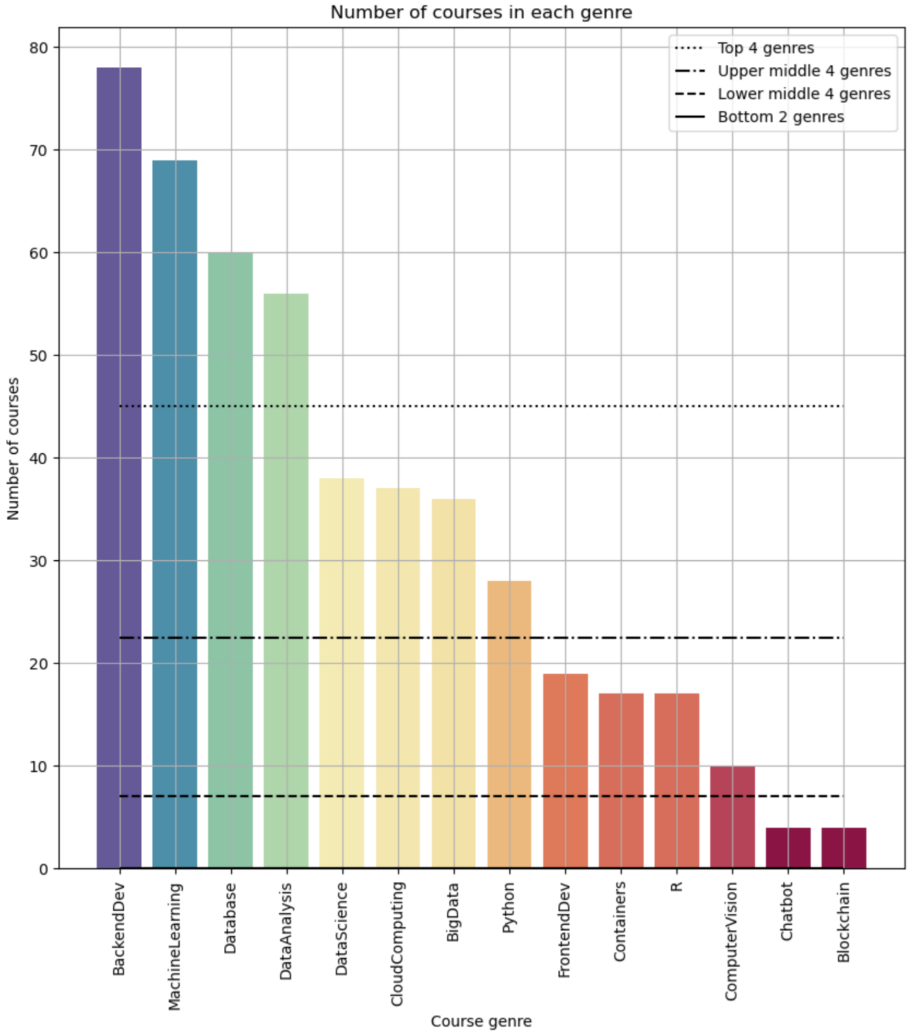


Fig 2: Number of course in each genre

User counts per genre

- Since we looked at the number of courses in each genre, let’s look at the number of users enrolled in each genre. We see that this time, “Database” is the most popular genre.
- We can plot a graph of the number of users in a genre against the number of courses in a genre. Since it costs us money to host courses on our platform, this graph gives us an idea of which genres are more cost effective than others.

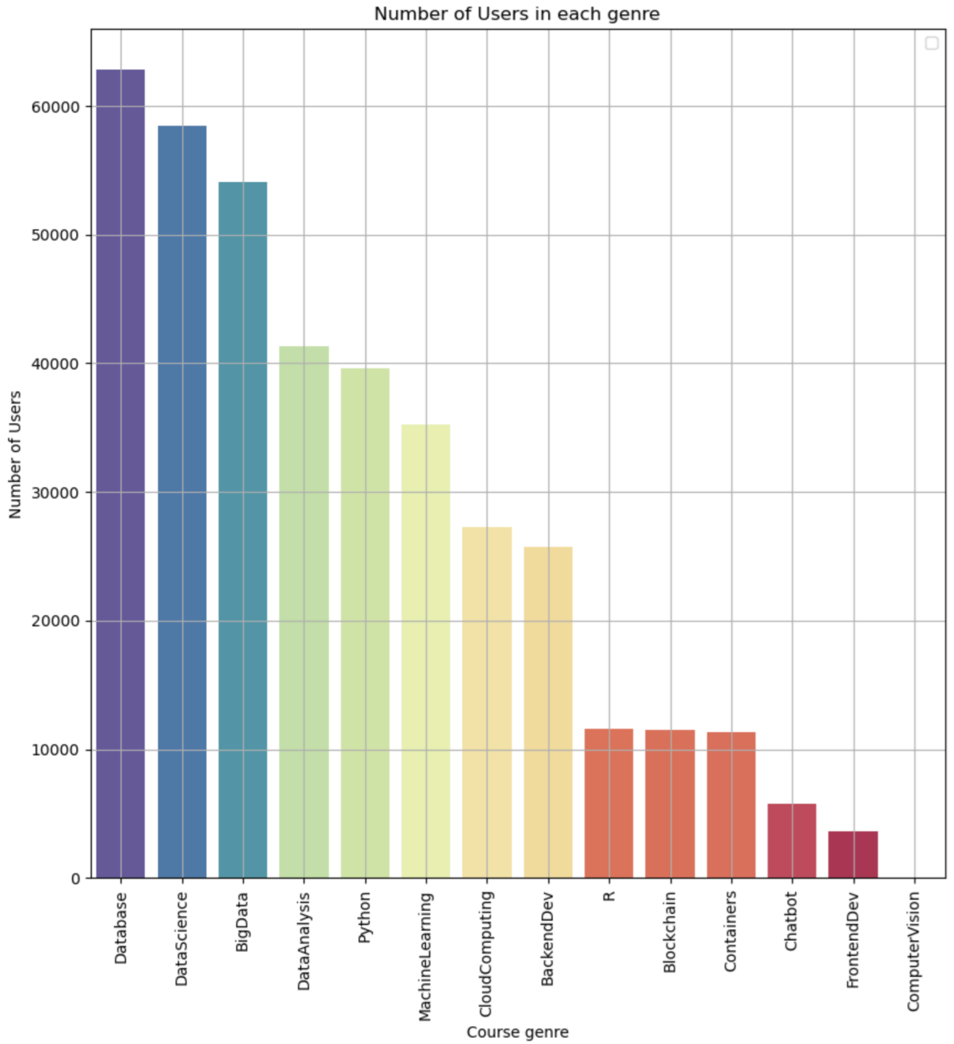


Fig 3: Number of users in each genre

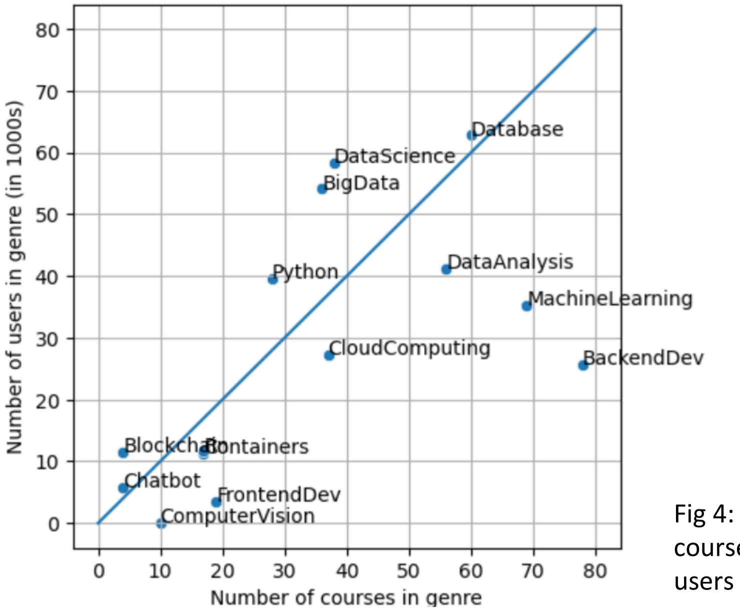


Fig 4: Scatter plot of courses in genres vs users in genres (000s)

Course enrollment distribution

- Now that we have looked at how many courses there are in each genre, we can look at the number of courses that users are enrolled in.
- We see that there is a bimodal structure (two peaks). It is possible that there are two types of user.
  - The beginner: This type could be just trying out the on-line platform, thus having a low number of course enrolls.
  - The committed: This type could have enjoyed their experience with the platform and has now committed to learning new skills with us.
- It might be worth finding out if this conjecture is true from the first group of users and pass this information on to the marketing team so that they can think of ways to retain those users.

