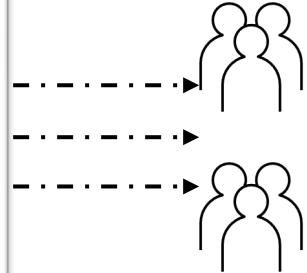
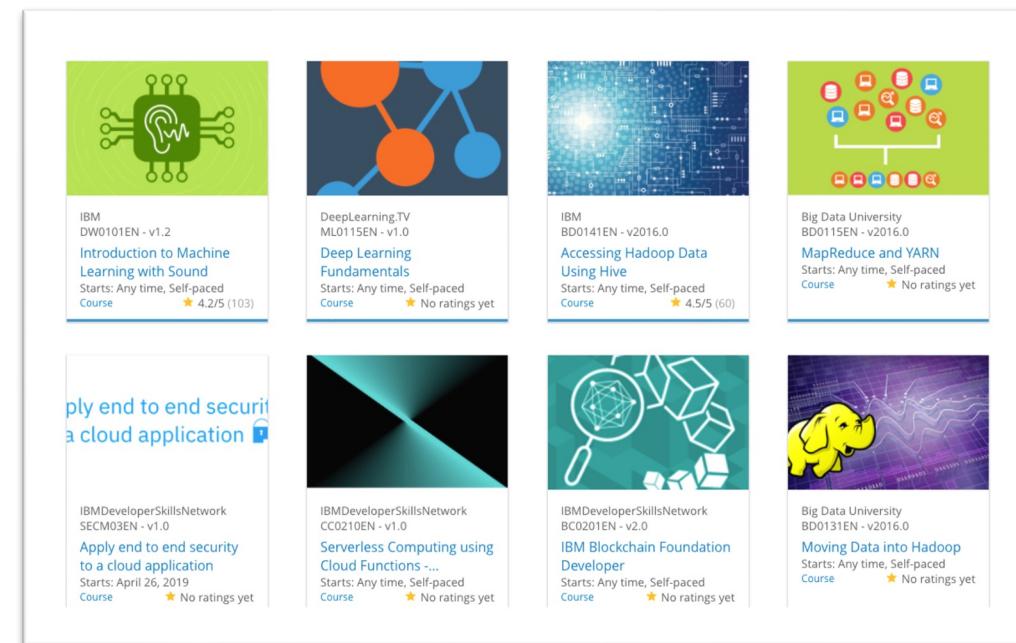


# Build a Personalized Online Course Recommender System with Machine Learning

BK

May 22, 2023



# 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

AI Training Room is a Massive Open Online Courses (MOOCs) startup for learners across the world. The course catalogue provides various subjects to learn leading technologies. The demand grows rapidly and reached millions of customers in a very short time. A central concern of the company is to find new interesting courses for students to support their learning paths. For that reason, a Proof of Concept (PoC) for a recommender system (RS) is launched to improve the students' learning experience and increase the interaction with more courses. As a side effect of an established RS, the AI Training Room's revenue may be increase over time.

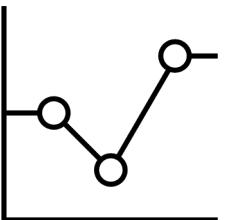
- Problem states

- Which model is the best for our RS?
- How much time does the training take?
- Which model is the most economical for our RS?

