RECOMMENDER SYSTEMS

Adrian P. Bustamante, Ph.D.
adrianpebus@gmail.com

OUTLINE

Introduction

EDA

Feature Engineering

Recommender systems

Conclusion

INTRODUCTION

We consider data about online courses and user interations with the courses. The objective of this notebook is to create a small survey of recommender systems to understand the main ideas behind contert-base filtering and collaborative filtering.

This work is based on the Machine Learning Capstone Project at https://www.coursera.org/learn/machine-learning-capstone?specialization=ibm-machine-learning

A jupyter notebook with the whole implementation can be found at: https://github.com/adrian-pbustamante/Recommender-systems-online-courses/blob/main/recomender_system.ipynb

ABOUT THE DATA

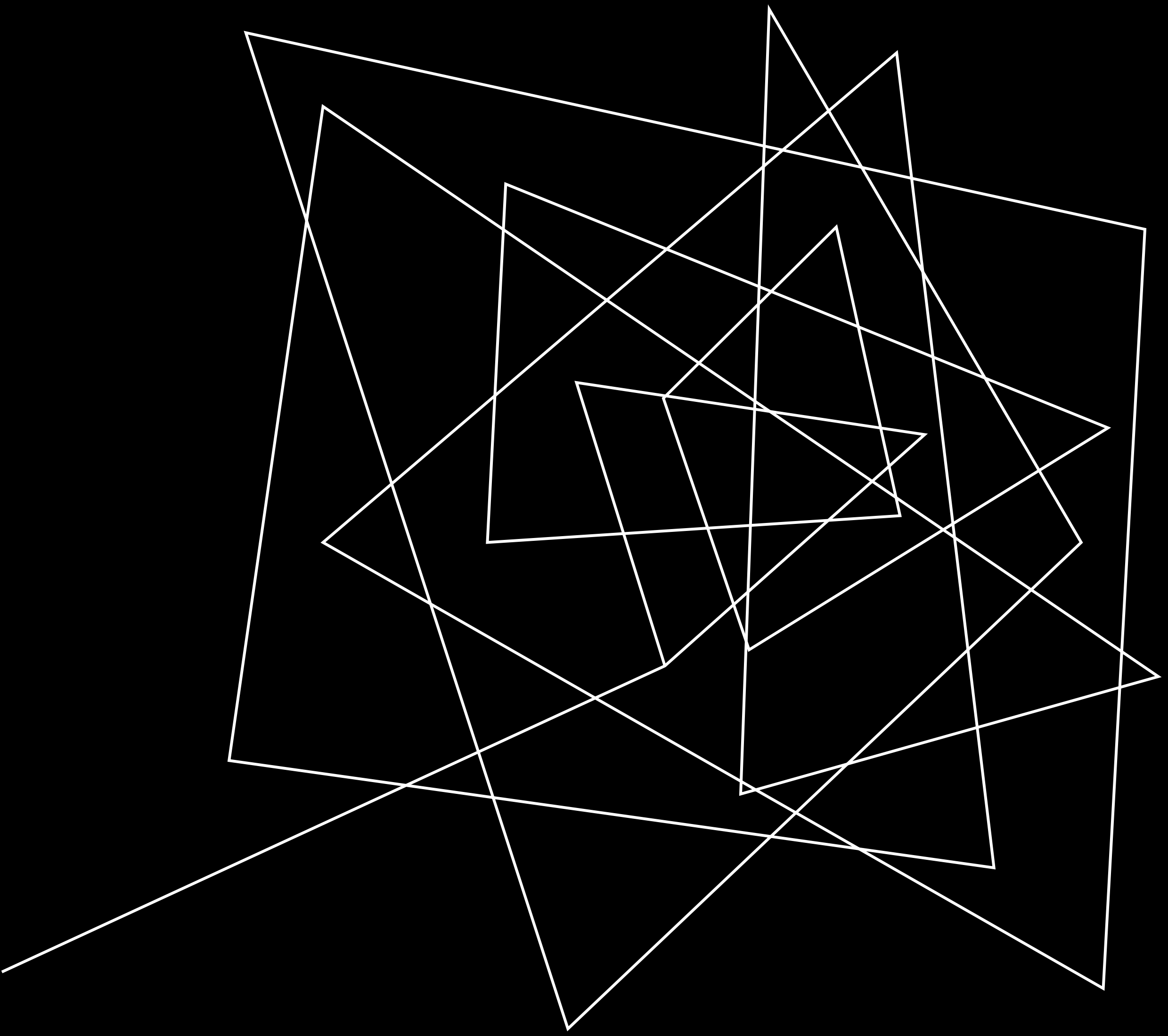
We start with 3 Data Frames

- **Course_df contains the courses id's, names and the genres of the courses in sparse matrix form**
- **course_content_df contains course id, title of the course, and a description of the course.**
- **ratings_df contains user id's , courses taken, and the rating given to each course taken.** The **rating** column consists of three potential values:

A rating of 5 signifies that users who have enrolled in the course find it excellent and have given it the highest rating, thus recommending it to other learners.