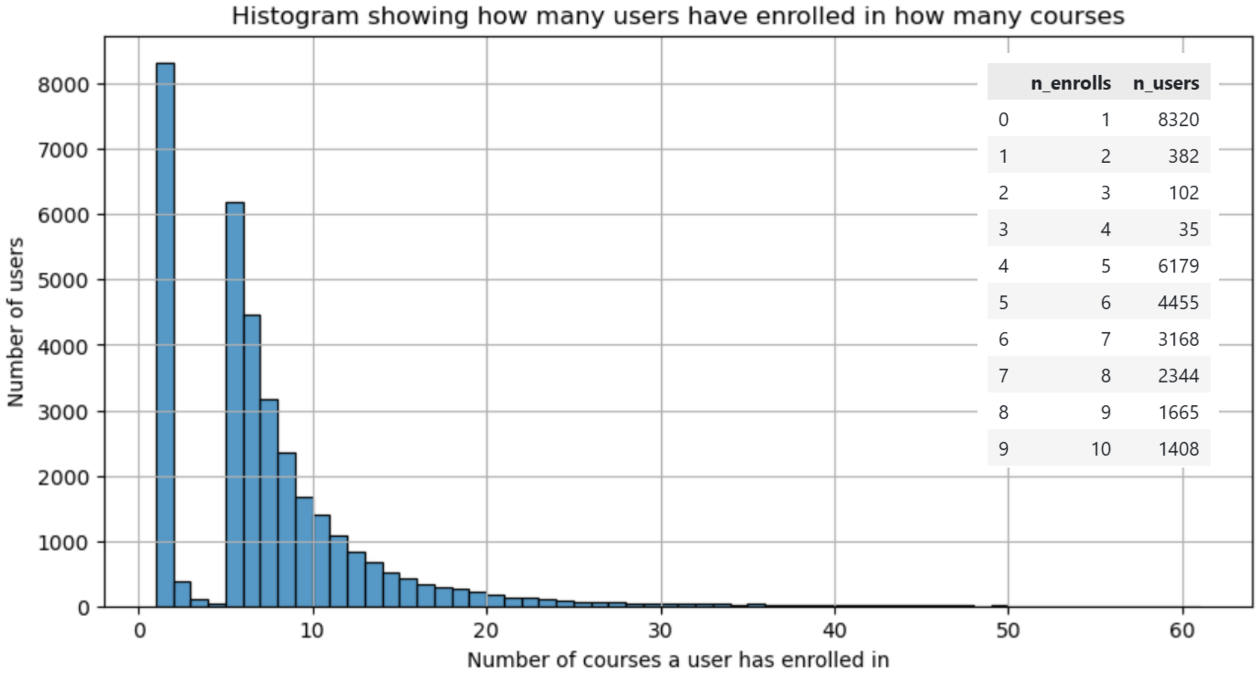


Fig 5: Histogram of how many users have enrolled into how many courses

20 most popular courses

- If we group our data by course instead of by user, we can see how many users are enrolled in a specific course.
- By ordering the resultant table, we can find the most popular courses on our platform.
- We clearly see that Data Science and Big Data with related courses on the r-language, statistics, SQL and machine learning are the most popular.
- This corroborates what we saw earlier in Fig 4; that Data Science, Big Data and Database has many users.
- The top 20 courses account for 63.3% of all users.
- The top 53 courses account for 90.4% of all users

	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

Fig 6: Number of users enrolled in each of the top 20 courses

Word cloud of course titles

- A way to get a qualitative feel for the data is through a word cloud.
- By breaking up the course titles into individual words; we can create a picture, where the size that a word appears in the picture is proportional to the number of times it appears in the titles.
- We can see that Data Science, Data, Python and Machine Learning are very big. This means that they are very common words in our course titles.

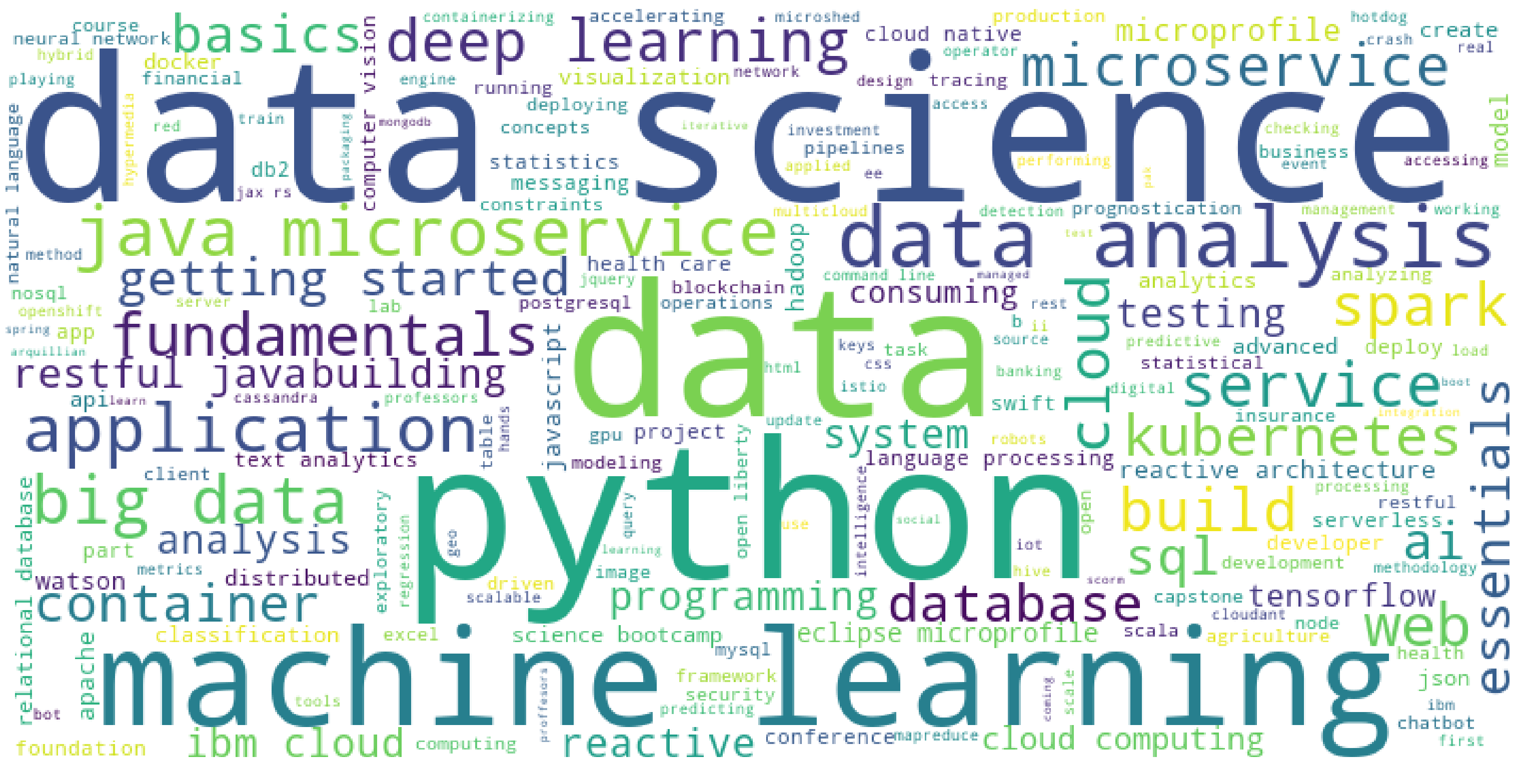
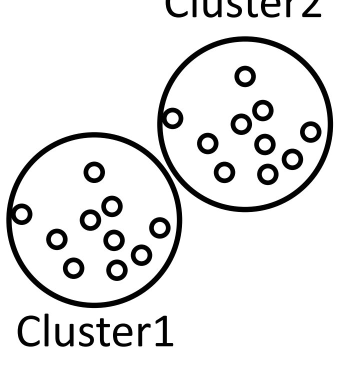


