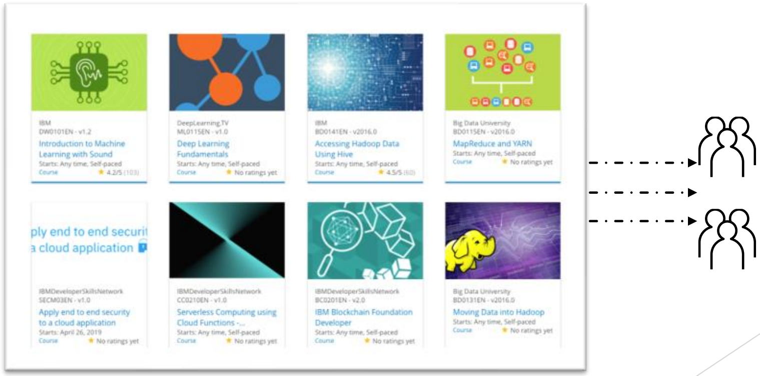
Build a Personalized Online Course Recommender System with Machine Learning

Young Rha
2023-12-18



Outline

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

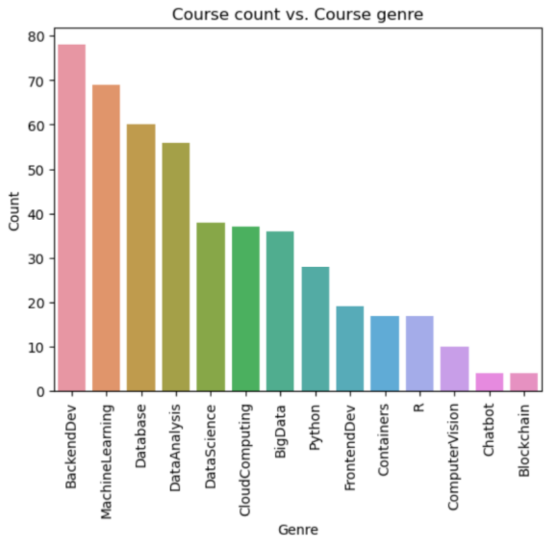
Introduction

- Goal
  - This project aims to explore recommendation system through analyzing online courses dataset. Both content based and collaborative filtering methods will be used to measure the performance between the various methods.

- Problem:
  - The online courses dataset consist of courses data and user profile data. The recommendation system aims to generate course recommendations to the user where the likelihood of accepted recommendation is optimized.
  - The recommendation system will assume several hypotheses such as:
    - A user will likely accept courses similar to courses they have taken in the past
    - Users with the similar interest are likely to accept courses taken by other users with the group

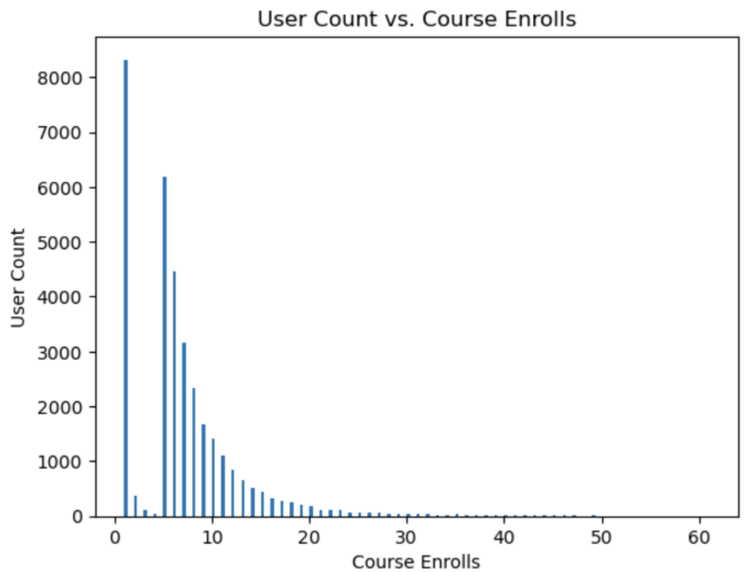
Exploratory Data Analysis

Course counts per genre



The bar chart displays count vs. genre from the course dataframe.
We can see that backend dev genre is the most occurrence within the dataset whereas chatbox and blockchain have the least occurrences.
We can also see that the count ranges from 4 – 78 with the total sum over 307 (number of courses within the dataset) as expected since there will be some overlap of genre per courses.