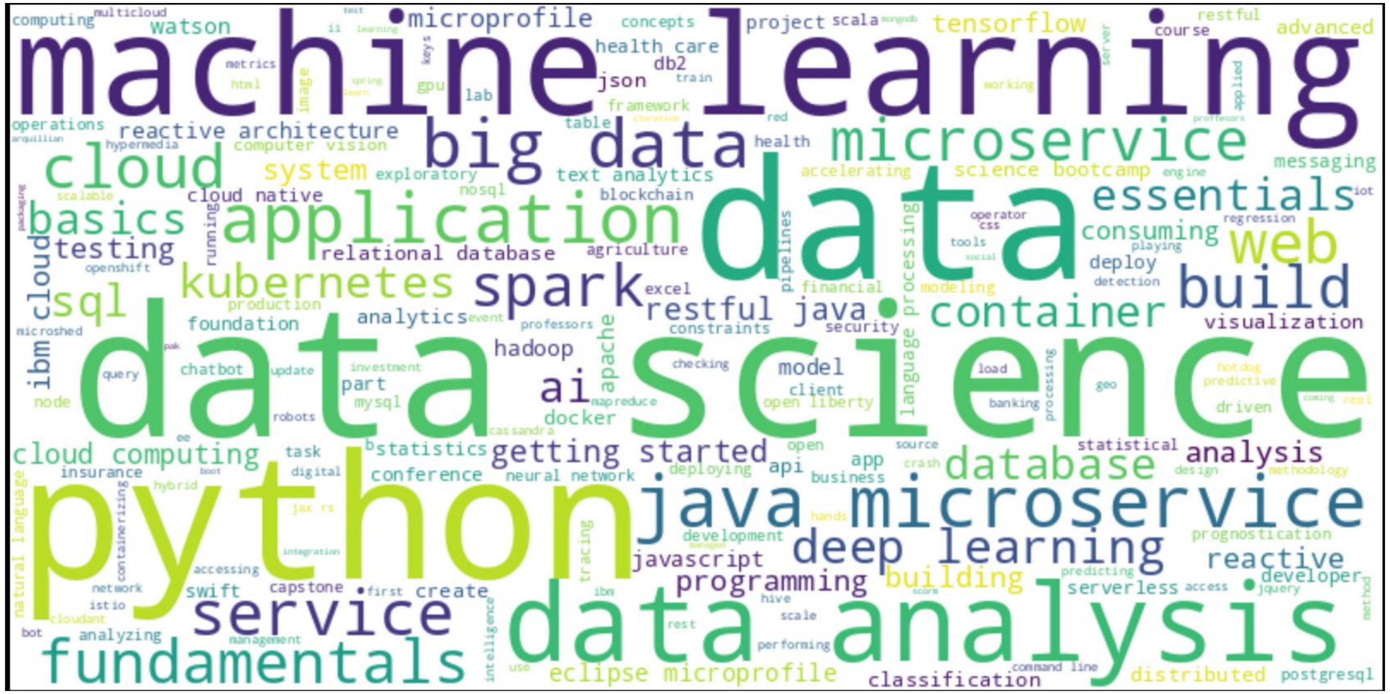
A rating of 4, indicates that the enrolled users perceive the course as good and will recommend to the other learners,but suggest minor improvements.

A rating of 3indicates that enrolled users find the course below expectations and need significant modifications.



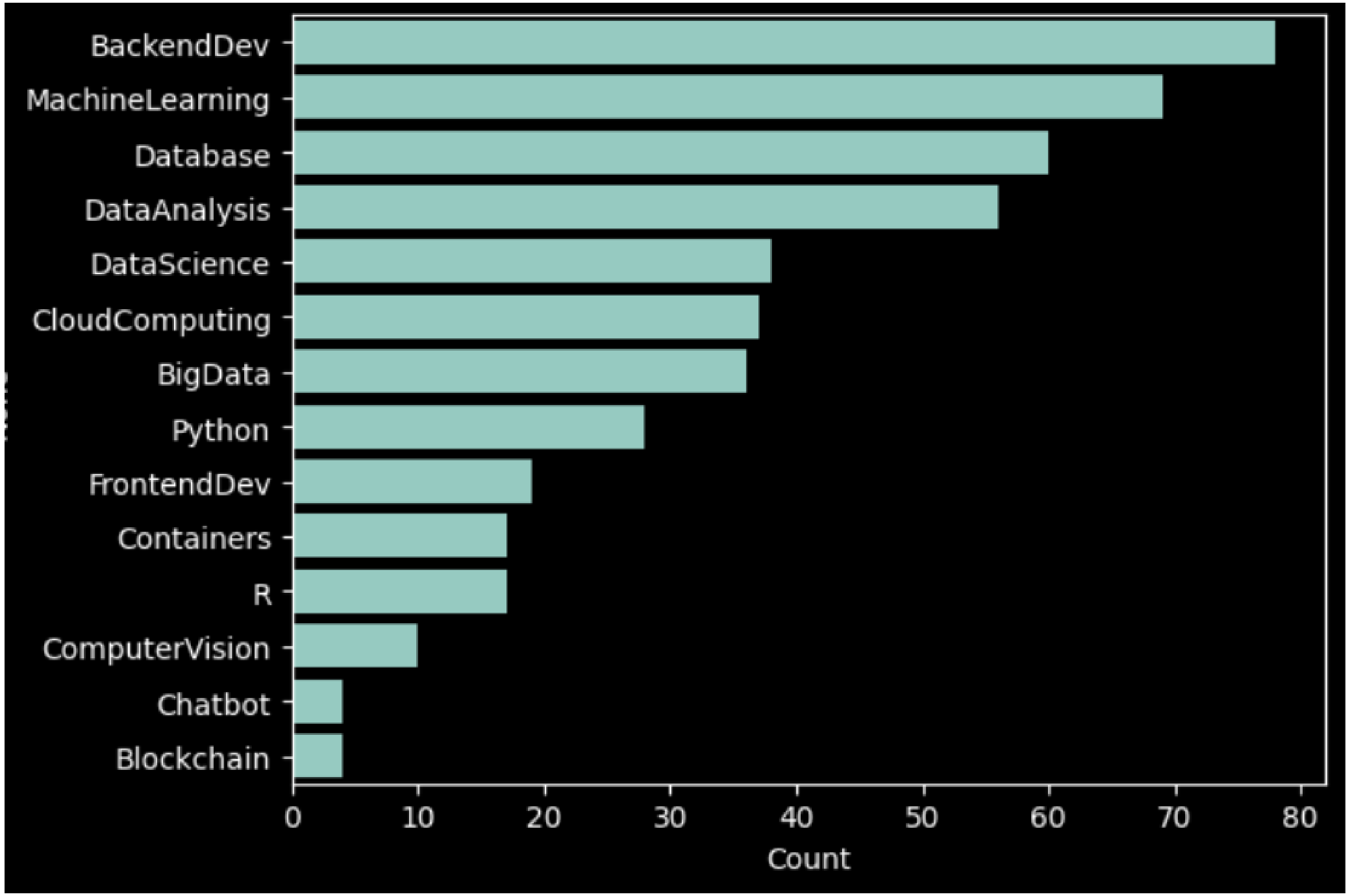
EXPLORATORY DATA ANALYSIS (EDA)