Fig 7: Word Cloud of Course Titles

Content-based Recommender System using Unsupervised Learning



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

- Users have taken some courses which they have rated.
- Courses belong to different genres.
- Combining these data sets, we can create a user profile that says how strongly they like different genres.
- We have a list of all the courses on our platform, from which we can subtract the courses the user has already taken, leaving us with courses unknown to the user.
- Combining the user profiles with the unknown course genres, we can create a score of how much a user will like the unknown courses based on how much their profile says they like the genres those courses are a part of.
- If the score for a course is above a set threshold, we recommend the course to that user.
- If not, we do not recommend the course.

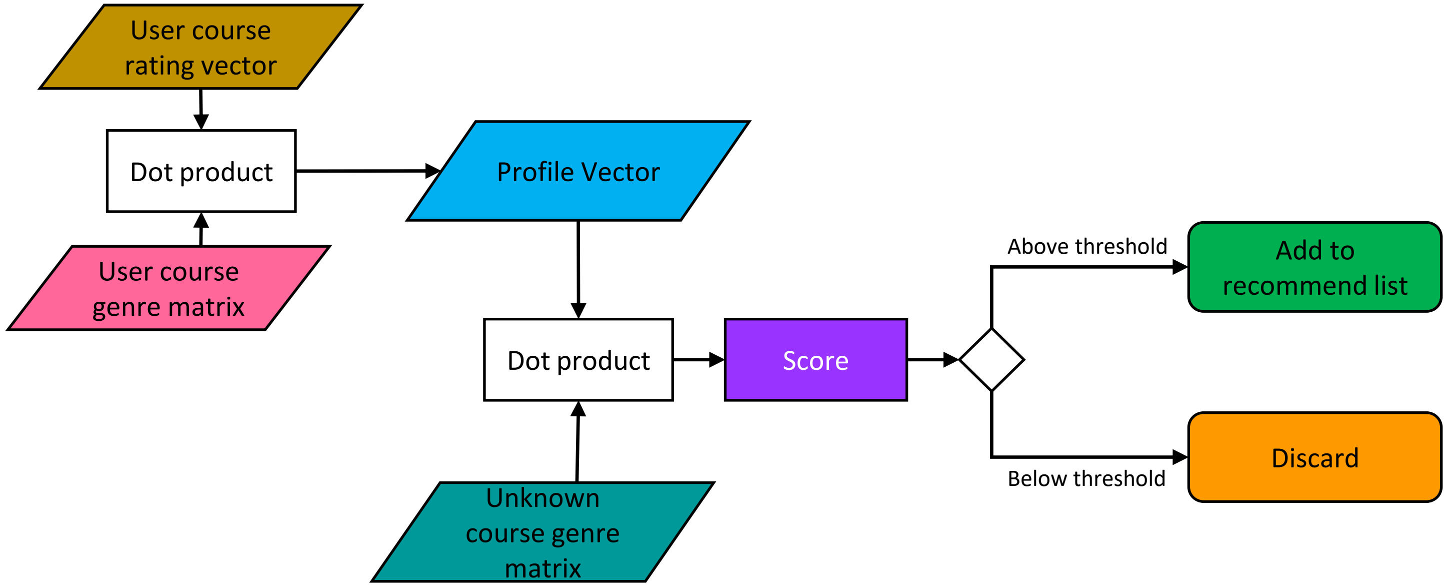


Fig 8: Flowchart of content base recommender system

Evaluation results of user profile-based recommender system

The recommendations are based on the dot product of vectors and matrices so there are no hyperparameters in the model. The threshold for recommendation is set at a score of 10.

- On average, 44.26 courses have been recommended per user (in the test user dataset) with a median of 33 courses.
- However, 27.24% of users have zero recommendations as their scores are too low. While some users were given 267 recommendations.
- If we have the resources, perhaps setting the threshold to zero and recommending to the users an ordered list or their own top 10 might be better.

- The course recommended to the most people was TA0106EN: Text Analytics at scale. However, we see that it is not the most highly rated course. Excourse72 and Excourse73 have less mass appeal, but are rated more highly.

	n_Users	Mean_Score	COURSE_ID	TITLE
0	17390	21.082806	TA0106EN	text analytics at scale
1	15656	21.823263	excourse21	applied machine learning in python
2	15656	21.823263	excourse22	introduction to data science in python
3	15644	21.538098	GPXX0IBEN	data science in insurance basic statistical a...
4	15603	21.777222	ML0122EN	accelerating deep learning with gpu
5	15062	19.564533	excourse04	sql for data science
6	15062	19.564533	excourse06	sql for data science capstone project
7	14689	21.441215	GPXX0TY1EN	performing database operations in the cloudant...
8	14464	28.835730	excourse73	analyzing big data with sql
9	14464	28.835730	excourse72	foundations for big data analysis with sql

Fig 9: Table of top 10 courses from user profile-based recommender system

Flowchart of content-based recommender system using course similarity

- We can break up course titles and their descriptions into individual words.
- This can be used to create a dictionary that maps a word to a number and vice versa
- Using the dictionary, we can make create a list of how many unique words there are and how many times they appear in a particular course title and description
- Small words like "the" and "for" are not helpful is assessing similarity so we can get rid of them to create a table detailing course IDs, the words in them and how many times they appear.
- We then create a pivot table with course ID as rows, words as columns and word count as values.
- Individual rows form BOW feature vectors, which can be compared to each other using a similarity measure. In this case we used the cosine similarity. Comparing all rows to each other gives a look-up table (similarity matrix) showing how similar course titles and descriptions are to one another.
- Courses that users take are then checked against others using the look-up table and if the similarity is above a certain threshold (we chose 0.6), then we add them to the recommendation list.
- If they are below the threshold then they are not added.