Course enrollment distribution



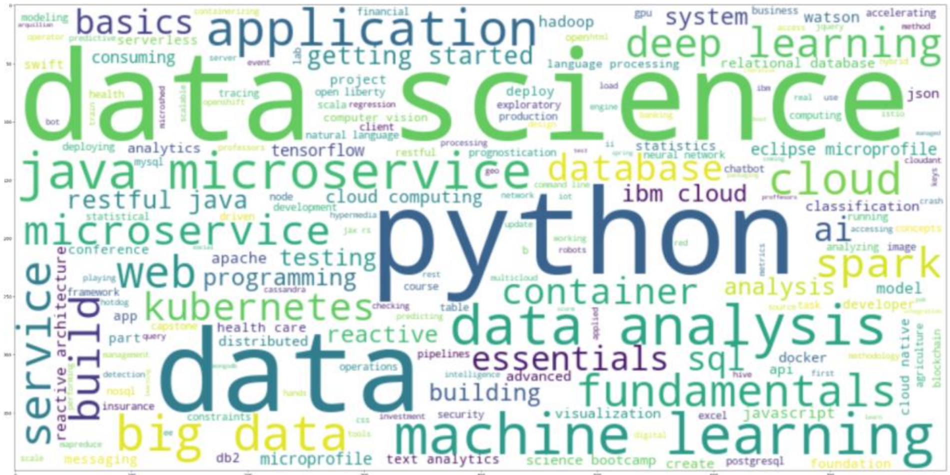
This is a histogram of user count vs. course enrolls to observe the distribution.
We can see that majority of users only enroll to a single course.
however, if they do enroll for more than one, they do seem to continue enrolling to more courses with the curve approaching a plateau toward ~50 enrolls.

20 most popular courses

	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

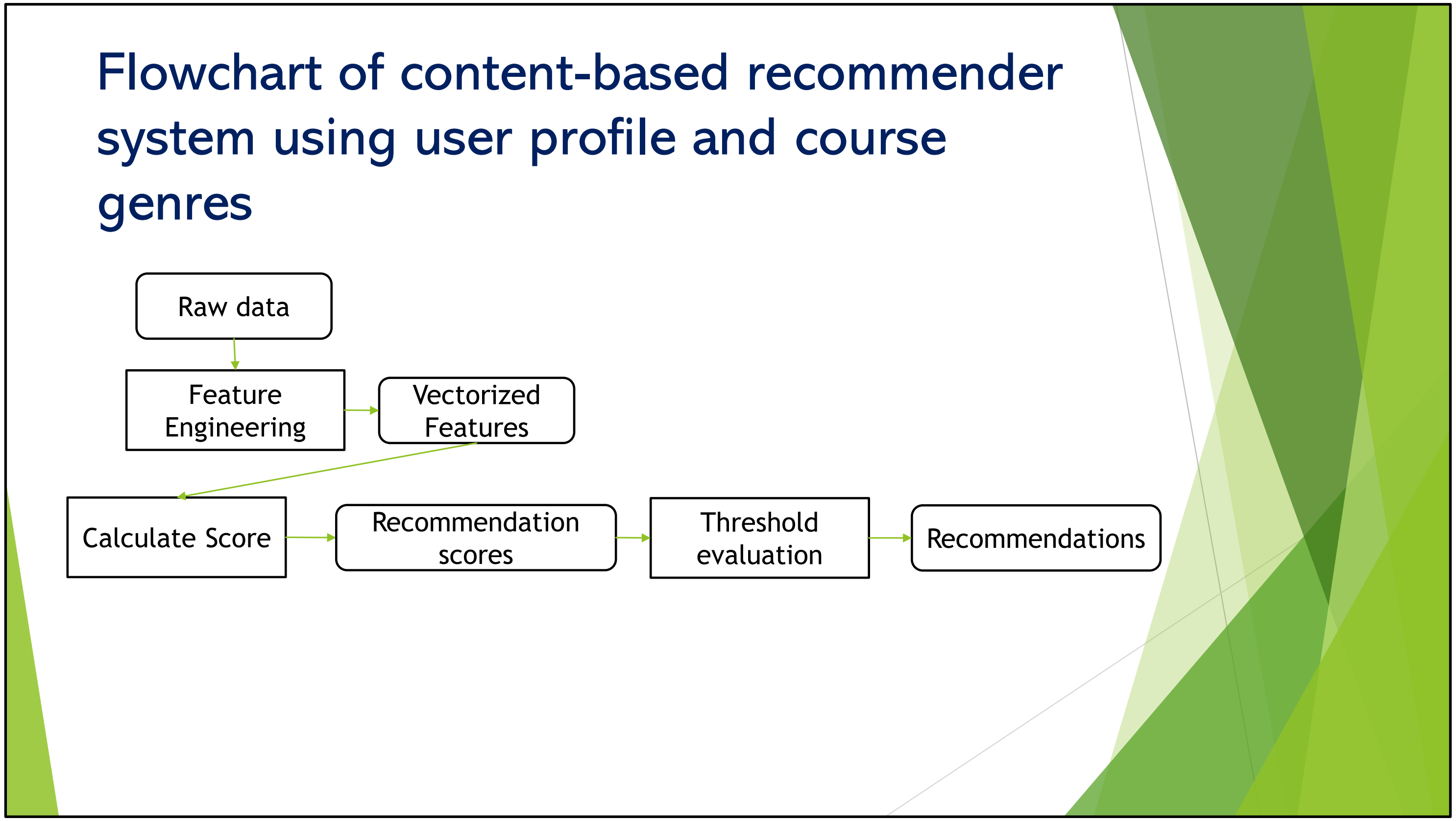
Here we have top 20 most popular courses from the dataset.
We can see that python for data science is the most popular course with 14936 enrolls followed by introduction to data science with 14477 and big data 101 13291.
The 20th popular course is data privacy fundamentals with the enrollment of 3624