WORD CLOUD FROM COURSE TITLES



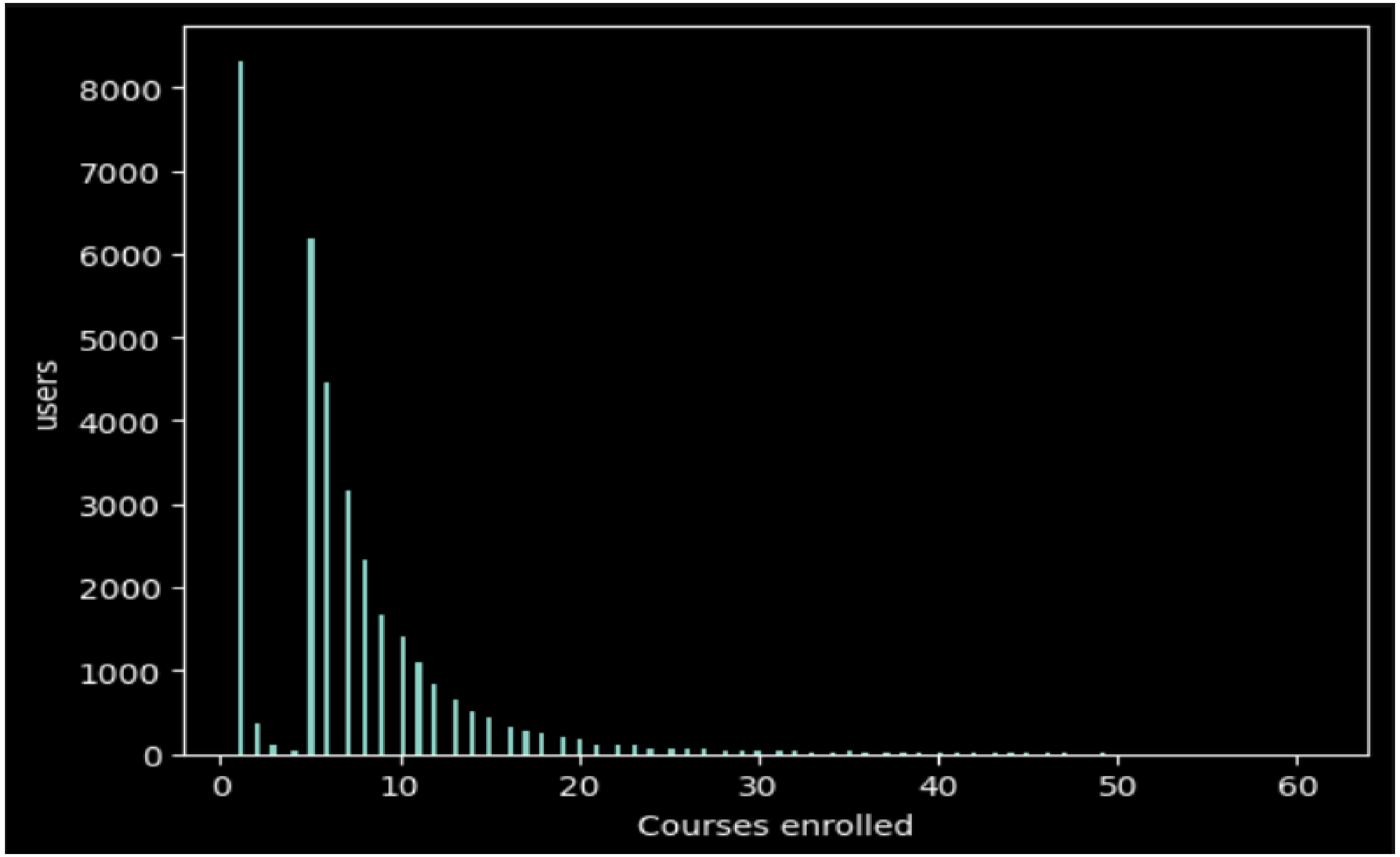
There are mainly IT related words.

