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:
  - The IBM Course Recommendations dataset is a collection of data related to courses and student interactions with them. It includes information about courses, such as the course name, description, and rating, as well as data on student interactions with the courses. The dataset may be used to build a recommendation system that suggests courses to students based on various parameters such as course similarity.
- Problem states and hypotheses:
  - For this project the dataset had both supervised and unsupervised models. Based on the EDA plots which showed that the data was not normal the hypothesis is that the supervised models will work better and that the best model should be a neural network due when it comes to its accuracy as it is designed for far more complex tasks.

Key terms

- Silhouette score:

  Evaluates the quality of recommendations generated by the recommendation system with values ranging between -1 and 1 to show how well the courses fit together and how different they are from other items in other clusters.

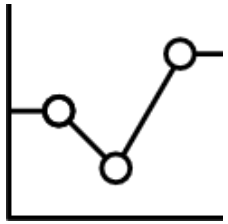
- F1_score:

  Combines precision and recall and indicates whether a system has a balance of high precision and high recall i.e. it shows how relevant the courses are to users.

- Similarity matrix:

  Matrix which represents the similarity between different items in a database which allows a recommendation system to identify items that are similar to one another (this is given by IBM)

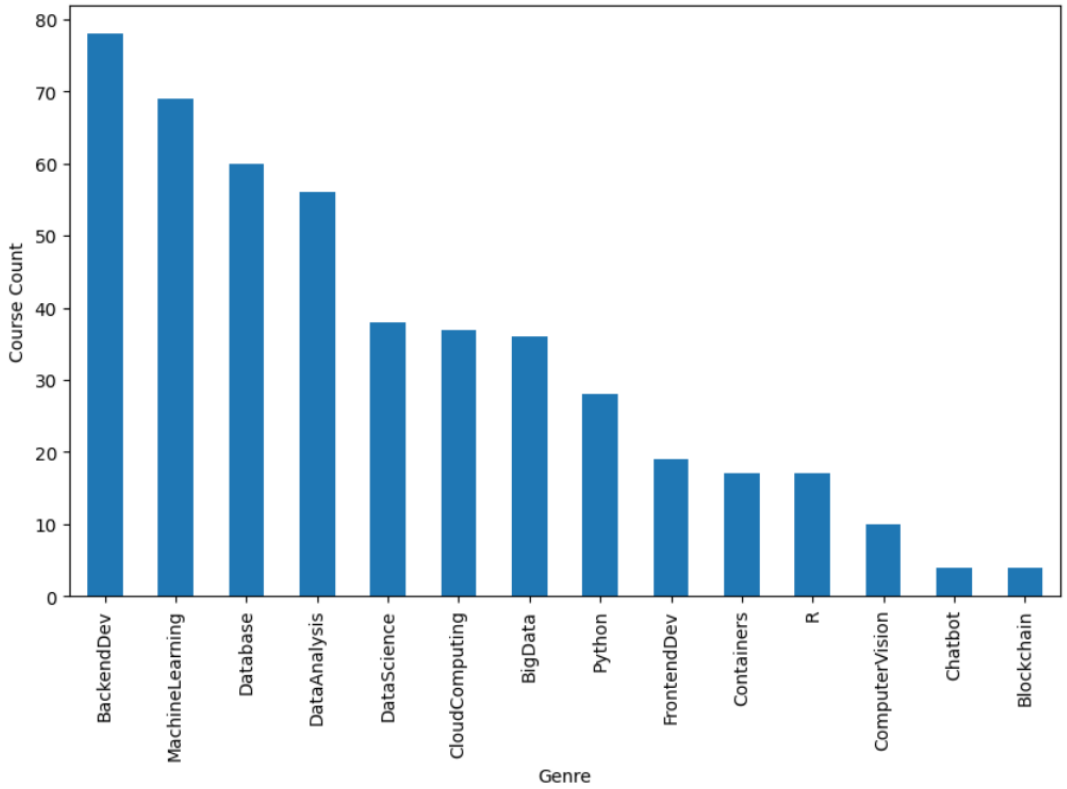
Exploratory Data Analysis



Summary of EDA Process

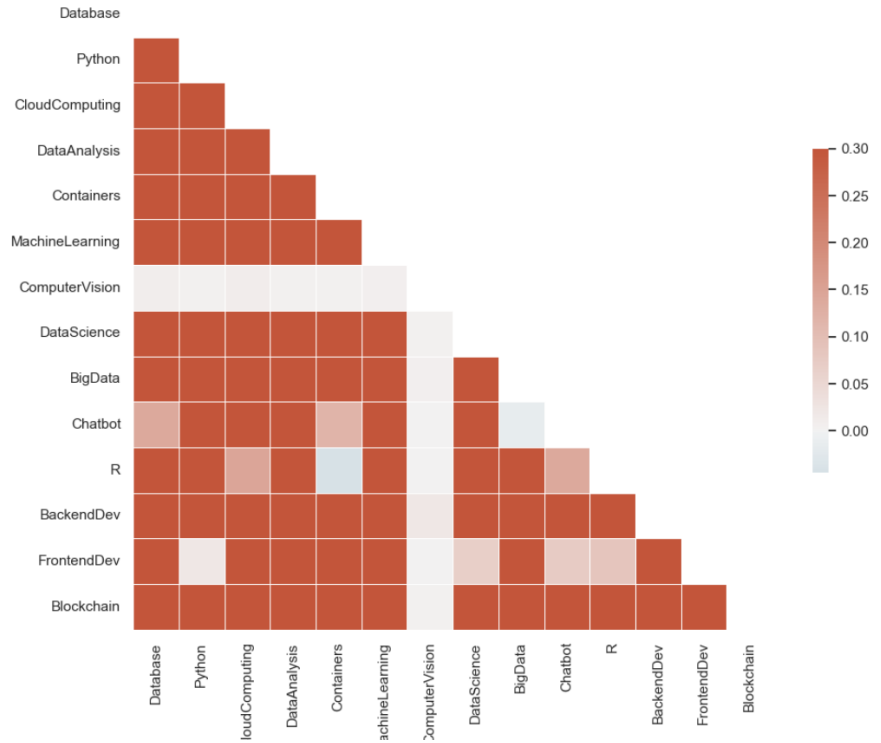
- imports a number of libraries, including STOPWORDS, Pandas, NumPy, Matplotlib, Seaborn, and Pandas.
- 8 CSV files are read in, and they are then stored in Pandas DataFrames.
- prints some facts and data regarding each DataFrame.
- shows the number of courses in each category as a bar graph.
- combines the user ratings data and displays a histogram of the average number of user ratings.
- combines the course rating information and creates a bar graph of the 20 most well-liked courses.
- creates a word cloud using the phrases that appear the most in the course descriptions.
- calculates and prints the scores of similarity between each course and the graded courses by course rating.