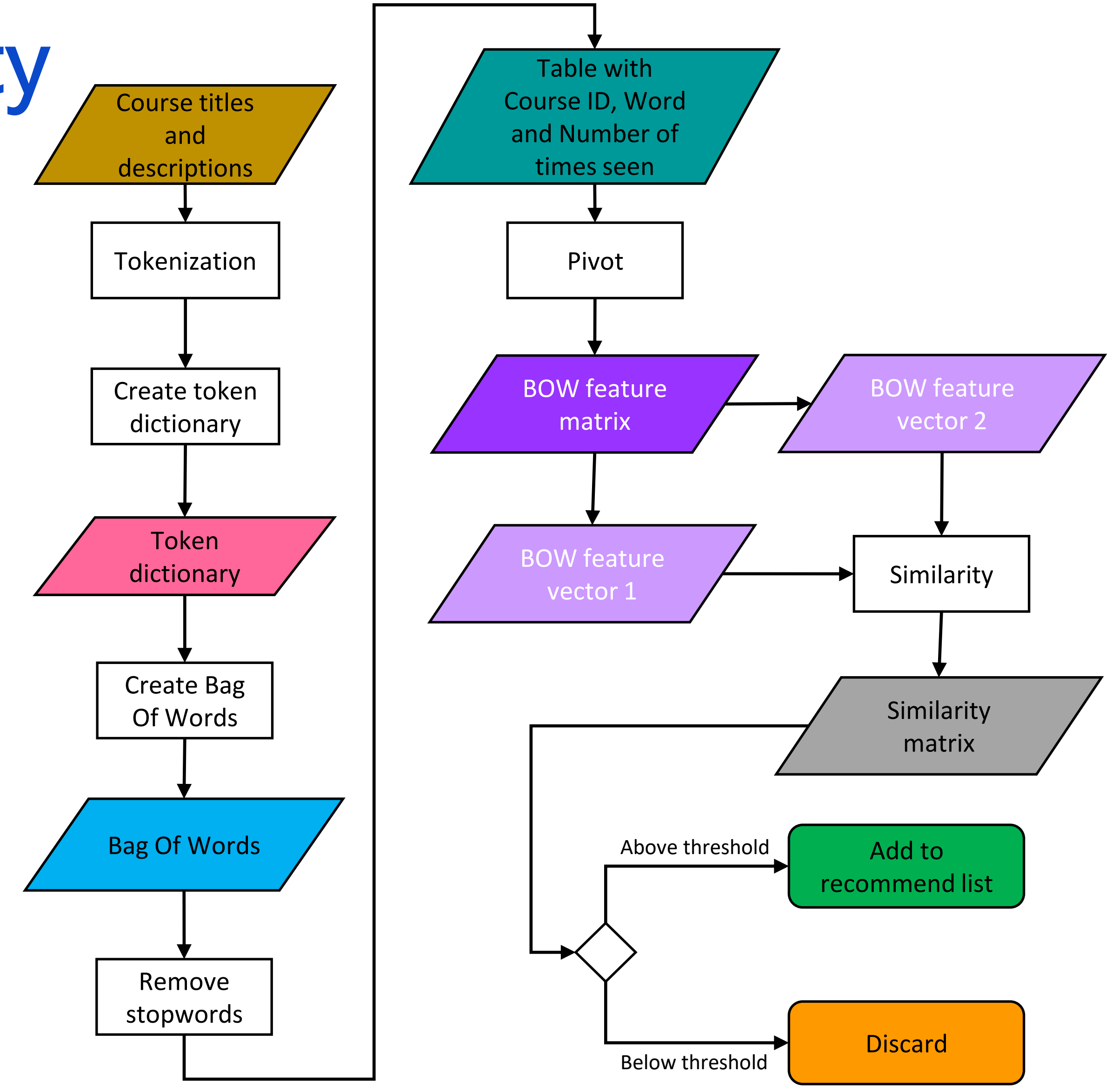


Fig 10: Flowchart of course similarity-based recommender system

Evaluation results of course similarity based recommender system

The recommendations are based on the similarity scores of profile vectors so there are no hyperparameters in the model. The threshold for recommendation is set at a cosine similarity score of 0.6.

- On average, 8.54 courses have been recommended per user (in the test user dataset) with a median of 8 courses.
- However, 12.51% of users have zero recommendations as their course similarity scores are too low. While some users were given 30 recommendations.
- If we have the resources, perhaps setting the threshold to zero and recommending to the users an ordered list or their own top 10 might be better.

The course recommended to the most people was DS0110EN: Data Science with open data.

	n_Users	Mean_Score	COURSE_ID	TITLE
0	15003	0.725971	DS0110EN	data science with open data
1	14937	0.647499	excourse62	introduction to data science in python
2	14937	0.647499	excourse22	introduction to data science in python
3	14641	0.695299	excourse63	a crash course in data science
4	14641	0.639832	excourse65	data science fundamentals for data analysts
5	13551	0.618134	excourse68	big data modeling and management systems
6	13512	0.656375	excourse72	foundations for big data analysis with sql
7	13291	0.708214	excourse67	introduction to big data
8	13291	0.650071	excourse74	fundamentals of big data
9	12497	0.623544	BD0145EN	sql access for hadoop

Fig 11: Table of top 10 courses from course similarity-based recommender system

Flowchart of clustering-based recommender system

- Starting from our user profile vectors we need to scale the data, both Principal Component Analysis (PCA) and K-Nearest Neighbours (KNN) use distance metrics, so the data must be on the same scale.
- KMeans clustering uses distances between points and consequentially suffers from the curse of dimensionality, so we use PCA to reduce the dimensionality of the dataset. This transforms the genres of our profile vectors into linear combinations of genres.
- With our reduced dimensionality dataset, we can perform KMeans clustering, searching of different numbers of clusters (k) to find "good" clusters, which we judge using the elbow method and the silhouette score. This allows us to assign users to different clusters.
- Once users and their courses have been clustered, we can look at the most popular courses in each cluster. In this case, we measure popularity by number of enrollments. We set a threshold of 100 to limit results.
- Those below 100 enrollments are discarded.
- For each user, we can recommend popular courses in their cluster, removing those that the user has already enrolled in.

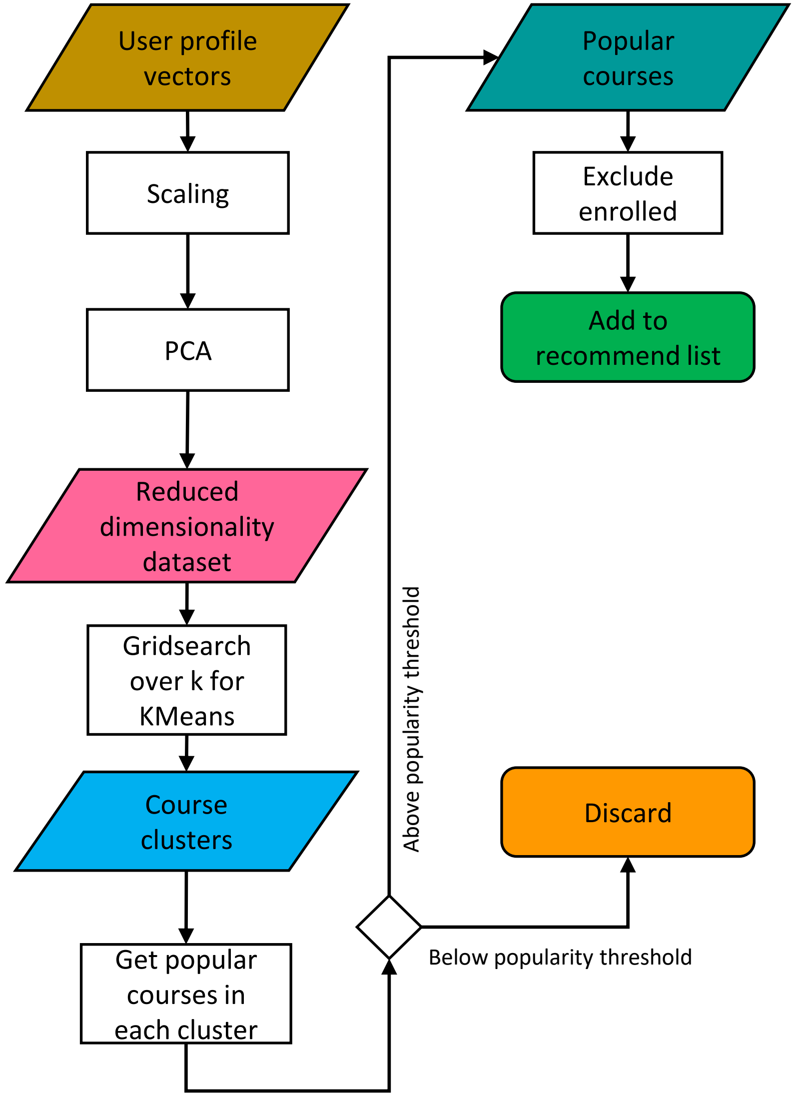


Fig 12: Flowchart of clustering-based recommender system

Evaluation results of clustering-based recommender system

We used 9 PCA components, 12 clusters (k=12) and 100 enrollments as the threshold for course popularity.