- Hypothesis

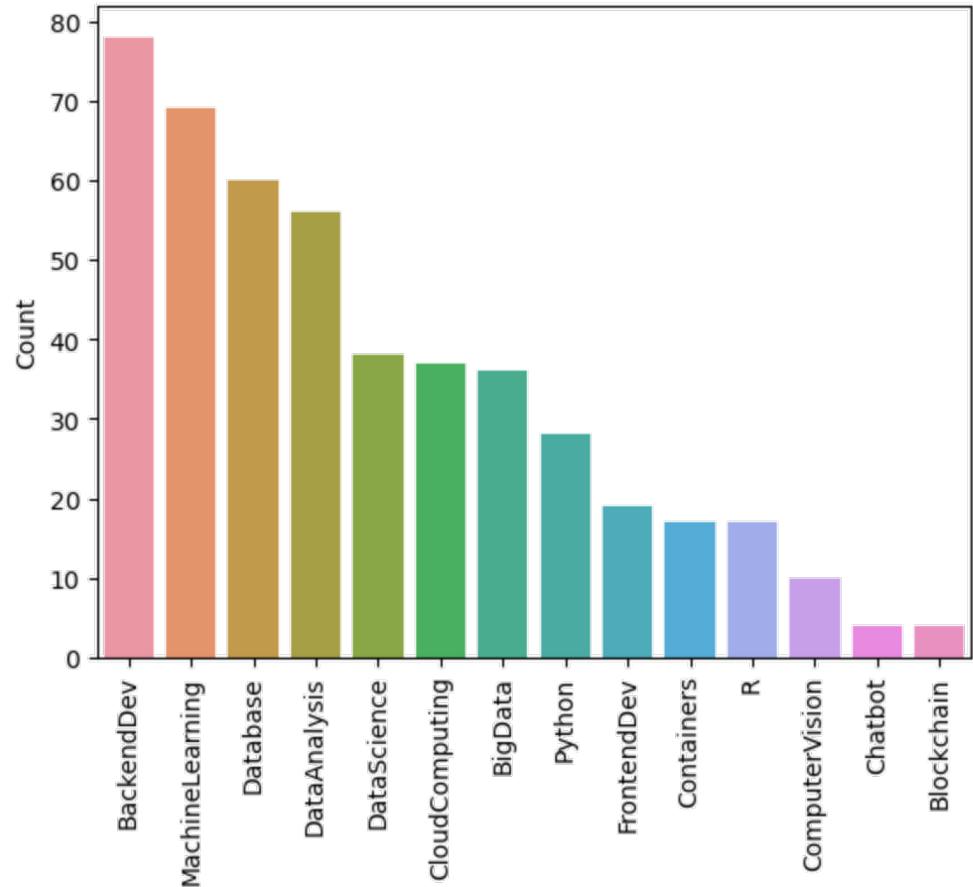
Using enrollment data and  
course properties will provide sufficient information to build  
a recommender system to evaluate new courses for our students.