Word cloud of course titles



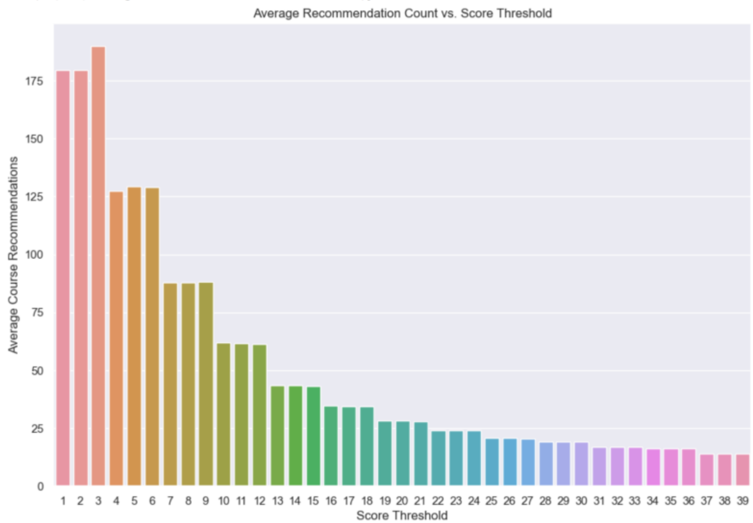
The wordcloud of the course titles show the popularity of key words in the course titles such as data science, python, machine learning, and big data.

Content-based Recommender System using Unsupervised Learning



The categorical features are extracted from raw data in form of profile and course genres.
Both profile and course genre is then turned into vectors for score calculation.
The recommendation score is then calculated via dot product of a course vector and a profile vector.
The only hyperparameter in recommender system is score threshold. If the score is higher than the threshold, the course will be recommended to the user.
The threshold value is then fine tuned to adjust the number of recommendations.

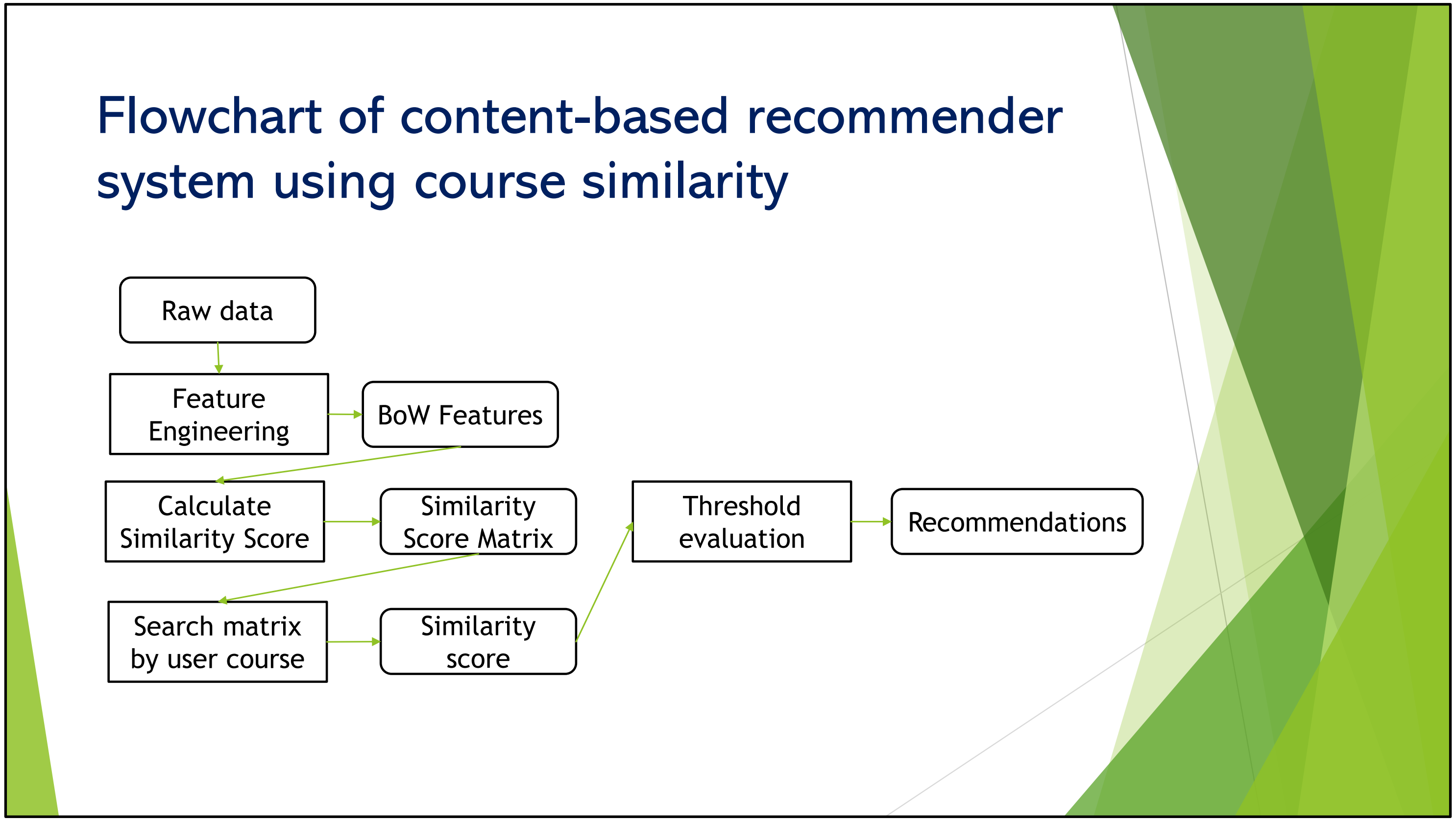
Evaluation results of user profile-based recommender system