Course counts per genre



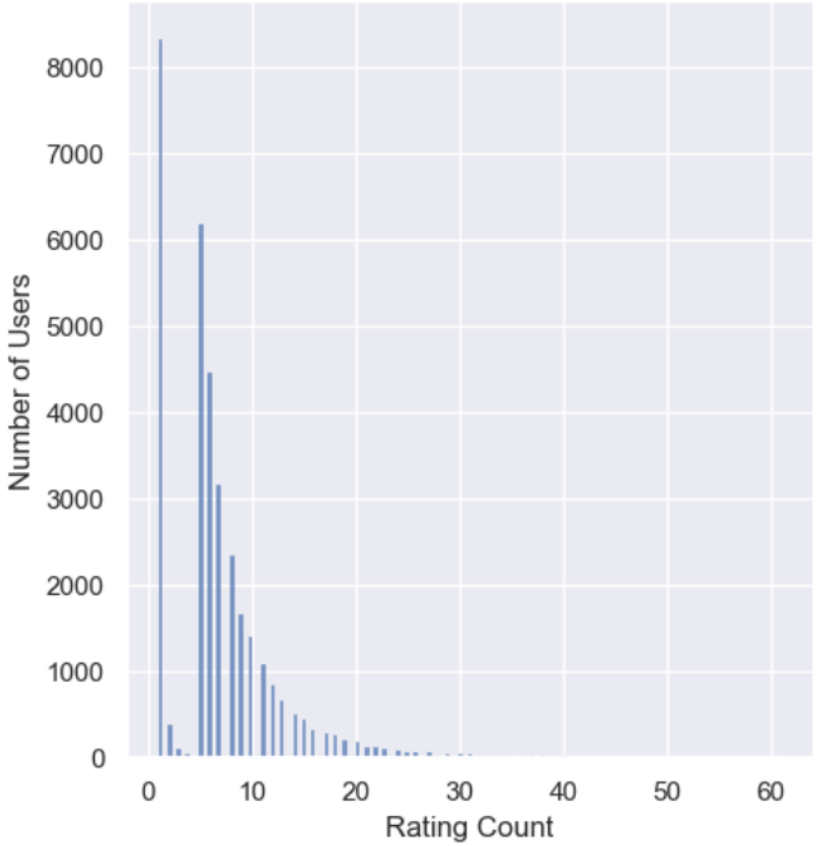
- The bar graph shows a visual representation of the balance of genres in the dataset.
- The chart shows that the data heavily favours certain genres such as Backend Development and Machine Learning but underrepresents other genres such as Chabot and block chain
- Based on the Bar chart it can be assumed that courses based on Backend Development are more likely to be recommended which provides a general expectation on what the recommender system will be suggesting

Heat map of genres and their correlation



- The heat map shows how closely linked the courses are with their content which will be important for the second unsupervised model,
- The data shows that the genre: computer vision has low correlation with other variables thus it can be ignored for the second model to improve model speed.
- The data also shows that most other variables do have fairly similar levels of correlation which will make it harder for the content based recommender to suggest appropriate results

Course enrollment distribution



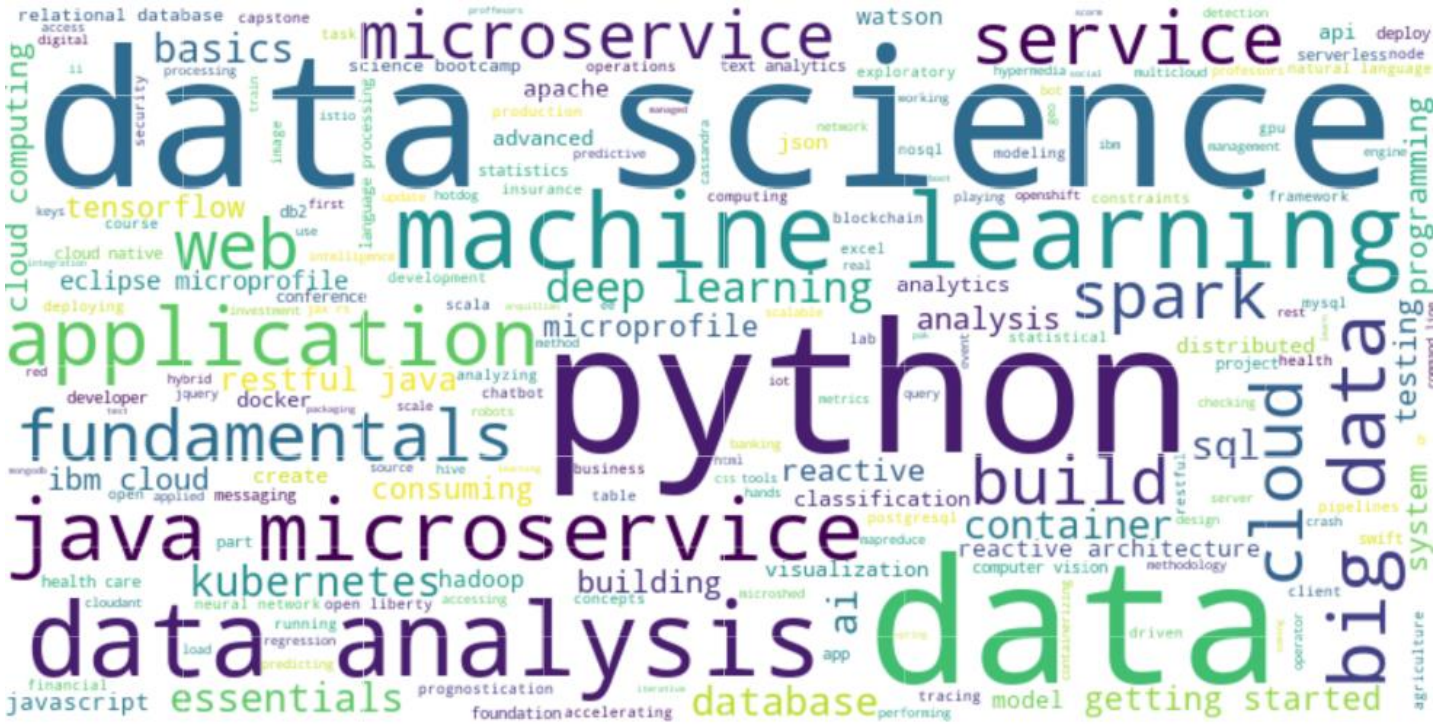
- A histogram shows the graphical representation of the distribution of a dataset.
- The shape of the histogram shows that the data is left-skewed which can impact a recommendation system in several ways such as:
  - Poor predictions of users who generally rate courses highly as the data skews towards lower ratings
  - Poor performance as it is harder for the recommender to identify models with higher rating counts