GENRES DISTRIBUTION

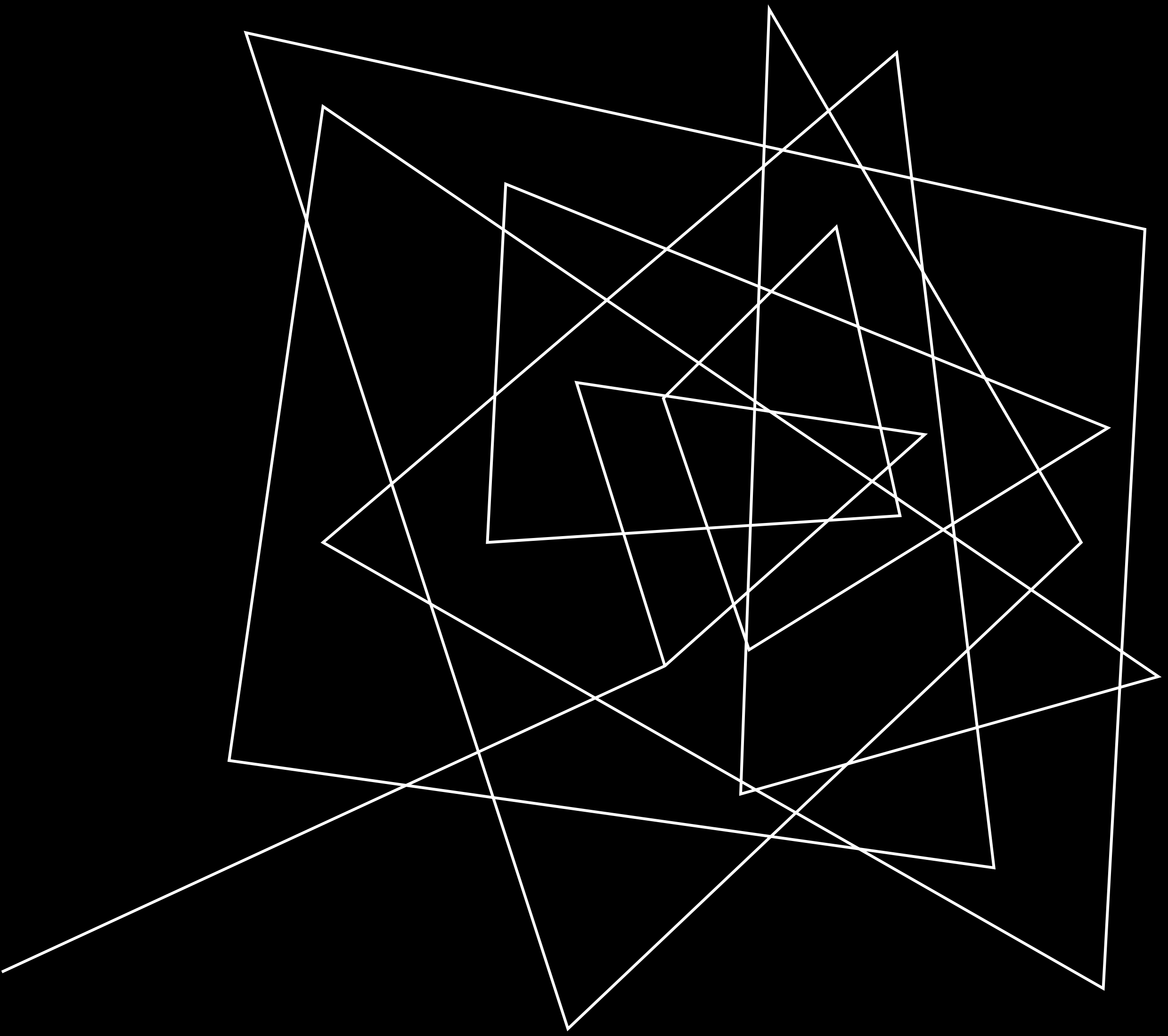


In total we have 307 courses.

ENROLLMENT DISTRIBUTION



In total 33901 users



FEATURE ENGINEERING

EXTRACTED BAG OF WORDS (BOW) FROM COURSE CONTENT

```
course_content_df.head()
```

	COURSE_ID	TITLE	DESCRIPTION
0	ML0201EN	robots are coming build iot apps with watson ...	have fun with iot and learn along the way if ...
1	ML0122EN	accelerating deep learning with gpu	training complex deep learning models with lar...
2	GPXX0ZG0EN	consuming restful services using the reactive ...	learn how to use a reactive jax rs client to a...
3	RP0105EN	analyzing big data in r using apache spark	apache spark is a popular cluster computing fr...
4	GPXX0Z2PEN	containerizing packaging and running a sprin...	learn how to containerize package and run a ...

```
bow_df
```