Top 10 Courses
Threshold: 10

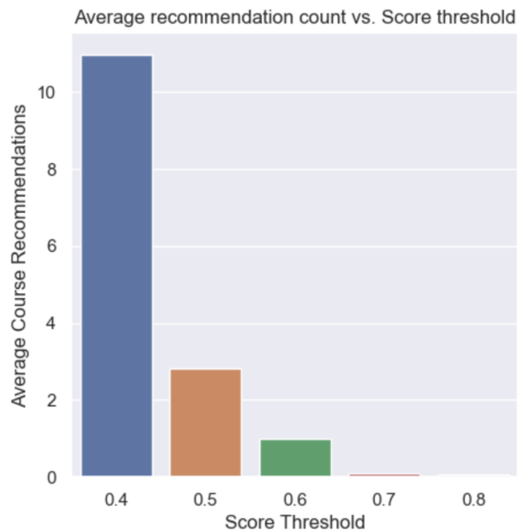
COURSE_ID	count
TA0106EN	608
GPXX0IBEN	548
excourse22	547
excourse21	547
ML0122EN	544
GPXX0TY1EN	533
excourse04	533
excourse06	533
excourse31	524
excourse73	516

Iterating through threshold score of 1 – 39
The number of recommendation goes down over increasing threshold as expected.
The top 10 was found from the recommendations using score threshold of 10



The raw data consist of courses with their title and description.
The feature engineering step takes the course titles and create bag of words features per course.
The pairwise similarity score was calculated based on bow features between all pair of courses resulting in the similarity score matrix.
The recommendations were then generated by using each user's enrolled course information as an input. Here, we are excluding the enrolled course from the list of available courses to avoid recommending the same course.
The similarity score is retrieved from the score matrix calculated in step 3 between the enrolled course and the rest of available courses. Each recommendation is required to pass a certain similarity score threshold which can be fine tuned to control the size and quality of recommendations.

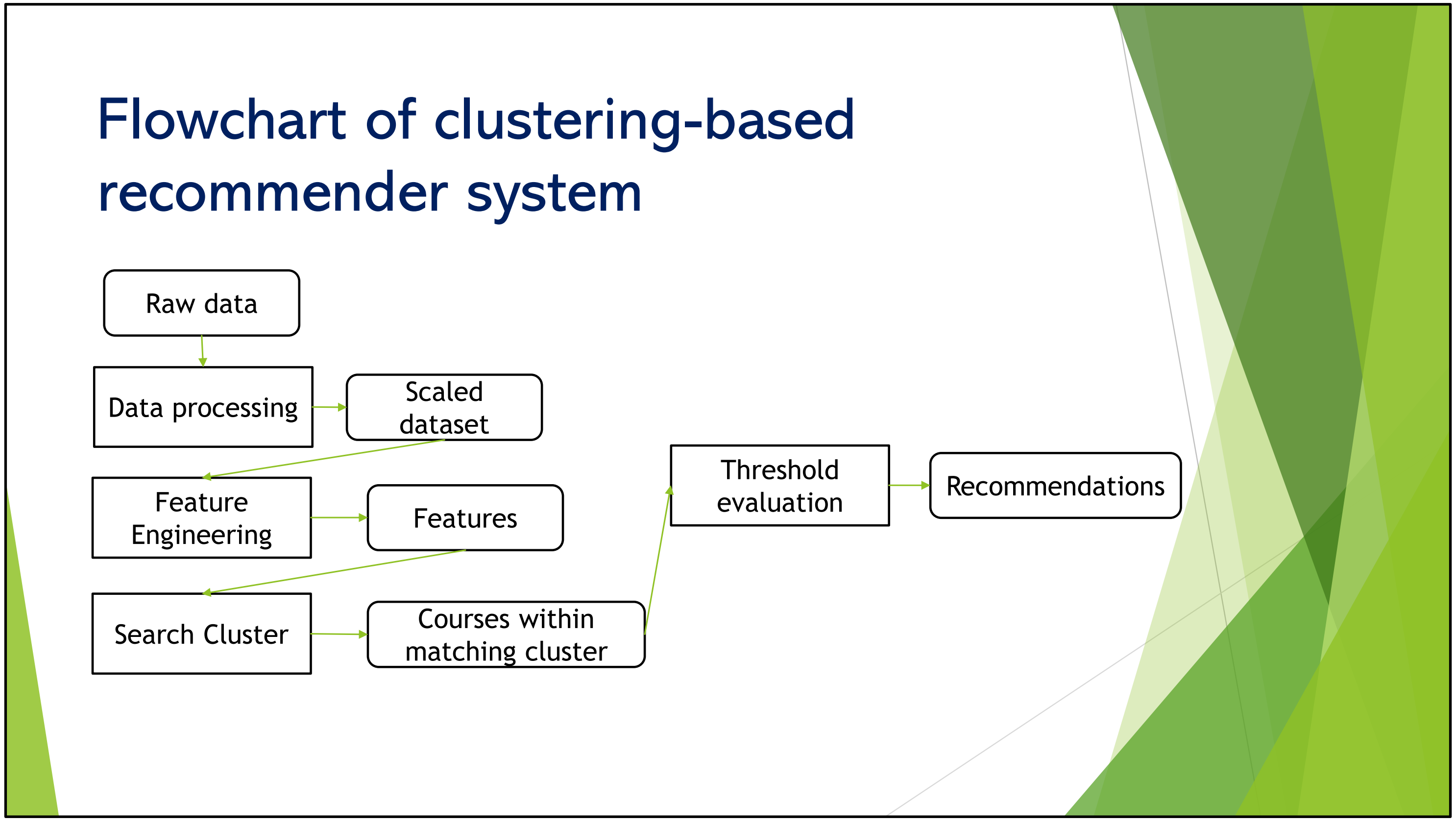
Evaluation results of course similarity based recommender system