20 most popular courses

	TITLE	enrollments
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

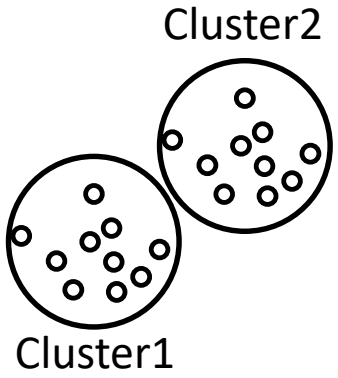
- Listing the top 20 most popular courses is found by checking the dataframe for the highest number of enrolments.
- It shows what courses are most likely to be recommended to other users e.g. based on the table it is expected that python for data science will be the most recommended course
- This can be used as an indicator for how effective a model is.

Word cloud of course titles

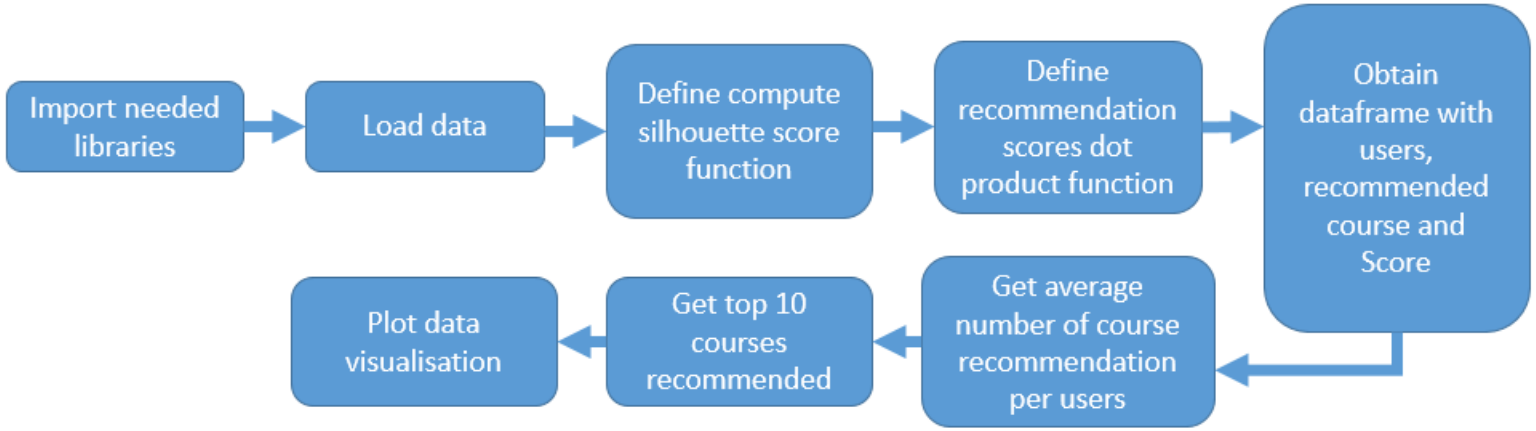


- The word cloud is a graphic representation of the importance or frequency of terms used in a text.
- The most popular or significant words are usually displayed in the larger font sizes in a list of terms with varying font sizes.
- From the graphic, some of the most common words which appear are python, data science and machine learning thus we can expect the models to predict courses based around these topics

Content-based Recommender System using Unsupervised Learning



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



- The code reads the csv(s) required and defines a function which will compute the silhouette score.
- It then defined a function which generates recommendations using the user's profile vector and the vectors for unknown courses to generate recommendation scores for each test user in test user ids (courses the user has not enrolled in).
- The functions are then run and the top 10 recommended courses is generated as well as the average number of course recommendations per users and the performance metrics of the model itself