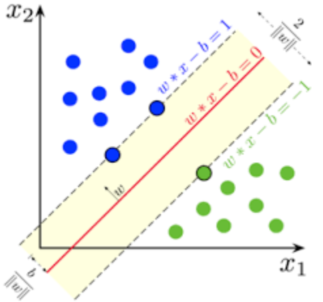
- On average, 26.75 courses have been recommended per user (in the test user dataset) with a median of 30 courses.
- Unlike our previous recommender attempts, only 0.62% of users have zero recommendations. We see that there is a cluster (4) that has low numbers of enrollment and that is filtered out by the 100-user enrollment threshold.
- If we have the resources, perhaps setting the threshold to zero and recommending to the users an ordered list or their own top 10 might be better.

The course recommended to the most people was ST010EN: Statistics 101. There is no score this time as we are looking popular courses within clusters.

	USER	COURSE_ID	TITLE
0	28338	ST0101EN	statistics 101
1	28108	RP0101EN	r for data science
2	27077	ML0115EN	deep learning 101
3	26860	CL0101EN	ibm cloud essentials
4	25664	DS0103EN	data science methodology
5	25580	CC0101EN	introduction to cloud
6	23292	BD0111EN	hadoop 101
7	21710	WA0101EN	watson analytics 101
8	21654	SC0101EN	scala 101
9	20721	DS0301EN	data privacy fundamentals

Fig 13: Table of top 10 courses from clustering-based recommender system

Collaborative-filtering Recommender System using Supervised Learning



Flowchart of KNN based recommender system

- Starting with the User-item interaction matrix, which tells you how users rated courses, we encode the users and items by turning them into numbers with a look up dictionary.
- The next step depends on the type of collaborative filtering we wish to perform. In our case we found that item-based filtering gave a better root mean squared error (RMSE).
- To recommend courses to users, item-based filtering uses the similarity between items (courses). We used the cosine similarity measure.
- We go through the list of courses, find the k most similar courses to each of them and how other users rated those similar courses. We then multiply the ratings of the similar courses by how similar we found the courses to be (cosine similarity). The value of k is found by searching over many values and picking the one that minimizes the RMSE.
- Using the value of k that we found, we can then calculate the predicted rating for an item by adding up the multiplied user ratings and similarities and then dividing by them by the sum of the similarities.
- If the ratings are predicted to be high, we can recommend them
- If not, then we don't recommend them.

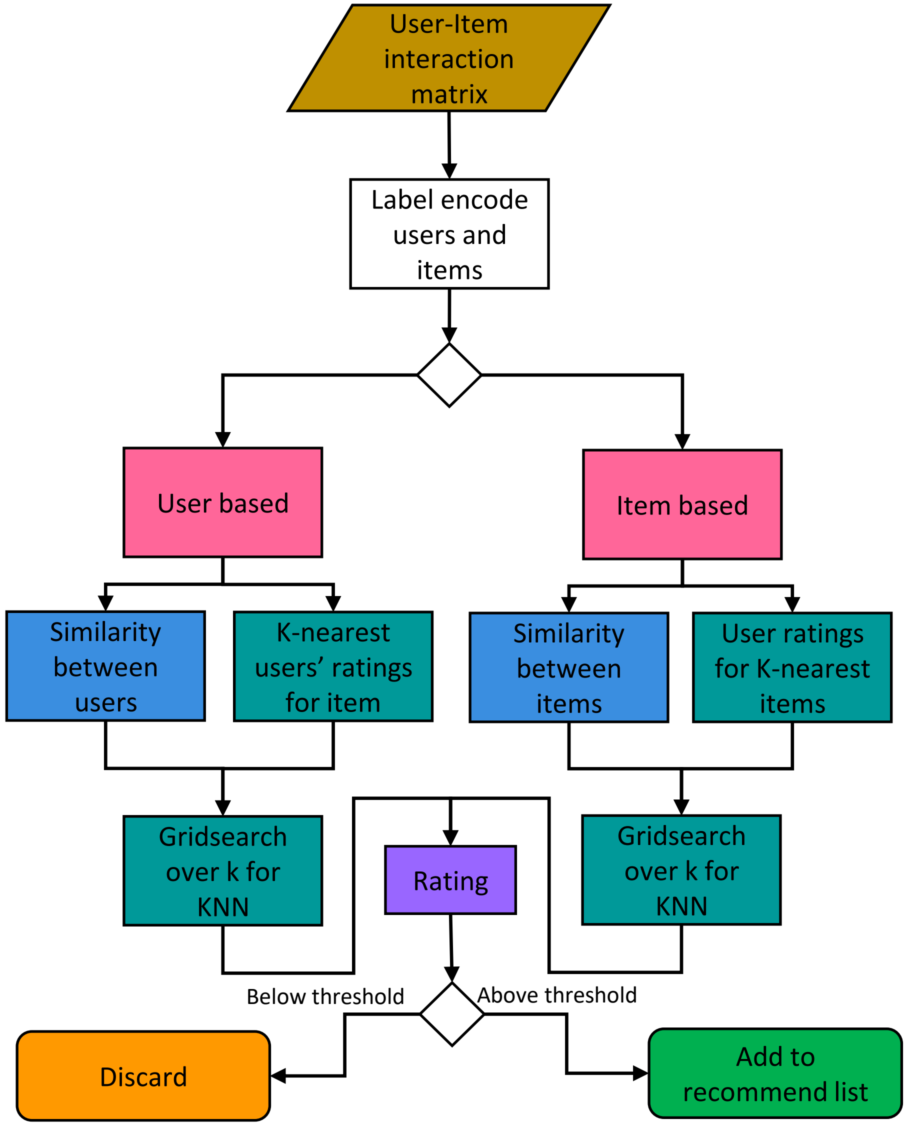


Fig 14: Flowchart of KNN-based recommender system

Flowchart of NMF based recommender system

- Non-negative matrix factorization is about breaking up a large matrix into two smaller matrices with latent features. We begin with the User-item interaction matrix, which tells us how users rated different courses.
- We then create two new matrices. The first matrix is the User matrix, the rows of which are the users and columns of which represent our latent features. We chose a latent feature size of 100.
- The second matrix is the item matrix, the rows of which represent our 100 latent features and the columns of which represent the items (courses).
- The dot product (matrix multiplication) of the two matrices relate the users from the first matrix to the items in the second matrix giving us back a predicted rating. At first, this predicted rating will be gibberish, because we haven't said how the latent features relate to the users or items. That is something we need to find by performing gradient descent over the mean squared error between the true and predicted rating.
- Once we have minimized the error, we have our best predictions for the ratings.
- If the predicted ratings are high, then we recommend the items (courses) to the users.
- If not, them we don't recommend them.