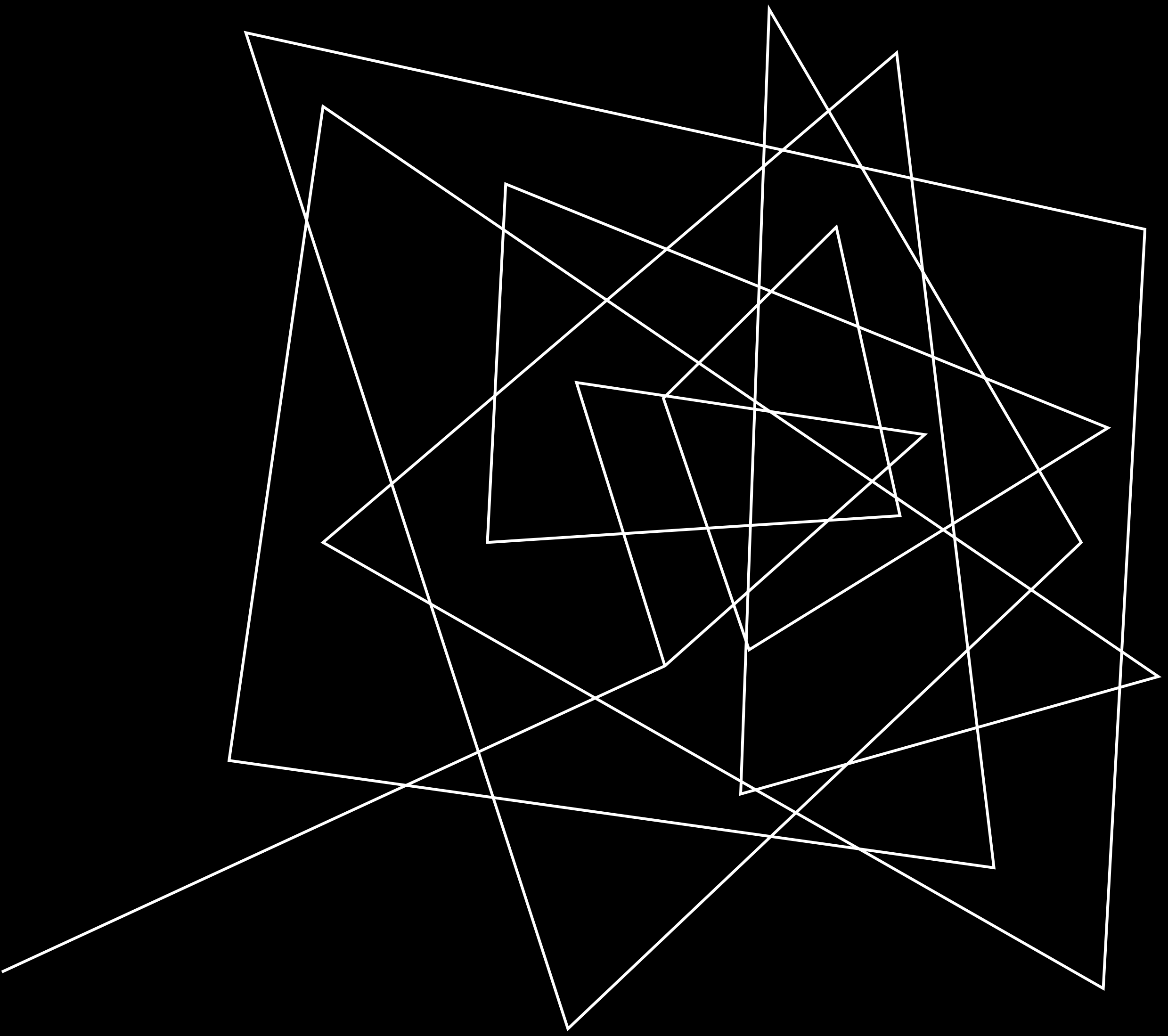
	course_index	course_id	token	bow
0	0	ML0201EN	ai	2
1	0	ML0201EN	apps	2
2	0	ML0201EN	build	2
3	0	ML0201EN	cloud	1
4	0	ML0201EN	coming	1
...	...	...	...	...
10358	306	excourse93	modifying	1
10359	306	excourse93	objectives	1
10360	306	excourse93	pieces	1
10361	306	excourse93	plugins	1
10362	306	excourse93	populate	1

FIND COURSE SIMILARITY USING BOW

```
a = random.choice(course_ids)
print('Chosen course: ', course_content_df[course_content_df['COURSE_ID']==a]['TITLE'].values)
print('Similar courses: ')
for i in similar_courses(a)['course_id'].values:
    print(course_content_df[course_content_df['COURSE_ID']==i]['TITLE'].values)
```

```
Chosen course:  ['cloud computing applications  part 2  big data and applications in the cloud']
Similar courses:
['cloud computing applications  part 1  cloud systems and infrastructure']
['big data modeling and management systems']
['introduction to big data']
['\ndistributed computing with spark sql']
['introduction to data analytics']
['fundamentals of big data']
['foundations for big data analysis with sql']
['getting started with the data  apache spark makers build']
['analyzing big data in r using apache spark']
['\nsql for data science']
```

Cosine similarity was used on BoW.



RECOMMENDER SYSTEMS

CONTENT BASED COURSE RECOMMENDER SYSTEM USING USER PROFILE AND COURSE GENRES

The most common type of content-based recommendation system is to recommend items to users based on their profiles. The user's profile revolves around that user's preferences and tastes. It is shaped based on user ratings, including the number of times a user has clicked on different items or liked those items.

The recommendation process is based on the similarity between those items. The similarity or closeness of items is measured based on the similarity in the content of those items. When we say content, we're talking about things like the item's category, tag, genre, and so on. Essentially the features about an item.

For online course recommender systems, we already know how to extract features from courses (such as genres or BoW features). Next, based on the course genres and users' ratings, we want to further build user profiles (if unknown).

A user profile can be seen as the user feature vector that quantifies a user's learning interests.

With the user profile feature vectors and course genre feature vectors constructed, we can use several computational methods, such as a simple dot product, to compute or predict an interest score for each course and recommend those courses with high interest scores.

Generate user profiles using course genres and ratings

To compute the user profile, u_i, for user i we consider the courses user i has taken. We store the ratings in the vector $v_i \in \mathbb{R}^{1\times r_i}$ (user i has rated/taken r_i courses). For each course taken by user i we consider the features of each course, we store courses genres in a sparse matrix $M_i \in \mathbb{R}^{r_i \times 14}$ (we only have 14 genres, see course_df). Finally, we compute u_i as follows

$$u_i = v_i M_i \in \mathbb{R}^{1\times 14}.$$

u_i quantifies the interest of user i, based on the ratings the user has provided.

```
profiles
```

	user	Database	Python	CloudComputing	DataAnalysis	Containers	MachineLearning	ComputerVision	DataScience
0	2	74.0	21.0	8.0	66.0	5.0	50.0	0.0	40.0
1	4	81.0	4.0	8.0	57.0	0.0	29.0	0.0	41.0
2	5	47.0	16.0	37.0	50.0	0.0	58.0	0.0	41.0
3	7	4.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0
4	8	13.0	0.0	0.0	8.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...
33896	2102054	5.0	4.0	5.0	7.0	0.0	0.0	0.0	9.0
33897	2102356	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0
33898	2102680	4.0	10.0	8.0	0.0	0.0	21.0	0.0	14.0
33899	2102983	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0
33900	2103039	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0

Recommendation scores

To compute recommendation scores for user i we consider the matrix storing the genres of the courses that user i has not seen/taken, denote it by $A_i \in \mathbb{R}^{s_i \times 14}$. The recommendation scores for user i, then, are given be

$$A_i u_i^\top \in \mathbb{R}^{s_i \times 1}$$

where u_i is user's i profile vector. The recommended courses will be the ones with the largest scores.