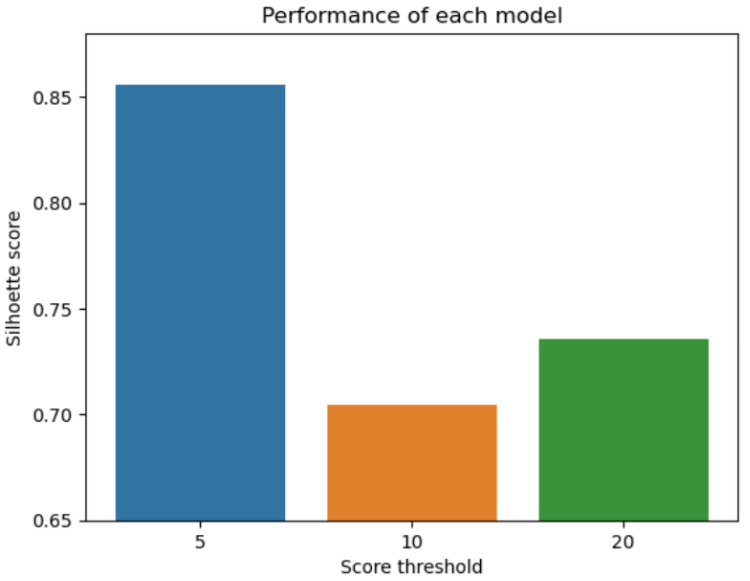
Evaluation results of user profile-based recommender system

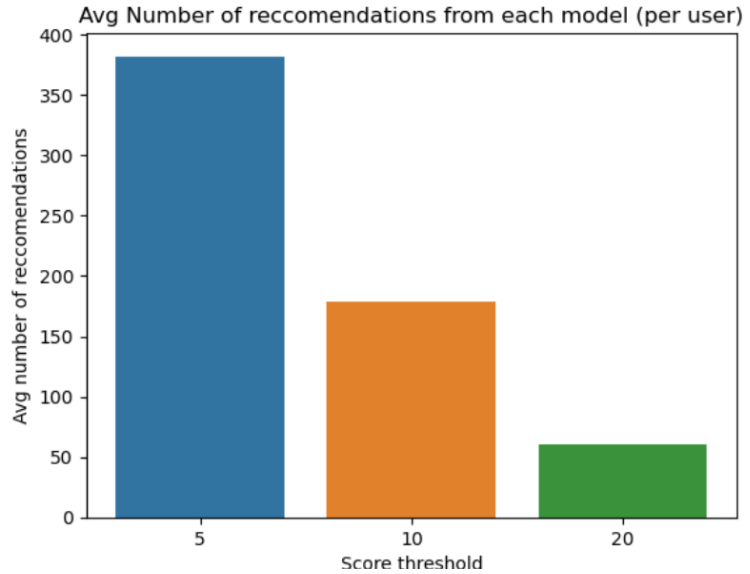
The hyper parameters which were used for this model was the score threshold:

- Model 1 score threshold = 5
- Model 2 score threshold = 10
- Model 3 score threshold = 15

The accuracy of each model is noted by the silhouette score which is clearly plotted on the graph to the right. The results follow similar trends to the histogram which saw a left leaning skew with dips occurring around a rating of 10.

The average number of recommendations for each model (per user) is plotted on the bottom right graph with the top 10 recommendations for the best model being noted in the dataframe being noted below:

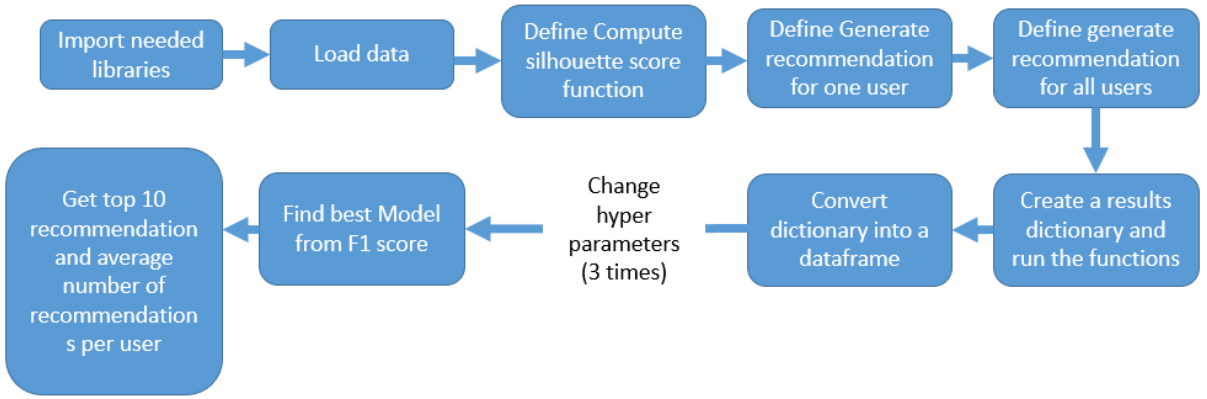




The courses being recommended are similar to what was expected from the word plot and most popular course list with a lot of courses being data science courses

	COURSE_ID	number_of_reccomendations	TITLE
0	TA0106EN	783	text analytics at scale
1	GPXX0TY1EN	780	performing database operations in the cloudant...
2	GPXX0IBEN	756	data science in insurance basic statistical a...
3	excourse06	755	sql for data science capstone project
4	excourse04	755	sql for data science
5	excourse21	741	applied machine learning in python
6	excourse22	741	introduction to data science in python
7	ML0122EN	730	accelerating deep learning with gpu
8	excourse12	729	python scripting files inheritance and data...
9	excourse31	713	cloud computing applications part 2 big data...