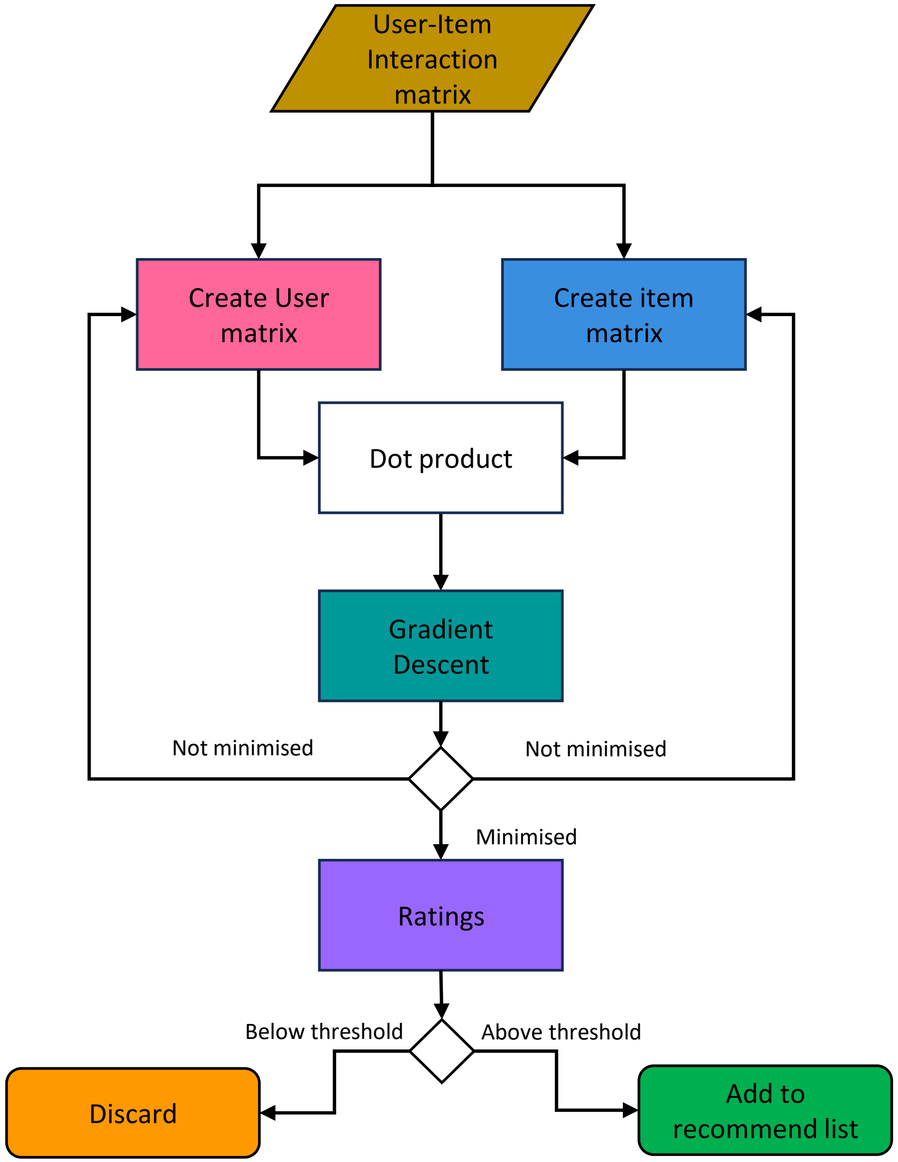


Fig 15: Flowchart of NMF-based recommender system

Flowchart of Neural Network Embedding based recommender system

- Much like how NMF used latent features to encode information about how users can be related to items, we are going to create an embedding layer for users and items and use a neural network to predict user ratings.
- We create 2 vectors, one for users and one for items (courses)
- We then pass each of them into their own embedding layer (which we can control the size of) just like we did for the latent features of NMF
- These embedding layers encode information about how the users and items interact and they are fed into a dense layer, which helps account for non-linear relationships.
- From here we compute the dot product of the dense layers and send them through a sigmoid function to make sure the predictions come out in a 0 – 1 range. Since the ratings came in the range 3 – 5, we need to multiply the output from the neural network by 2 and then add 3.
- Now that we have our predicted ratings, if they are high, we can recommend those courses
- If the predicted ratings are low, then we don't recommend the course to those users.

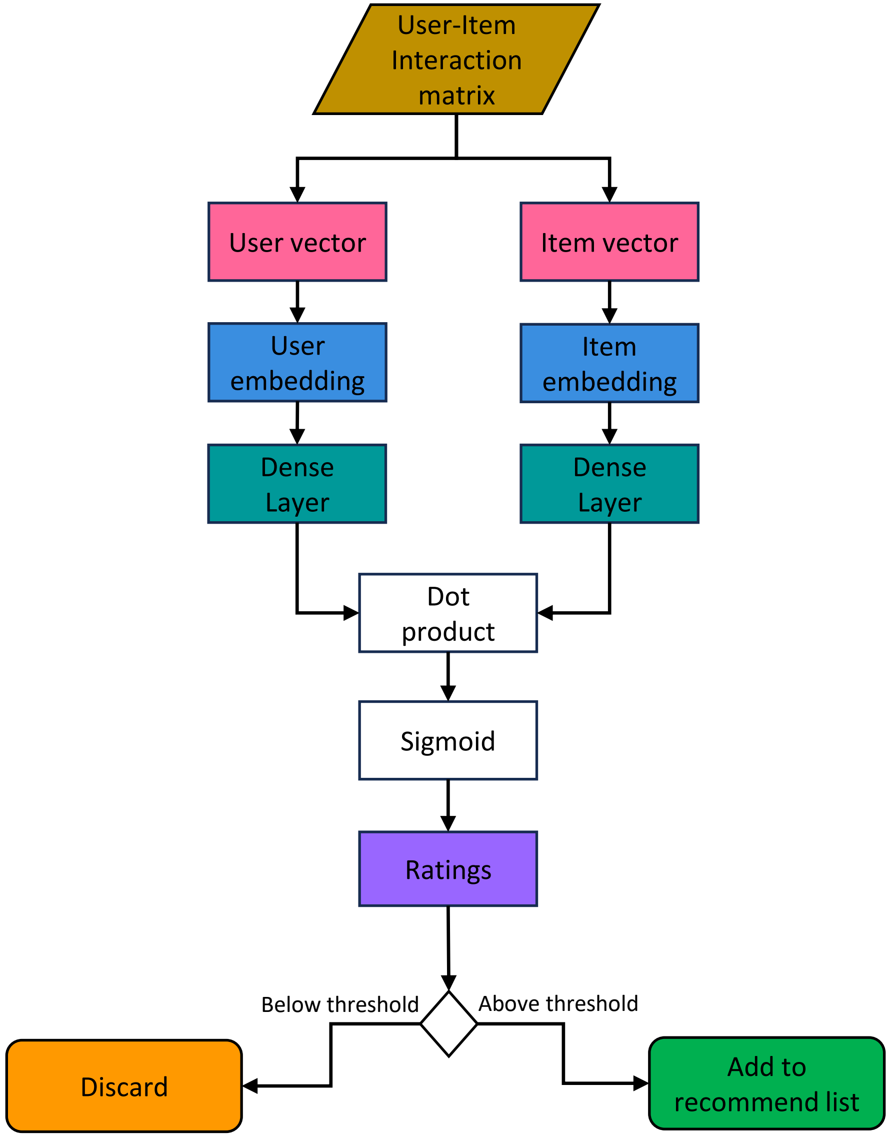


Fig 16: Flowchart of Neural Network Embedding-based recommender system

Compare the performance of collaborative-filtering models

Collaborative filtering methods did poorly in our testing. We know that our rating system is between 3 and 5. RMSE is minimized by predicting 4 for all items (courses).

Only the neural network was able to perform the same as predicting all 4s and that was by also predicting all 4s.

Using Machine learning classification algorithms on the embedding weights did very poorly equaling the RMSE for completely random predictions.