Top 10 courses
Threshold: 0.6

COURSE_ID	count
excourse62	257
excourse22	257
WA0103EN	101
TA0105	41
DS0110EN	38
excourse47	24
excourse46	24
excourse65	23
excourse63	23
TMP0101EN	17

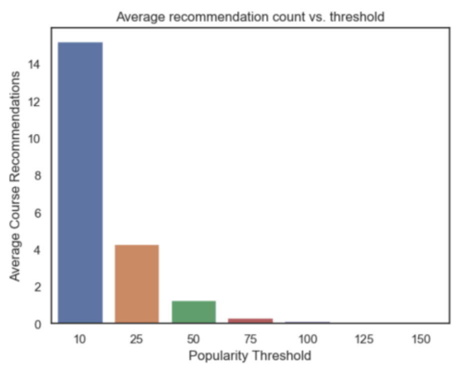
The graph displays average number of recommendations per similarity score ranging from 0.4 ~ 0.8
At 0.7 similarity score, the average number of recommendation approaches below 1 indicating that most courses have less than 0.7 similarity score given the dataset



Data processing step takes care of unbalanced dataset and uses standardscaler to normalized the skew
Feature engineering step includes PCA and clustering (ie: k-means) to reduce the dimensionality of the dataset as well as adding on cluster label as a new feature.
The features are then re-grouped based on the cluster label and user.
For each input (user), the matching cluster is searched for the list of available courses, which is then evaluated based on the threshold popularity of the course within the cluster.
If it passes the evaluation, the courses within the cluster will be recommended.