# Exploratory Data Analysis



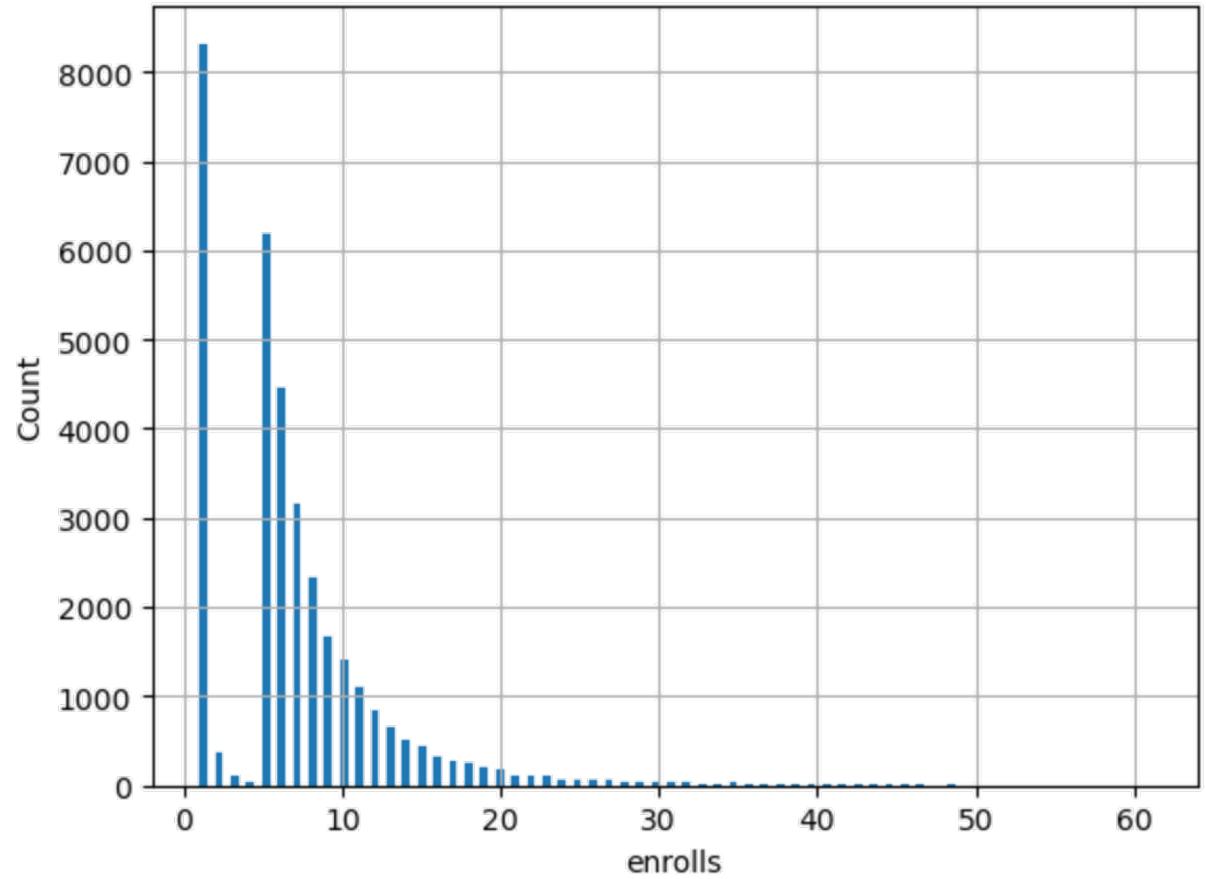
# Course counts per genre

- 14 genres are offered.
- Most courses are backend development, machine learning and databases.
- Only two genres about programming languages.
- Very little courses about chatbots and blockchain.



# Course enrollment distribution

- A bimodal distribution.
- Most students enrolled only one course.
- Little students enrolled two, three or four courses.
- Another peak at five enrolls.
- From that one the student's interest for courses dropped.