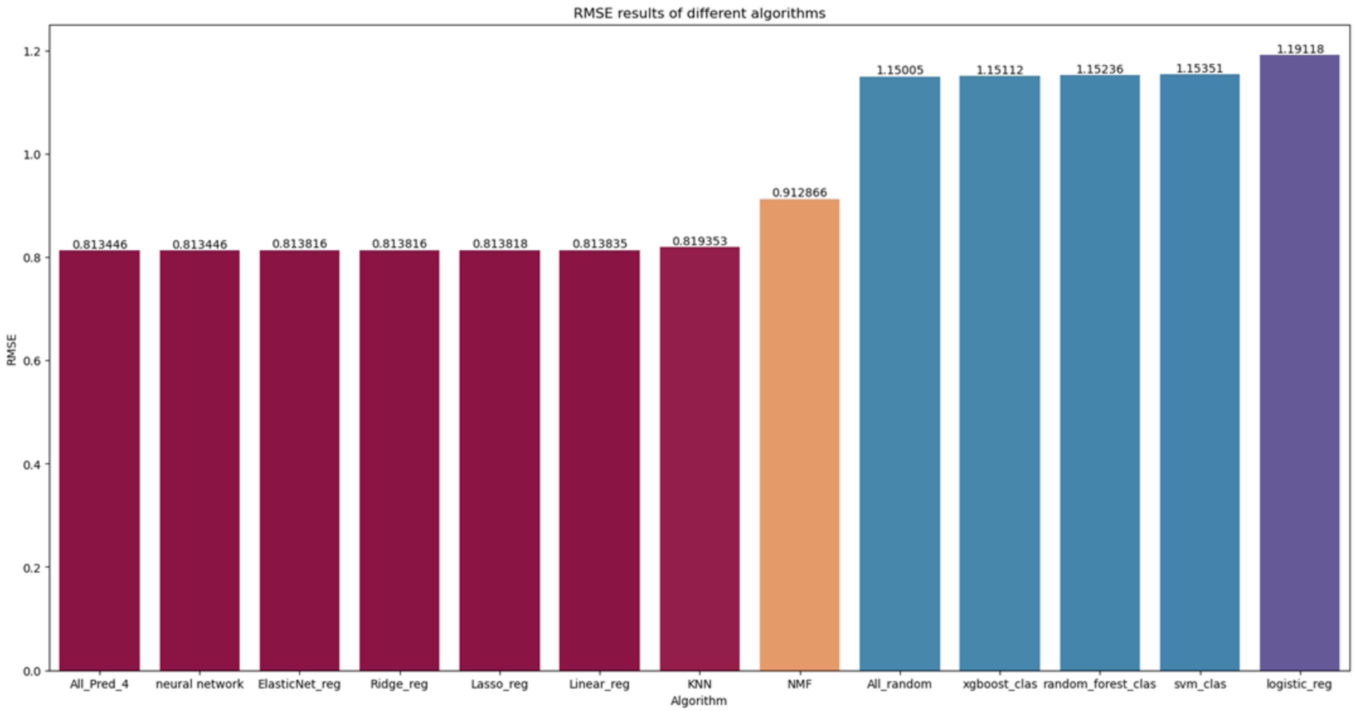
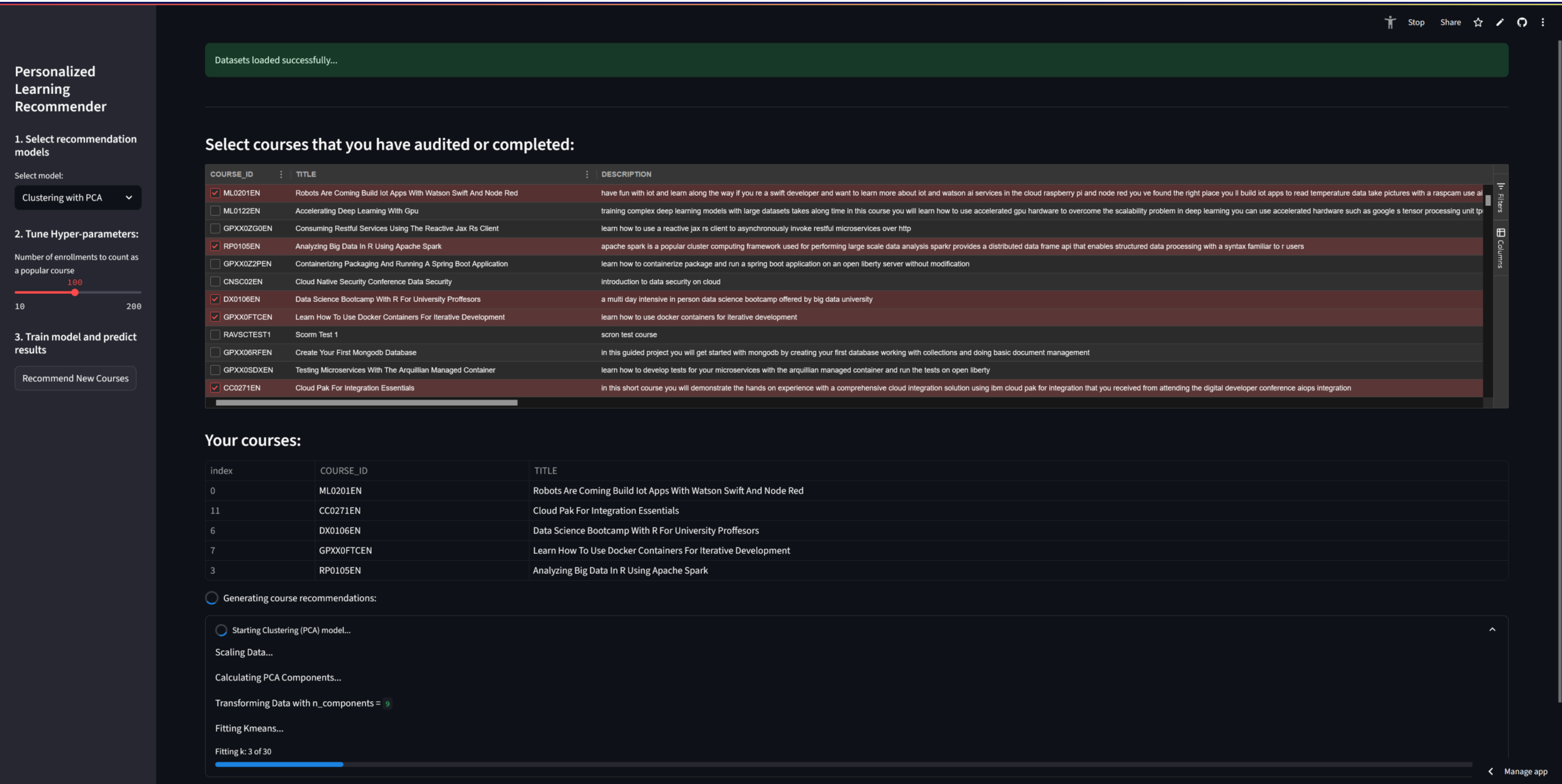


Fig 17: Bar chart showing the RMSE for different algorithms for collaborative-filtering recommender systems

Streamlit course recommender system app (screenshot 1)
https://app2-8iudwu6jwezqbwdhtgsh24.streamlit.app/



Streamlit course recommender system app (screenshot 2)
https://app2-8iudwu6jwezqbwdhtgsh24.streamlit.app/

Share

Personalized Learning Recommender

1. Select recommendation models

Select model:

Regression with Em...

2. Tune Hyper-parameters:

Regression Algorithm

Ridge | Lasso | ElasticNet

Your selected option: ElasticNet.