Recommendations:

```
User 1933599 has taken the following courses:
['introduction to containers  kubernetes  and openshift v2']
['beyond the basics  istio and ibm cloud kubernetes service']
['data visualization with python']
['statistics 101']
['getting started with microservices with istio and ibm cloud kubernetes service']
['data analysis with python']
--------------------
Recommended Courses
['programming foundations with javascript  html and css']
['checking the health of java microservices by using kubernetes readiness and liveness probes']
['fundamentals of digital image and video processing']
['introduction to data science']
['machine learning with big data']
['interactivity with javascript and jquery']
['how to build watson ai and swift apis and make money']
['deploy an ai powered discord bot with a voice']
['using clustering methods for investment portfolio analysis']
['database architecture  scale  and nosql with elasticsearch']
['building fault tolerant microservices with the  fallback annotation']
['enabling cross origin resource sharing  cors  in a restful java microservice']
['build your own chatbots']
['python and statistics for financial analysis']
['python scripting  files  inheritance  and databases']
```

CONTENT BASED COURSE RECOMMENDER SYSTEM USING COURSE SIMILARITIES

We use the course similarity matrix (computed from BoW using cosine similarity) to recommend new courses which are similar to a user's presently enrolled courses.

```
sim_mat_courses.head()
```

	ML0201EN	ML0122EN	GPXX0ZG0EN	RP0105EN	GPXX0Z2PEN	CNSC02EN	DX0106EN	GPXX0FTCEN	RAVSCT
ML0201EN	1.000000	0.088889	0.088475	0.065556	0.048810	0.104685	0.065202	0.143346	0.00
ML0122EN	0.088889	1.000000	0.055202	0.057264	0.012182	0.078379	0.032545	0.119251	0.04
GPXX0ZG0EN	0.088475	0.055202	1.000000	0.026463	0.039406	0.000000	0.000000	0.154303	0.00
RP0105EN	0.065556	0.057264	0.026463	1.000000	0.000000	0.250490	0.390038	0.000000	0.00
GPXX0Z2PEN	0.048810	0.012182	0.039406	0.000000	1.000000	0.000000	0.000000	0.085126	0.00

Recommendations --->

```
User 784146 has taken the following courses:
['spark fundamentals i']
['scala 101']
['introduction to data science']
['hadoop 101']
['build your own chatbot']
['big data 101']
--------------------
Recommended Courses
['build your own chatbots']
['data science with open data']
['introduction to big data']
['a crash course in data science']
['foundations for big data analysis with sql']
['fundamentals of big data']
['data science fundamentals for data analysts']
['sql access for hadoop']
['big data modeling and management systems']
['data science bootcamp']
['introduction to data analytics']
['\nsql for data science']
['data analysis using python']
['data analysis using python']
['working with big data']
```

CLUSTERING BASED COURSE RECOMMENDER SYSTEM USING PCA AND KMEANS

We can perform clustering algorithms such as K-means or DBSCAN to group users with similar learning interests.For each user group, we can come up with a list of popular courses.

If we know a user belongs to a group, we may recommend the most enrolled courses to them and it is very likely the user will be interested in them.

We cluster users base on profile:

profiles

	user	Database	Python	CloudComputing	DataAnalysis	Containers	MachineLearning	ComputerVision	DataScience
0	2	74.0	21.0	8.0	66.0	5.0	50.0	0.0	40.0
1	4	81.0	4.0	8.0	57.0	0.0	29.0	0.0	41.0
2	5	47.0	16.0	37.0	50.0	0.0	58.0	0.0	41.0
3	7	4.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0
4	8	13.0	0.0	0.0	8.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...
33896	2102054	5.0	4.0	5.0	7.0	0.0	0.0	0.0	9.0
33897	2102356	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0
33898	2102680	4.0	10.0	8.0	0.0	0.0	21.0	0.0	14.0
33899	2102983	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0
33900	2103039	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0

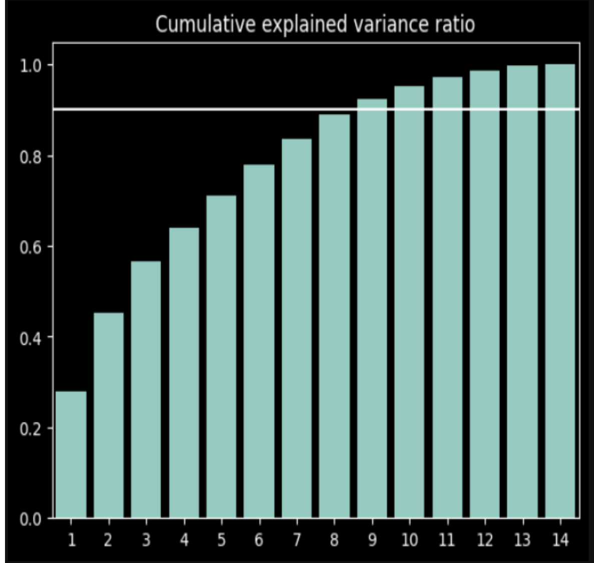
Using PCA on user profile feature vectors to reduce dimensions

```
pca = sklearn.decomposition.PCA()
pca.fit(features)
pca.components_.shape, pca.n_components_
```

```
((14, 14), 14)
```

```
ax=plt.gca()
sns.barplot(x=range(1,15),y=np.cumsum(pca.explained_variance_ratio_), ax=ax).axhline(0.9)
ax.set_title('Cumulative explained variance ratio')
```

```
Text(0.5, 1.0, 'Cumulative explained variance ratio')
```