Flowchart of content-based recommender system using course similarity

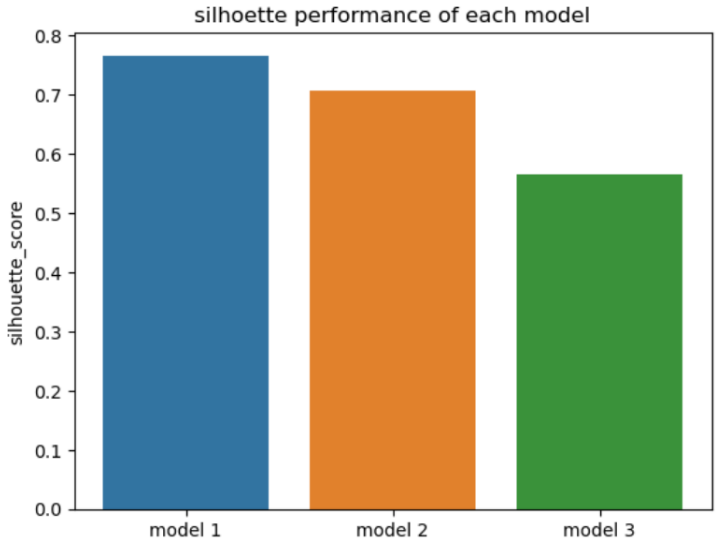


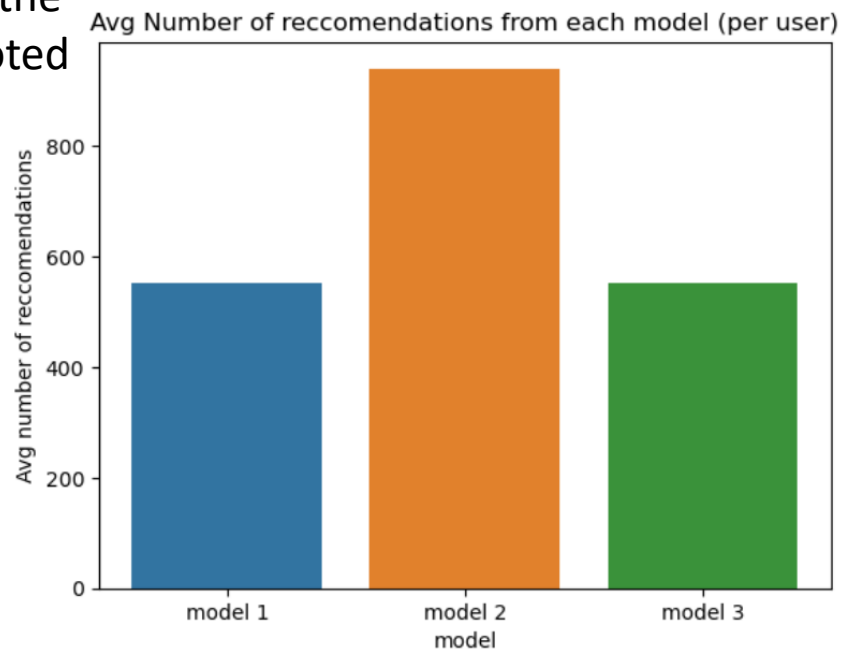
- The code reads the csv(s) required and defines a function which will compute the silhouette score.
- It then defined a function which generates recommendations using the course content for unknown courses to generate recommendation scores for each test user in test user ids (courses the user has not enrolled in).
- The functions are then run and the top 10 recommended courses is generated as well as the average number of course recommendations per users and the performance metrics of the model itself

Evaluation results of course similarity based recommender system

The hyper parameters which were used for this model was the sim_matrix (one was premade by IBM and one was generated) as well as using a generated bag of words (as opposed to the one given by IBM:

- Model 1 : original_sim_matrix and bag_of_words_df
- Model 2 : new_sim_matrix and original_bag_of_words_df
- Model 3 : new_sim_matrix and new_bag_of_words_df
- The accuracy of each model is noted by the silhouette score which is clearly plotted on the graph to the right.
- The results showed that the bag of words and similarity matrix were optimized by IBM and parameters of the method to generate the Sim matrix and bag of words matrix need further tweaking to get the optimal result.
- The average number of recommendations for each model (per user) is plotted on the bottom right graph with the top 10 recommendations for the best model being noted in the dataframe being noted below:

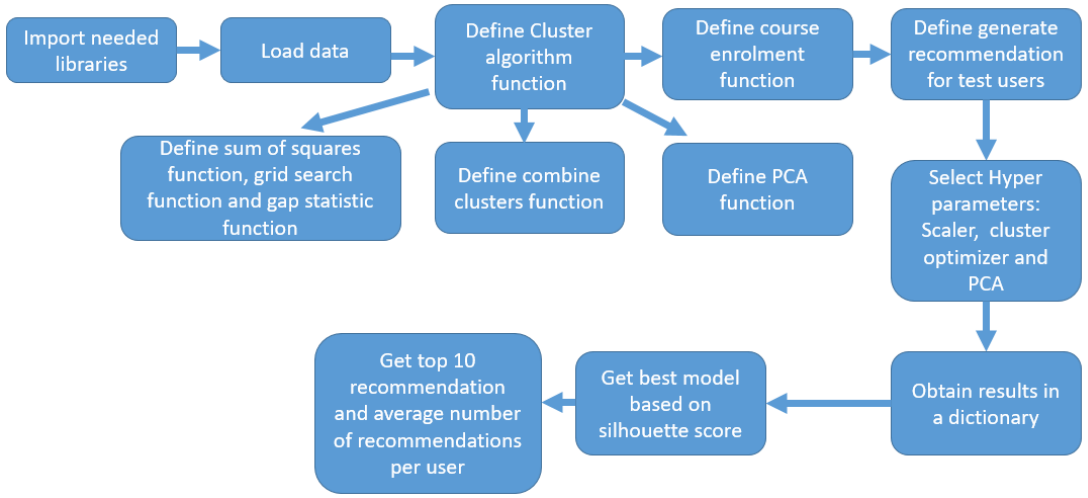




	TITLE	Frequency of reccomendation
0	introduction to data science in python	579
1	introduction to data science in python	579
2	data science with open data	562
3	a crash course in data science	555
4	data science fundamentals for data analysts	555
5	foundations for big data analysis with sql	551
6	big data modeling and management systems	550
7	introduction to big data	539
8	fundamentals of big data	539
9	sql access for hadoop	506

The courses being recommended are similar to what was expected from the word plot and most popular course list with a lot of courses being data science courses like the previous model