# 20 most popular courses

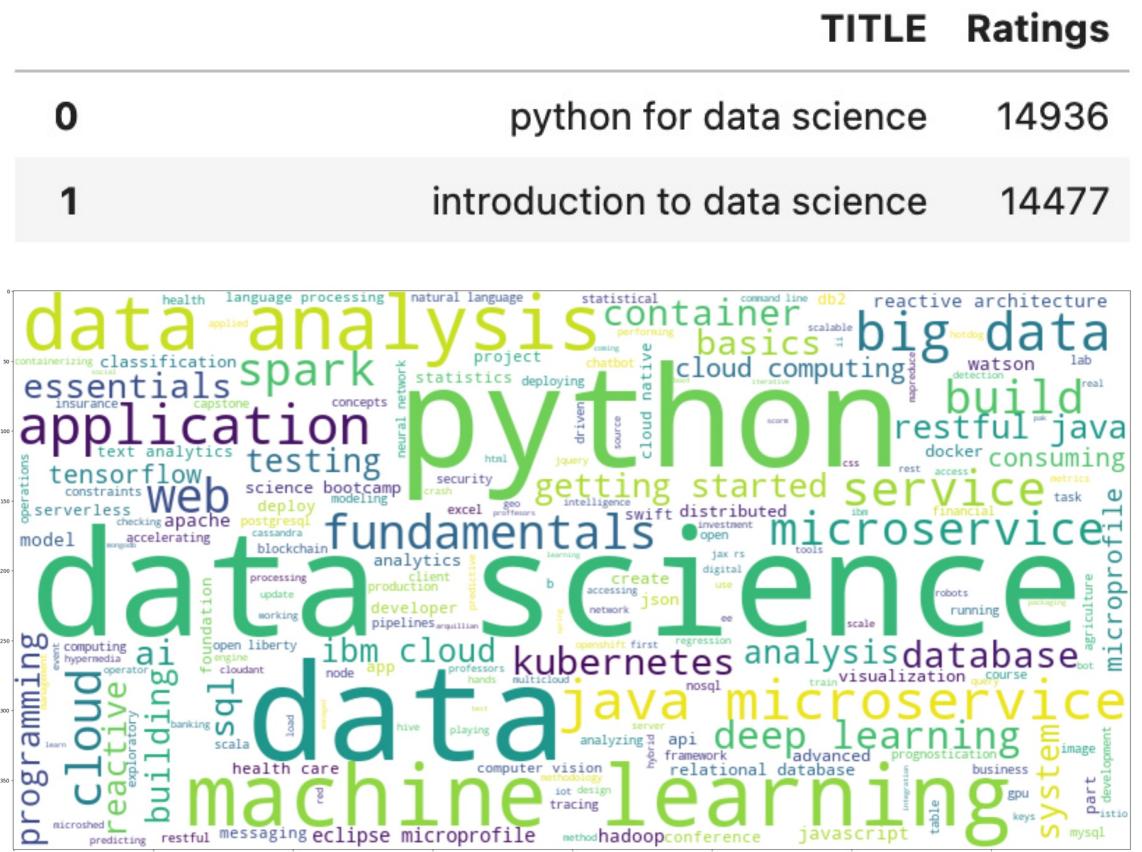
- Most courses are entry level.
- Overbalanced on data science.
- Only two programming languages.
  - Weighted to Python.
  - #1 is data science subject.
- Top rating on big data & hadoop.
- Machine learning starts at #7.
  - First course is intermediate level.



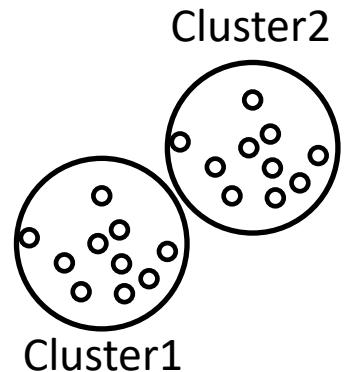
	TITLE	Ratings
0	python for data science	14936
1	introduction to data science	14477
2	big data 101	13291
3	hadoop 101	10599
4	data analysis with python	8303
5	data science methodology	7719
6	machine learning with python	7644
7	spark fundamentals i	7551
8	data science hands on with open source tools	7199
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