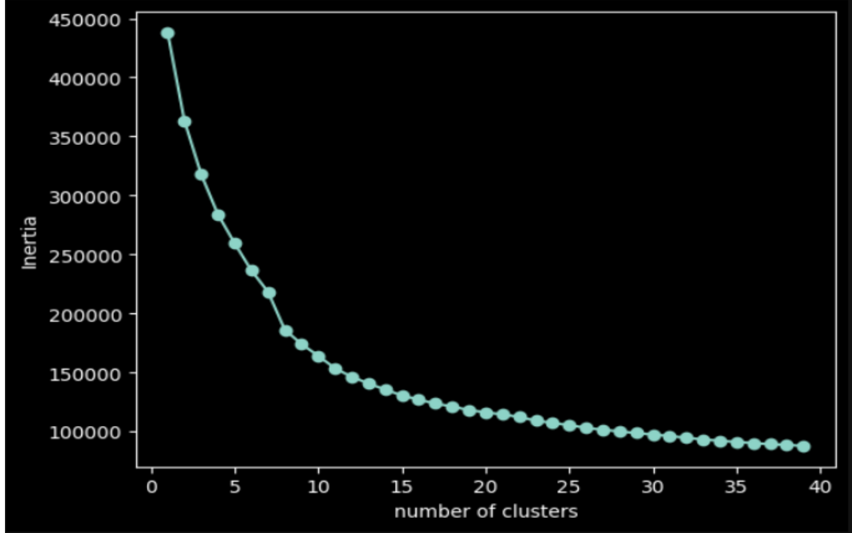
We can see above that using 9 components we can keep about 90% of the variances

We choose K=20 clusters based on Inertia

```
inertia = []
list_num_clusters = list( range(1,40) )
# Find an optimized number of neighbors k from a candidate list such as
for num_clusters in list_num_clusters:
    km=sklearn.cluster.KMeans(n_clusters=num_clusters, random_state=rs)
    km.fit(features_pca)
    inertia.append(km.inertia_)

plt.plot(list_num_clusters,inertia)
plt.scatter(list_num_clusters,inertia)
plt.xlabel('number of clusters')
plt.ylabel('Inertia')
```

```
Text(0, 0.5, 'Inertia')
```



The clustering-based recommender system first groups all users based on their profiles, and maintains a popular courses list for each group.

For any group member who needs course recommendations, the algorithm recommends the unselected courses from the popular course lists.

```
##recommended courses
print('--------------------')
print('Recommended Courses')
top_suggestions = recomend_courses_in_cluster(user_id=user_id)
for course in top_suggestions['item'].values:
    print(course_content_df[course_content_df['COURSE_ID']==course]['TITLE'].values)
```

```
User 1590388 has taken the following courses:
['spark fundamentals i']
['mapreduce and yarn']
['moving data into hadoop']
['hadoop 101']
['big data 101']
--------------------
Recommended Courses
['python for data science']
['python for data science']
['introduction to data science']
['data analysis with python']
['data visualization with python']
['machine learning with python']
['build your own chatbot']
['r for data science']
['introduction to data science']
['blockchain essentials']
['introduction to data science']
['cloud native security conference   data security']
['data analysis with python']
['accessing hadoop data using hive']
['deep learning 101']
```

COLLABORATIVE FILTERING BASED RECOMMENDER SYSTEM USING NEAREST NEIGHBORS

Collaborative filtering is probably the most commonly used recommendation algorithm, there are two main types of methods:

- **User-based** collaborative filtering is based on the user similarity or neighborhood
- **Item-based** collaborative filtering is based on similarity among items

They both work similarly, let's briefly explain how user-based collaborative filtering works.

User-based collaborative filtering looks for users who are similar. This is very similar to the user clustering method done previously; where we employed explicit user profiles to calculate user similarity. To determine if two users are similar we use the user-item interaction matrix.

User-item interaction matrix

```
ratings_sparse
```

	user	AI0111EN	BC0101EN	BC0201EN	BC0202EN	BD0101EN	BD0111EN	BD0115EN	BD0121EN	BD0123EN	...	SW0101EN	SW0201EN	TA0105
0	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	...	0.0	0.0	0.0
1	4	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
2	5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
3	9	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	...	0.0	0.0	0.0
4	12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
15774	2099010	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
15775	2099019	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
15776	2100030	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
15777	2101535	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
15778	2102054	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0

To determine if two users are similar, we can simply calculate the similarities between their row vectors in the interaction matrix. Then based on the similarity measurements, we can find the k nearest neighbors as the similar users.