Flowchart of clustering-based recommender system

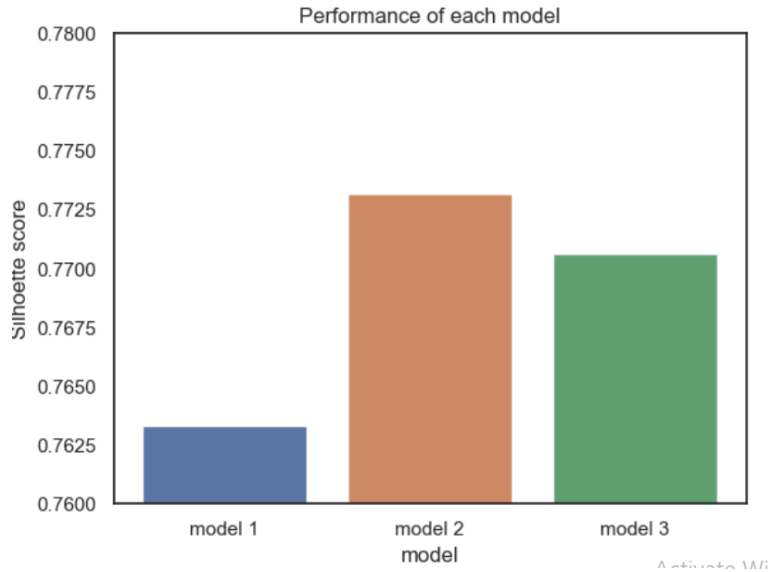


- First the relevant CSVs are loaded and the define cluster algorithm function is defined which returns the silhouette score of the model and a dataframe with the clusters.
- This function takes scalers, optimizers and threshold values and returns what has been asked of it e.g. the sum of squares optimizer will iterate through a range of values for KNN clustering to get the best KNN value using the KNN formula, PCA is also available to be used.
- The course enrolment function is then defined which is designed to return a labelled dataframe of the test users and clusters.
- This is followed with the recommend unseen model which recommends the unseen courses to each user.
- These variables are then initialized and run with the results being appended in a dictionary and the best model is found using the silhouette score as well as the top 10 recommendations and average number of recommendations per user for the best model

Evaluation results of clustering-based recommender system

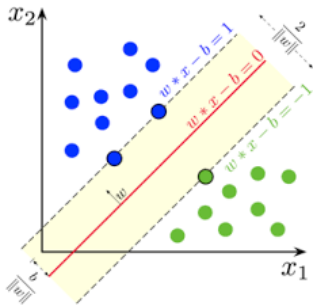
The hyper parameters which were used for this model was the cluster optimizer:

- Model 1 : gridsearchCV
- Model 2 : lowest sum of squares
- Model 3 : gap statistic
- The accuracy of each model is noted by the silhouette score which is clearly plotted on the graph to the right.
- The results showed that the lowest sum of squares was the best model and outperformed the gridsearchCV and gap statistic despite gridsearchCV being designed to find the optimal result although hyper parameters in the gridsearchCV model may have caused this
- The average number of recommendations for each model (per user) is noted in a pandas dataframe on the bottom right
- **The average number of recommendations for the best model(model_2) per user was: 41**
- The courses being recommended are similar to what was expected from the word plot and most popular course list with a lot of courses being big data and being based on backend development such as Hadoop and spark
- The number of recommendations is significantly lower than the other models which had more than 5 times the number of recommendations

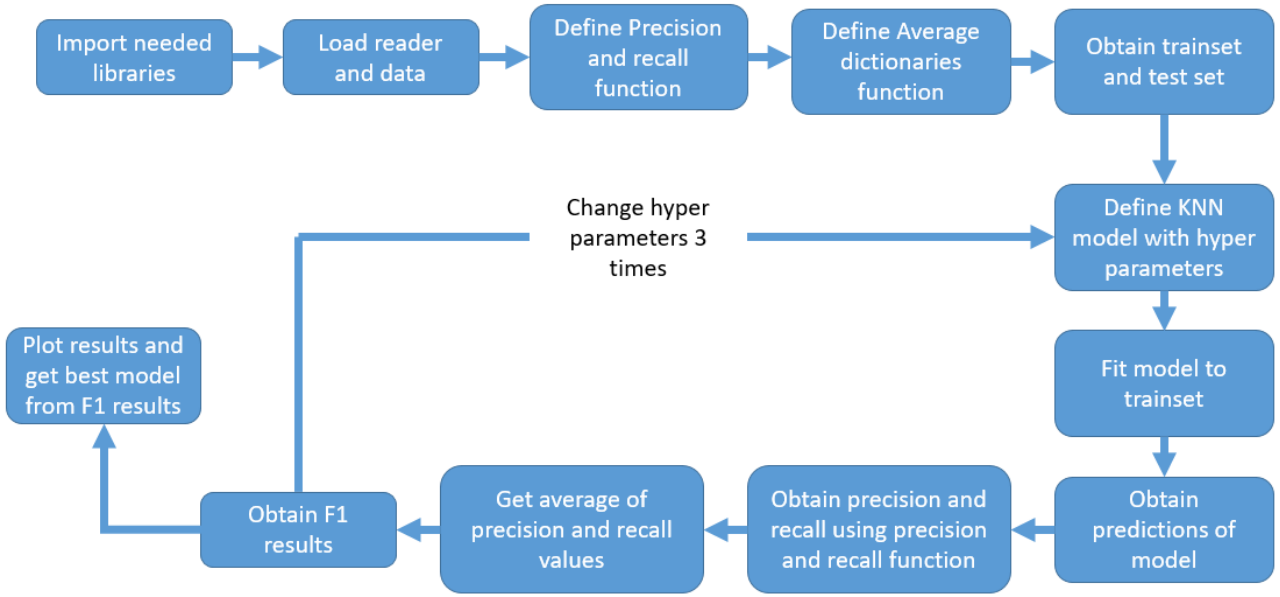


	enrollments	COURSE_ID	TITLE
0	100	BD0111EN	hadoop 101
1	94	BD0101EN	big data 101
2	85	BD0211EN	spark fundamentals i
3	57	BD0115EN	mapreduce and yarn
4	50	BD0141EN	accessing hadoop data using hive
5	46	BD0131EN	moving data into hadoop
6	32	DS0101EN	introduction to data science
7	32	PY0101EN	python for data science
8	27	SC0101EN	scala 101
9	21	BD0212EN	spark fundamentals ii