Evaluation results of clustering-based recommender system

K-Means: n_clusters = 15
PCA: n_components: 9

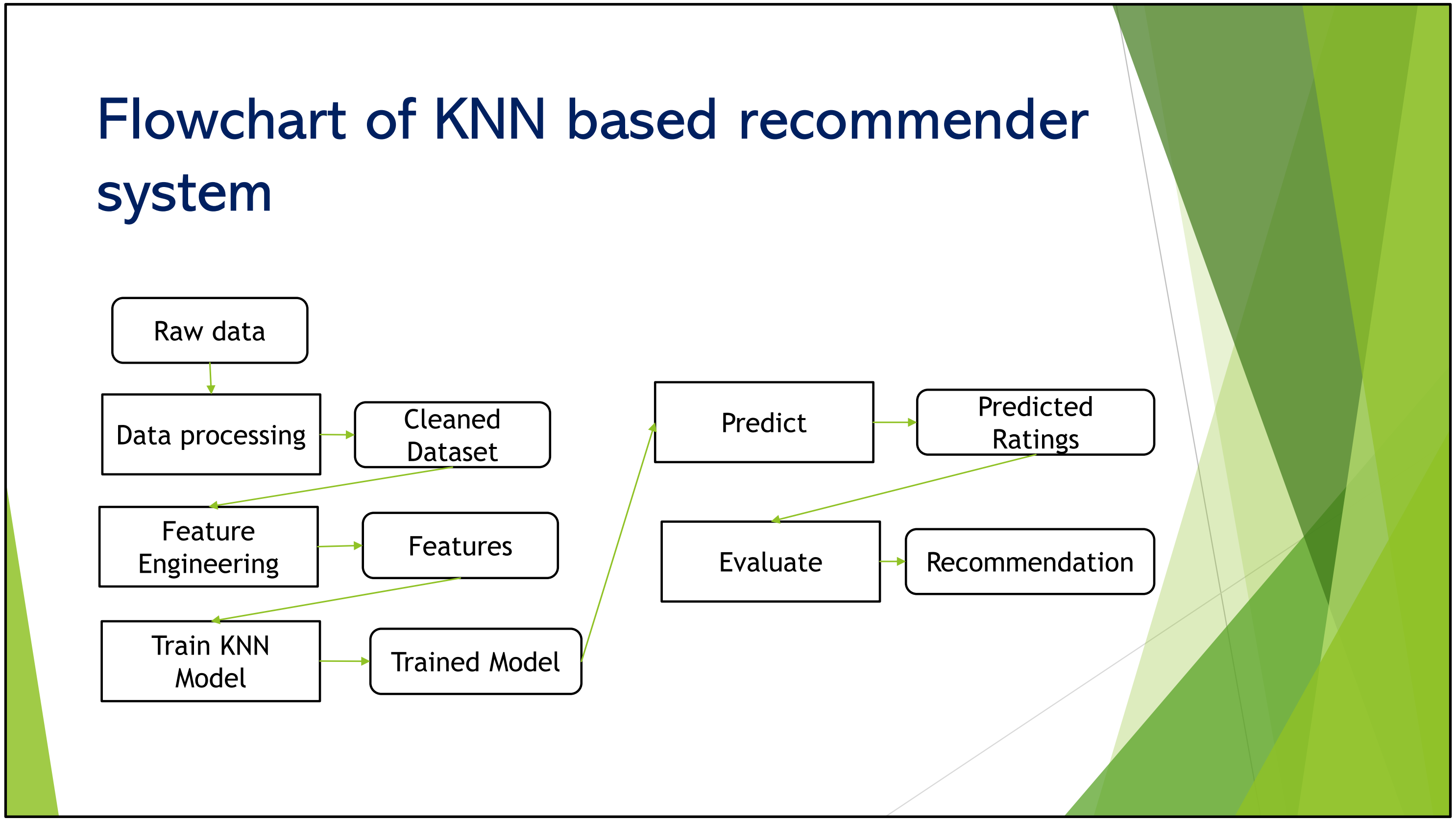


Top 10 Courses
Threshold: 50

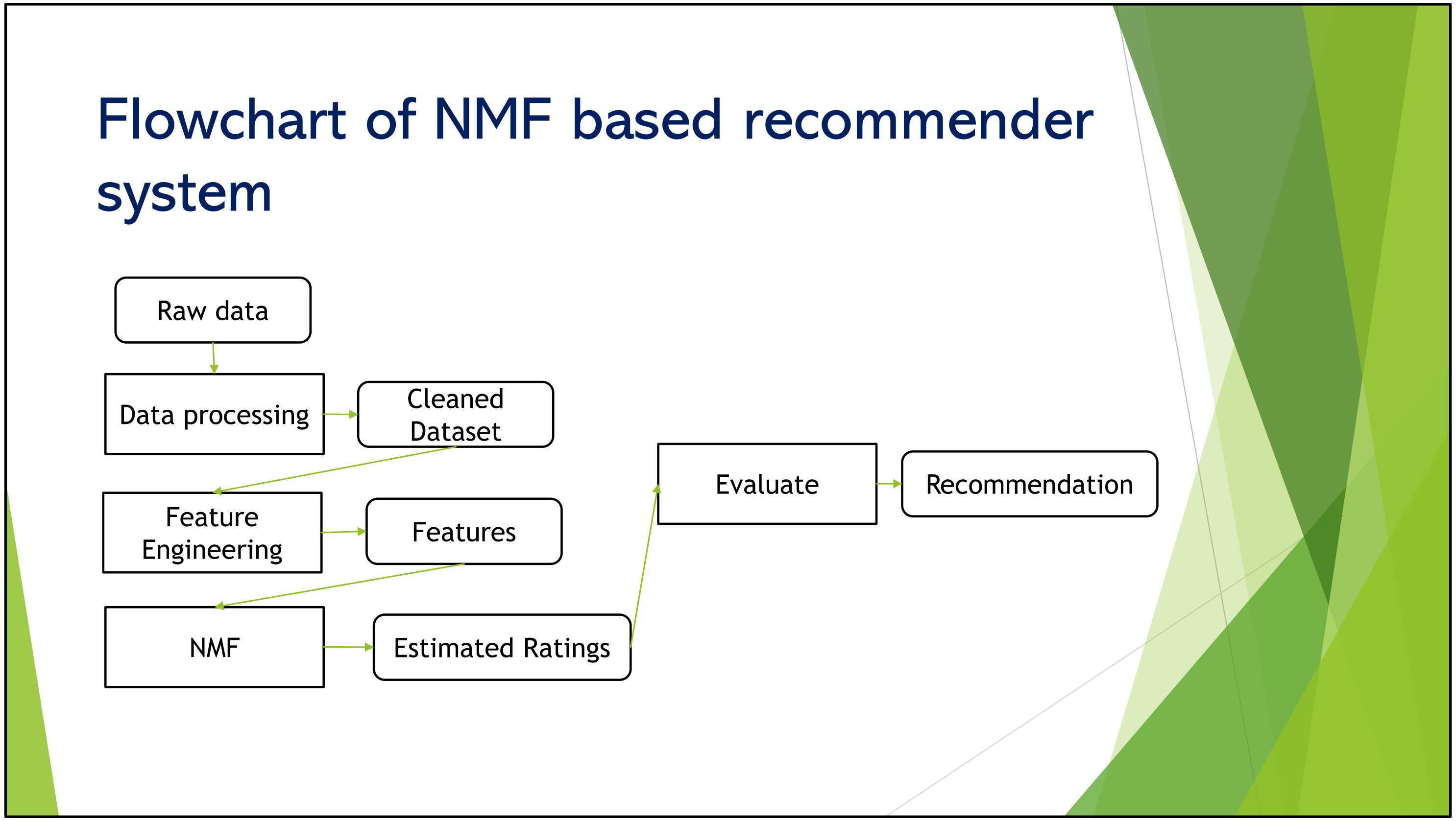
rec	count
DS0101EN	2326
BD0101EN	1958
PY0101EN	1123
DS0105EN	789
ML0115EN	762
DS0103EN	760
BD0115EN	594
DV0101EN	503
ML0101ENv3	464
DA0101EN	331