3. Train model and predict results

Recommend New Courses

RAVSCTEST1	Scorm Test 1	scron test course
GPXX06RFEN	Create Your First Mongodb Database	in this guided project you will get started with mongodb by creating your first database working with collections and doing basic document management
GPXX0SDXEN	Testing Microservices With The Arquillian Managed Container	learn how to develop tests for your microservices with the arquillian managed container and run the tests on open liberty
✓ CC0271EN	Cloud Pak For Integration Essentials	in this short course you will demonstrate the hands on experience with a comprehensive cloud integration solution using ibm cloud pak for integration that you received from attending the digital developer conference aiops integration

Your courses:

index	COURSE_ID	TITLE
0	ML0201EN	Robots Are Coming Build Iot Apps With Watson Swift And Node Red
11	CC0271EN	Cloud Pak For Integration Essentials
6	DX0106EN	Data Science Bootcamp With R For University Proffesors
7	GPXX0FTCEN	Learn How To Use Docker Containers For Iterative Development
3	RP0105EN	Analyzing Big Data In R Using Apache Spark

✓ Model complete!

ElasticNet ...

Cross validation over parameters...

Best Parameters:

```
{
"alpha" : 2068.0889542023733
"l1_ratio" : 0.00010501839692381458
}
```

Refitting with best Parameters...

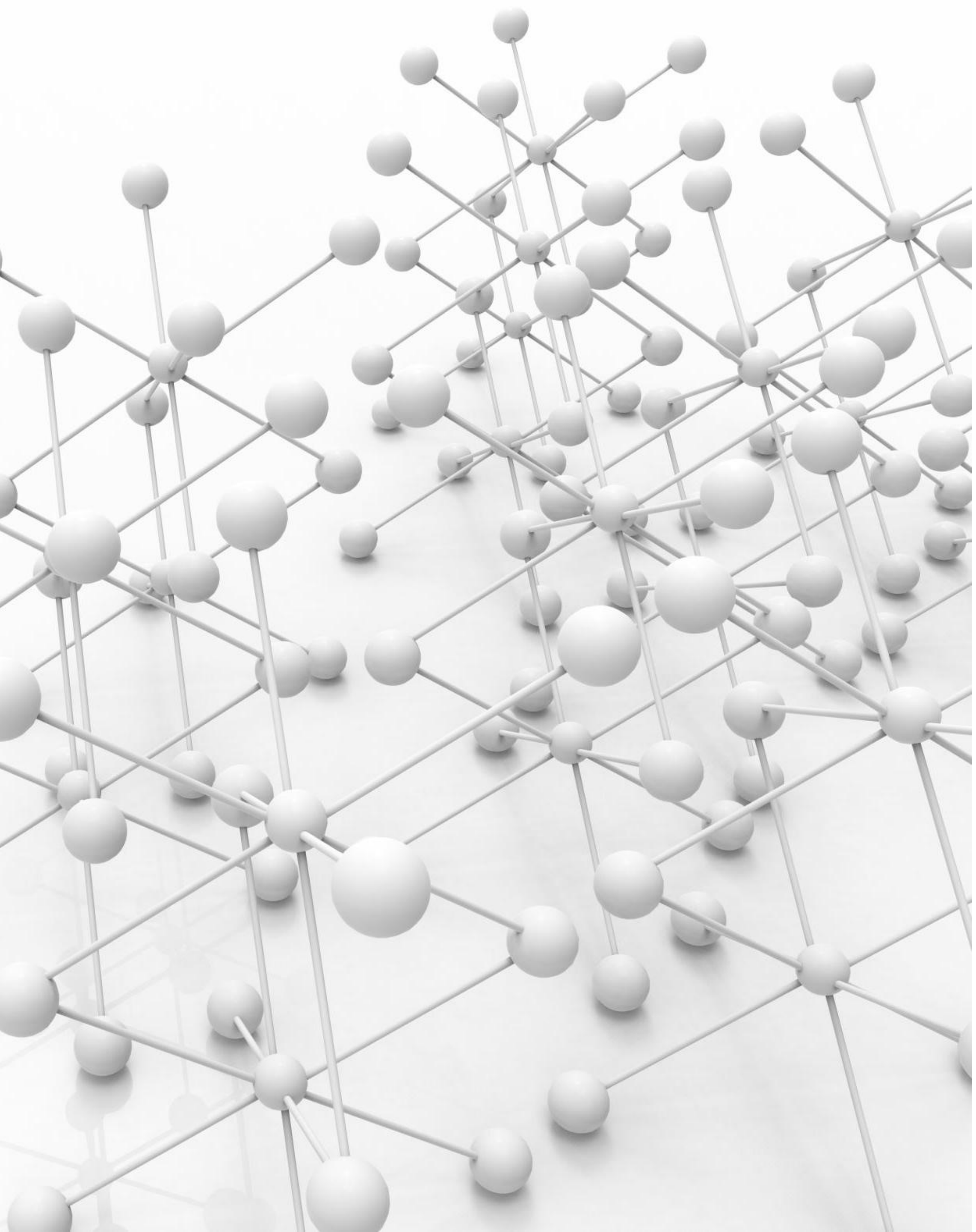
Predicting results...

Outputting...

Recommendations generated!

	SCORE	TITLE	DESCRIPTION
0	3.9984	Fundamentals Of Big Data	welcome to fundamentals of big data the fourth course of the key technologies of data analytics specialization by enrolling in this course you
1	3.9984	Machine Learning For All	machine learning often called artificial intelligence or ai is one of the most exciting areas of technology at the moment we see daily news stori
2	3.9984	Natural Language Processing With Sequence Models	in course 3 of the natural language processing specialization you will a train a neural network with glove word embeddings to perform sentim
3	3.9984	Creating Asynchronous Java Microservices Using Microprofile Reactive Messaging	learn how to write reactive java microservices using microprofile reactive messaging
4	3.9984	Apply End To End Security To A Cloud Application	this mini course walks you through key security services available in the ibm cloud,ñ¢ catalog and how to use them together an application tha
5	3.9984	Cloud Computing Foundations	welcome to the first course in the building cloud computing solutions at scale specialization in this course you will learn how to build foundati
6	3.9984	Monitoring The Metrics Of Java Microservices Using Eclipse Microprofile Metrics	you will explore how to provide system and application metrics from a microservice with microprofile metrics

Manage app



Conclusions

- Content based recommender systems performed better than collaborative-filtering recommender systems for our dataset. As can be seen from Fig 17, the best RMSE is for the neural network model and that predicted all ratings of 4, which is the middle value in a rating system of 3, 4 and 5. We rule these out as candidates for our recommender system.

- For the 3 content-based recommenders:

    - Clustering content-based recommenders have the consideration that some clusters are filtered out for having too few entries, but this can be remedied by adjusting the value of k in KMeans. There is also a scalability issue: as the number of genres on our platform increase, the features in the user profiles will increase. This can be mitigated somewhat by using PCA and while this may become an issue in the future, it only scales as O(n). On the downside, this is a more involved process than the other content-base recommenders.

    - Course similarity score models have no hyperparameters, but it does require calculating the similarities between all pairs of courses, which scales as O(n^2). However, while the scaling is of a higher order, the calculation does not need to be done unless a new course is added, whereas users will complete courses relatively more often.

    - User profile recommender systems only rely on dot products, but the profile vectors need to be updated every time a user completes a course so it scales as O(n), which may be more often than new courses are added. In addition, the user profiles may not give as tailored results as course similarities as those models may pick up on specific words in previously taken courses.

- We believe that course similarity scores are the best choice for our company.

Appendix

- All raw files for this course can be found in the raw files folder:
  - https://github.com/vishpuro/Streamlit2.git
- The streamlit app can be found
  - https://app2-8iudwu6jwezqbwdhtgsh24.streamlit.app/