Collaborative-filtering Recommender System using Supervised Learning

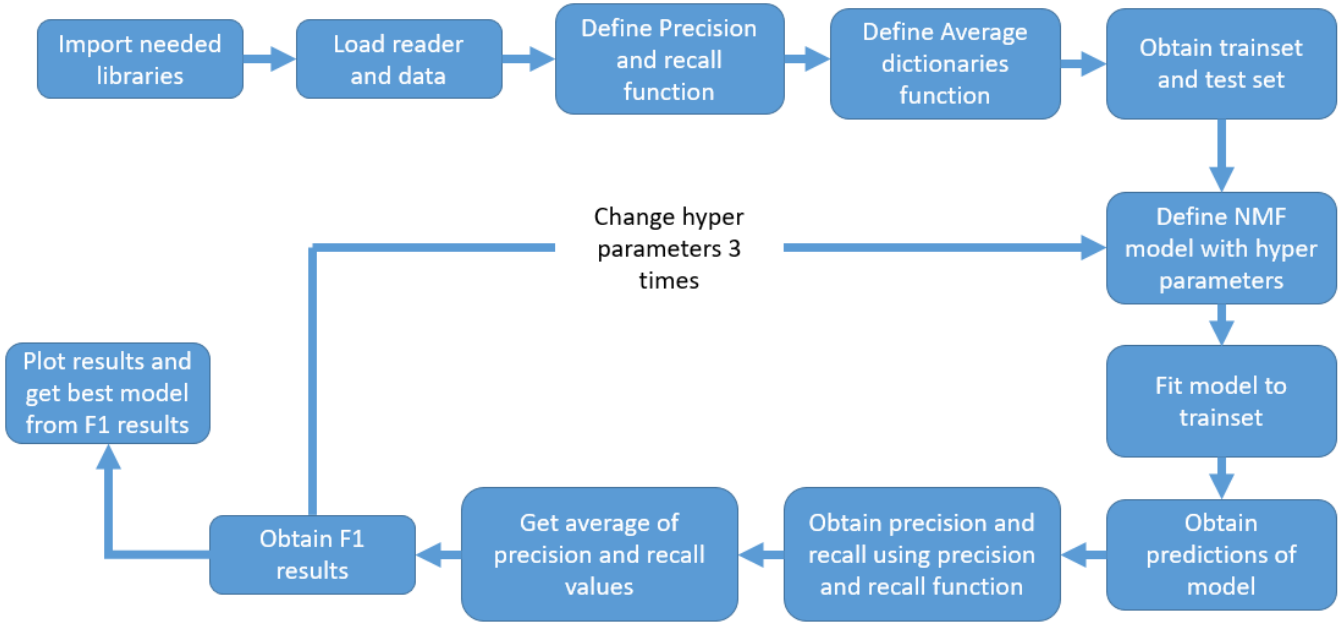


Flowchart of KNN based recommender system



- This code is loading in a CSV file called ratings.csv and using the surprise library to perform collaborative filtering on the ratings data. The data is prepared and split into a training set and a test set using the train_test_split function.
- The code then defines a function called precision_recall_at_k that takes in a list of predictions, the number of recommendations to make (k), and a threshold rating value. The function calculates precision and recall at k for each user and stores the results in dictionaries.
- Next, the code defines a function called average_dicts that calculates the average value for a dictionary.
- Three KNNBasic models are then trained and tested on the data, each with different values for the number of nearest neighbors (k) and similarity measure (msd, cosine). The f1 score is calculated for each model using the precision_recall_at_k function and the average_dicts function. Finally, the f1 scores are printed.

Flowchart of NMF based recommender system

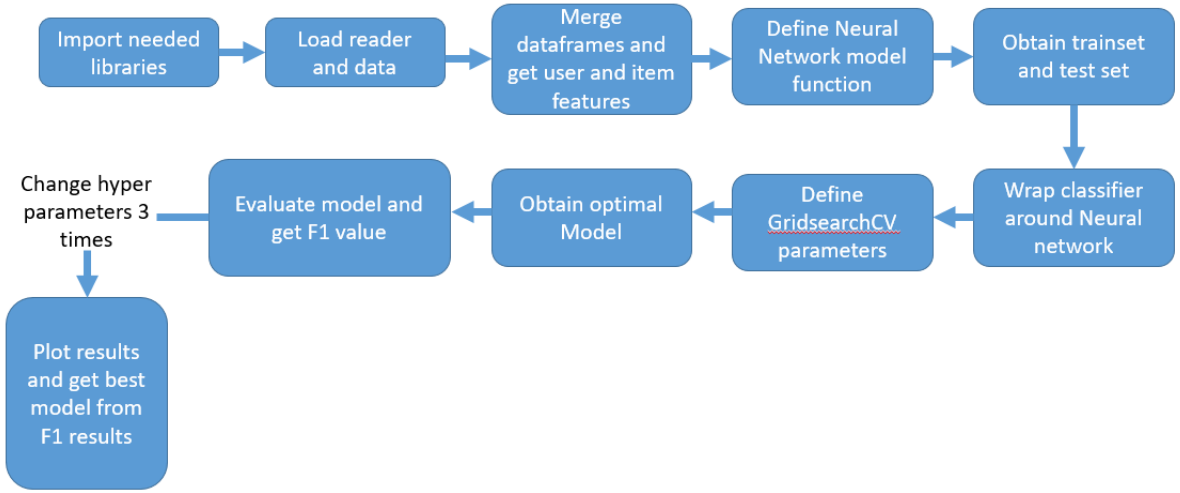


This code is implementing a collaborative filtering recommendation system using the non-negative matrix factorization (NMF) algorithm from the Surprise library. It reads in a csv file of course ratings called "course_ratings.csv" and defines a custom function called "precision_recall_at_k" that calculates the precision and recall of the recommendations at k, given a list of predictions, a value of k, and a threshold rating.