The average recommendation vs. popularity threshold graph shows that at threshold beyond ~75, the number of recommendation approaches near 0

Collaborative-filtering Recommender System using Supervised Learning



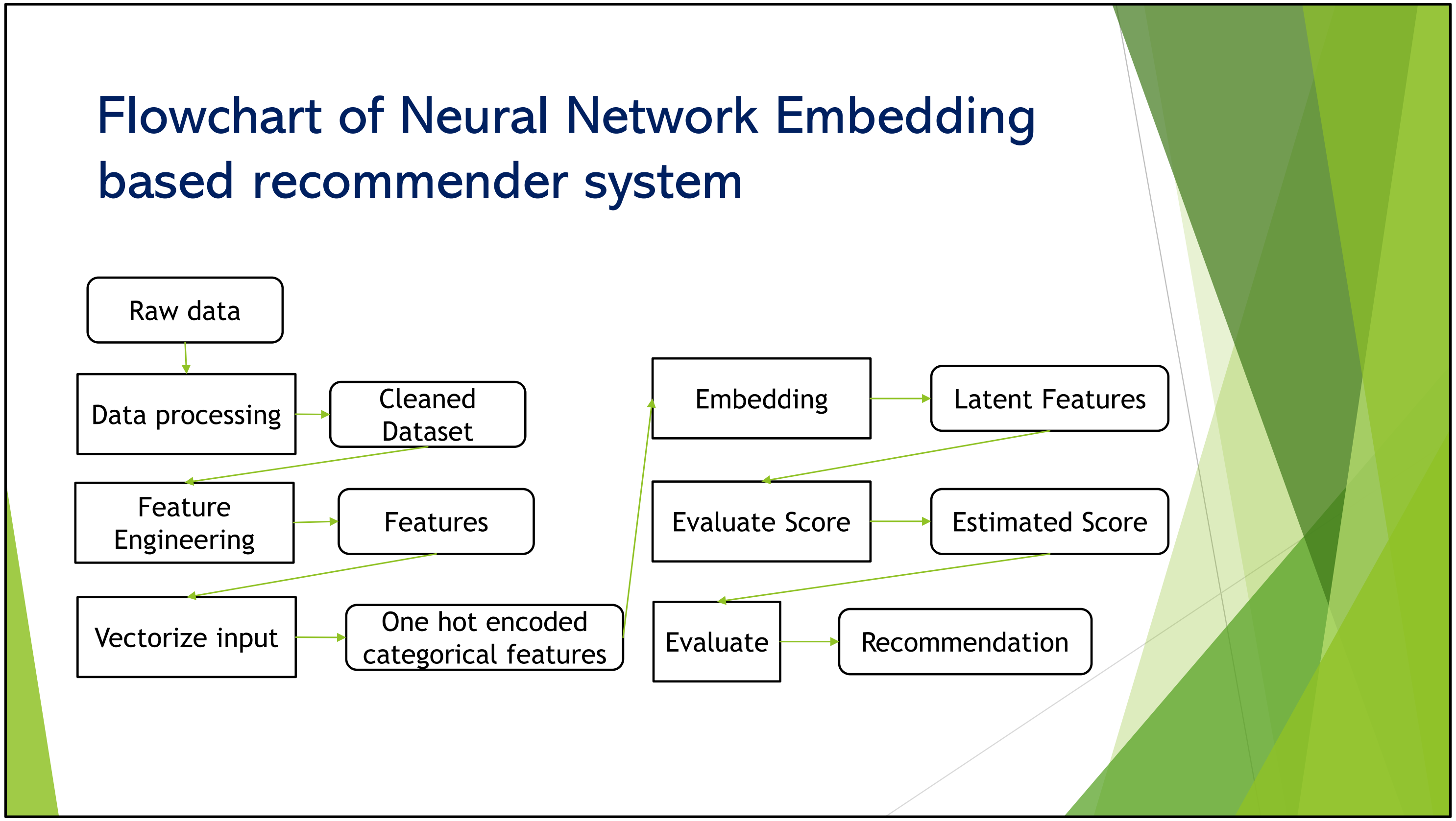
The raw data is cleaned up and engineered to retrieve relevant features.
The dataset of features are then split into train and test set
KNN model trained based on train set.
The model is then tested on the test set for validation.
The hyperparameters such as min/max number of neighbours as well as distance metrics can be further tuned until the desired validation result is achieved.
The model can be used to predict score/rating of an unknown items based on their similarity, which is then used to provide recommendation (ie: via collaborative filtering)



After clean up and engineering features, NMF will be applied to the dataset (m * n) to generate factorized matrices of non-negative features.