If we formulate the KNN based collaborative filtering, the predicted rating of user u to item i, $\hat{r}_{ui}$ is given by:

User-based collaborative filtering:

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} \text{similarity}(u, v) \cdot r_{vi}}{\sum\limits_{v \in N_i^k(u)} \text{similarity}(u, v)}$$

Item-based collaborative filtering:

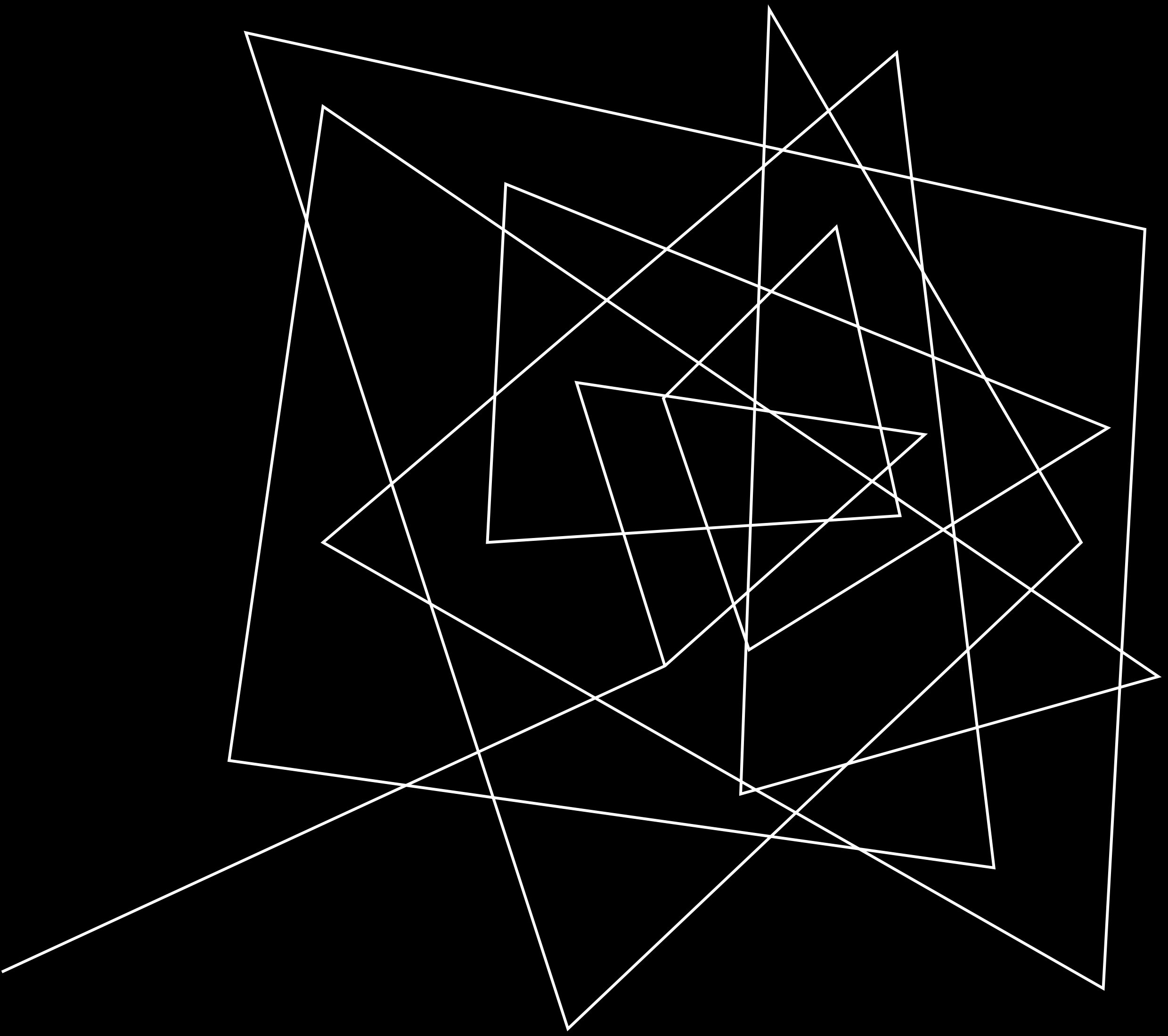
$$\hat{r}_{ui} = \frac{\sum\limits_{j \in N_u^k(i)} \text{similarity}(i, j) \cdot r_{uj}}{\sum\limits_{j \in N_u^k(i)} \text{similarity}(i, j)}$$

Here $N_i^k(u)$ notates the nearest k neighbors of u.

```
print('--------------------')
print('Recommended Courses')
top_suggestions = ubased_ratings_neigbors(user_id=user_id).head(15)
for course in top_suggestions['course_id'].values:
    print(course_content_df[course_content_df['COURSE_ID']==course]['TITLE'].values)
```

```
User 1883674 has taken the following courses:
['statistics 101']
['introduction to data science']
['data science hands on with open source tools']
['digital analytics   regression']
['data visualization with python']
['r for data science']
['data science methodology']
['data analysis with python']
['introduction to cloud']
['python for data science']
--------------------
Recommended Courses
['hybrid cloud conference  ai pipelines lab']
['build your own chatbots']
['text analytics 101']
['accelerating deep learning with gpus']
['data   ai  jumpstart your journey']
['watson analytics for social media']
['building cloud native and multicloud applications']
['building cloud native and multicloud applications']
['data   ai essentials']
['data visualization with r']
['cloud native security conference   data security']
['node red  basics to bots']
['modernizing java ee applications']
['scalable web applications on kubernetes']
['analyzing big data in r using apache spark']
```

The recommended courses are ones with the highest predicted ratins. ---------->



CONCLUSION

In this notebook we consired a dataset containing information of online courses as well as information about users interaction with the courses.

We construct different recommender systems using content-base and collaborative filtering with the objective to explore the main ideas to construct recommender systems.

This work can be expanded by exploring more complex approaches to recommender systems, e.g. considere hybrid filtering.

THANK YOU

Adrian P. Bustamante, Ph.D.

adrianpebus@gmail.com