It then splits the dataset into a training set and a test set, and defines three NMF models with increasing number of factors and epochs. It trains each model on the training data and then generates recommendations for the test set.

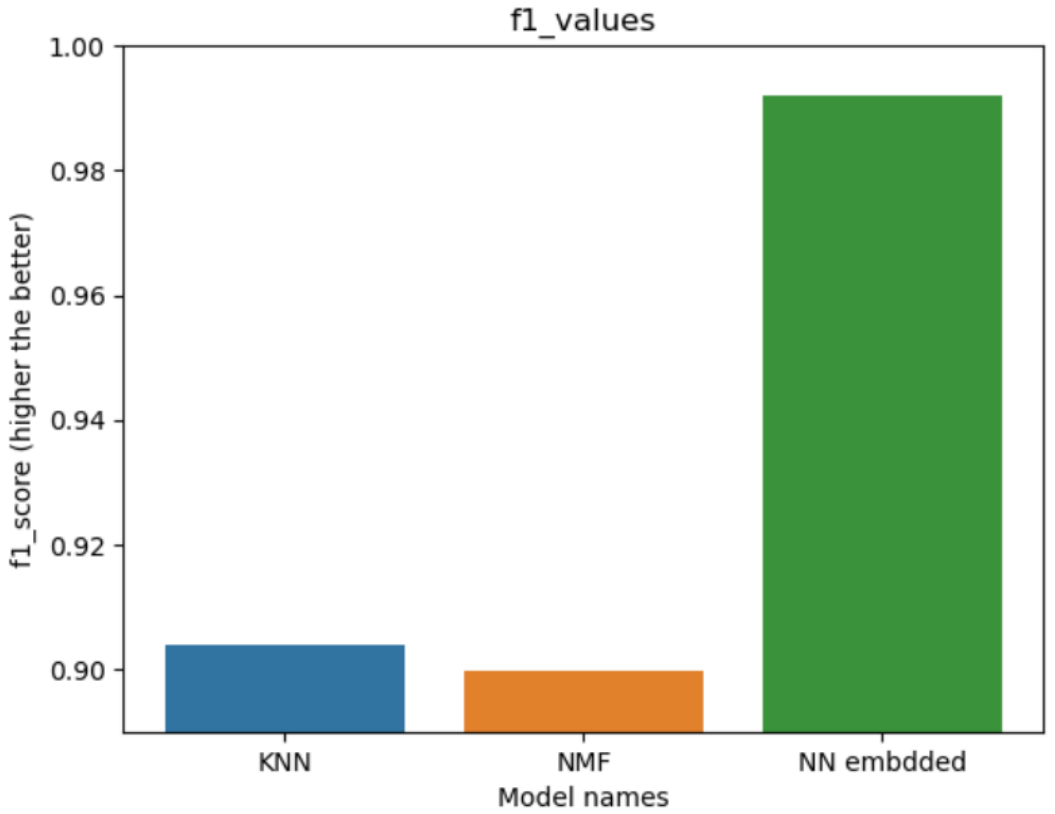
For each model, it calculates the precision, recall, and f1 score of the recommendations using the "precision_recall_at_k" function and the average_dicts function, which calculates the average value of the dictionary. Finally, it prints out the f1 scores for each model.

Flowchart of Neural Network Embedding based recommender system



- This code loads three datasets - a ratings dataset, a user embeddings dataset, and a course embeddings dataset.
- It then merges the datasets and extracts the user embeddings, course embeddings, and ratings from the merged dataset.
- It then creates an interaction dataset by adding the user embeddings and course embeddings element-wise and appends the ratings column to this dataset.
- It then encodes the ratings column in the interaction dataset using a label encoder. It then defines a function to build a model with a specified number of layers and units and a specified input shape.
- It then splits the interaction dataset into a training set and a test set and then creates a KerasClassifier object that wraps the model and defines a parameter grid for the number of layers and units.
- The algorithm then creates a GridSearchCV object to search over the parameter grid and fits the model to the training data. It then prints the best parameters and score from the grid search and gets the best model from the grid search. It then evaluates the best model on the test data and prints the F1 score.

Compare the performance of collaborative-filtering models



- The f1_values for the supervised models are graphically represented above, it should be noted that the axis has limits 0.88 and 1.0 as all three optimal models performed very well with scores of 0.904, .900 and 0.992 respectively.
- As expected the neural network ran the best as Neural networks are able to learn and model complex patterns in large amounts of data, the amount of data that was used was relatively small compared to what a Neural Network can handle
- Further hyper parameters could be tweaked to get better performance from KNN and NMF although it should be noted that these models ran much quicker than the NN model thus they may be considered over the NN model since they also had very high f1 scores.

Conclusions

- In conclusion the models performed similarly when contrasted with information obtained in the EDA: courses related to backend development and data science frequently appeared as the most recommended courses
- Some models such as the clustering based recommender system performed poorly and other methods such as using a different clustering library may help solve this issue such as the Sklearn module
- The Best model for the supervised model was the Neural network which was what was expected in the hypothesis. Furthermore, the supervised models had much higher accuracy values than the unsupervised models although it should be noted that different metrics were used to assess them
- For future work, more tuning on the hyper parameters for the unsupervised model would be needed such as altering the optimizers and trying other libraries. A Neural Network can be used for auto encoding the data for KNN clustering which would provide the unsupervised models something to compete against the supervised models.