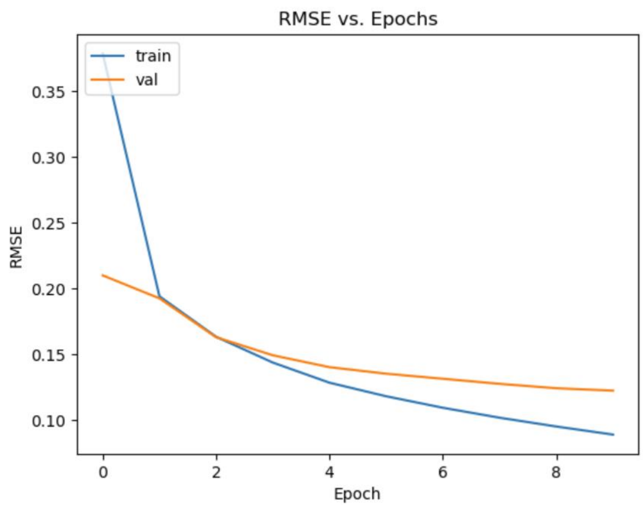
NMF can be used to estimate the original rating which can be then used to generate the estimated rating while reducing its dimensionality as well as highlighting the important features.

The estimated ratings then can be used to generate recommendations by collaborative filter.



The features are processed into one hot encoded categorical features per feature and fed into the embedding layer of neural network.
The neural network will construct a latent features from the input which will be then evaluated to predict the outcome.
The neural network will be trained to minimize the loss function on train set.
Once the model is trained, the further optimization can be done by tuning the hyperparameters.
The model then can be used to predict the outcome (ie: probability of completing the course) based on unknown input (ie: user, course)
Based on the evaluation (ie: probability threshold), the recommendation then can be generated.

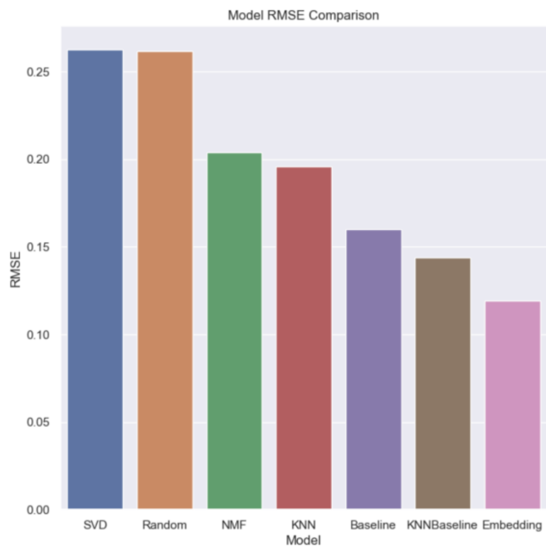
Validation result of Neural Network Embedding based recommender system



Hyperparameters
- Loss: MSE
- Optimizer: adam
- Metrics: RMSE
- Epochs: 10
- Batch size: 64
- Embedding layers: 16

The graph displays RMSE value of train and validation set over 10 epochs
The RMSE value seem slowly decrease with the epochs though not as much as the train value

Compare the performance of collaborative-filtering models



SVD: n_factors = 100, epochs =20, yields RMSE value of 0.2627
Random: yields RMSE value of 0.2615
NMF: n_factors = 15, epochs = 50 yields RMSE value of 0.2040
KNN: k = 40, yields RMSE value of 0.1958
Baseline: yields RMSE value of 0.1598
KNNBaseline: k=40, yields RMSE value of 0.1440
Embedding: 16 embedding layers, yields RMSE value of 0.1190
The result shows that SVD performed the worst with RMSE value of 0.2647 and embedding with the best RMSE value of 0.1190

Conclusions

- Content-based recommendation methods
  - All 3 methods seem to return varying courses as their top recommendations. Suggesting that they capture different area of interest for the recommendation.
  - The results seem to heavily depend on the type of methods as well as the hyperparameters.
  - Choosing hyperparameters requires good insight into the dataset as well as the objective of the recommendation system.
  - Perhaps combining various content based recommendation methods may yield a better result as it may capture a larger scope of the user's interest. It may also be possible to capture the hit rate of each method and build a supervised learning model such as neural network to further improve the results.
- Collaborative filtering
  - Given the strong performance of the baseline method, the dataset doesn't seem to suffer from the overfit. It may be the reason why knn and nmf based methods performed worse than the baseline method as well. Perhaps the hyperparameters for knn and nmf require more tuning.
  - Embedding based method seems to have decreasing RMSE value at 10 epochs. The results may be further improved with fine tuning the hyperparameters.
  - While all methods seem to yield relatively low RMSE value, the embedding method seems to be the best method for the recommendation system.

Appendix

- Content-based clustering
  - Kmeans inertia vs. clusters
  - PCA acc. variance vs. n components