# Word cloud of course titles

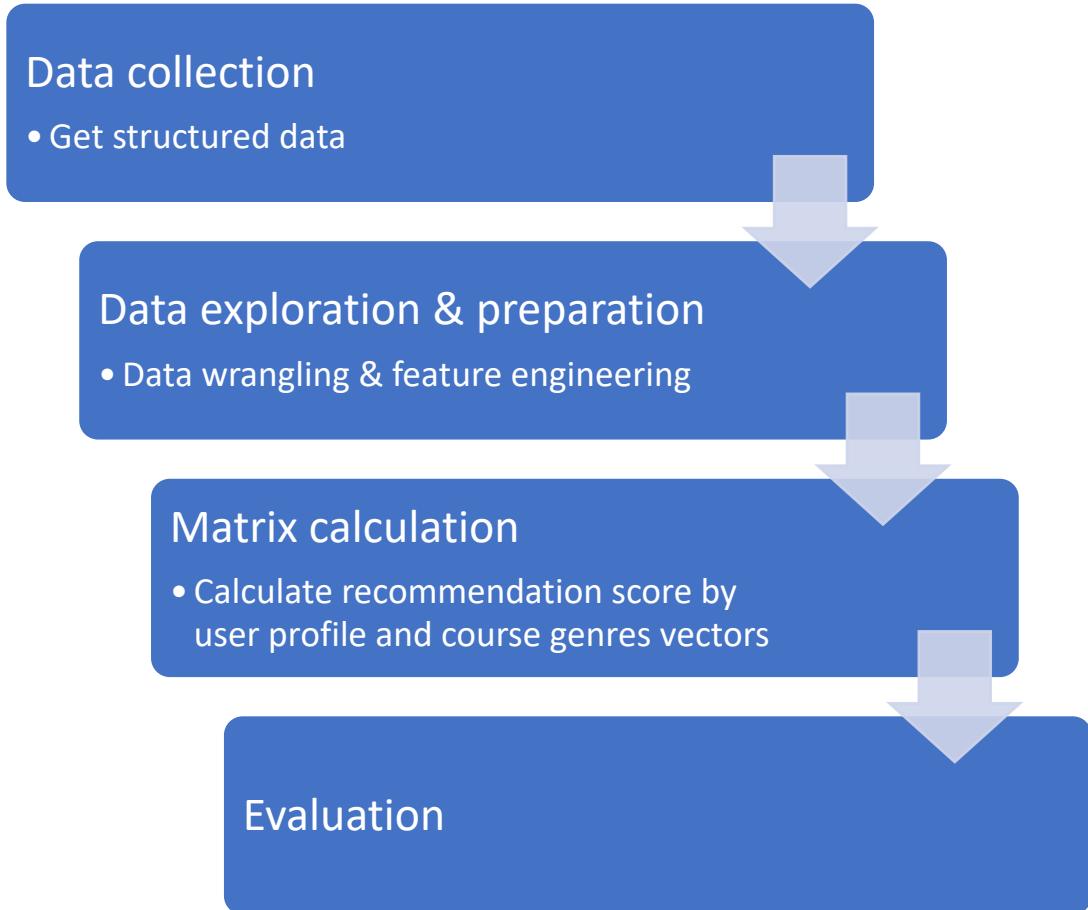
- The most frequent buzzwords used for course titles are data, data science and python.
  - There seems a relationship between the frequency of these buzzwords used for course titles and the most popular courses.



# Content-based Recommender System using Unsupervised Learning



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



# Matrix calculation

Let  $A$  be a  $m \times n$  matrix with  $m$  students and  $n$  courses where each element denotes a rating of the student  $i$  about the course  $j$  within the interval  $[0,3]$ . A zero means that the student  $i$  has no interaction with the course  $j$ , a one means that the student  $i$  just browsed the course  $j$ , a two means that the student  $i$  just audited the course  $j$  without completing it and a three means that the student  $i$  completed the course  $j$  and earned a certificate.

Let  $B$  be a  $n \times p$  matrix with  $n$  courses and  $p$  genres where each element denotes that course  $i$  belongs to genre  $j$  if and only if the value is 1.

Then  $A \cdot B$  is the product matrix of the size  $m \times p$  with  $m$  students and  $p$  genres where each element denotes a rating of the student  $i$  about the genre  $j$  within the same interval as the elements of the matrix  $A$ .

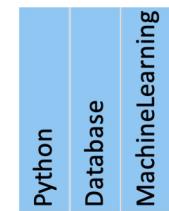


$$\begin{matrix} \text{Student 1} \\ \text{Student 2} \end{matrix} A = \begin{pmatrix} 3 & 0 \\ 2 & 2 \end{pmatrix}$$

Rating:  $a_{ij} \in [0,3]$



$$\begin{matrix} \text{Machine Learning with Python} \\ \text{SQL with Python} \end{matrix} B = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$



$$\begin{matrix} \text{Student 1} \\ \text{Student 2} \end{matrix} A \cdot B = \begin{pmatrix} 3 & 0 & 3 \\ 2 & 2 & 0 \end{pmatrix}$$

# Matrix calculation and recommendation score

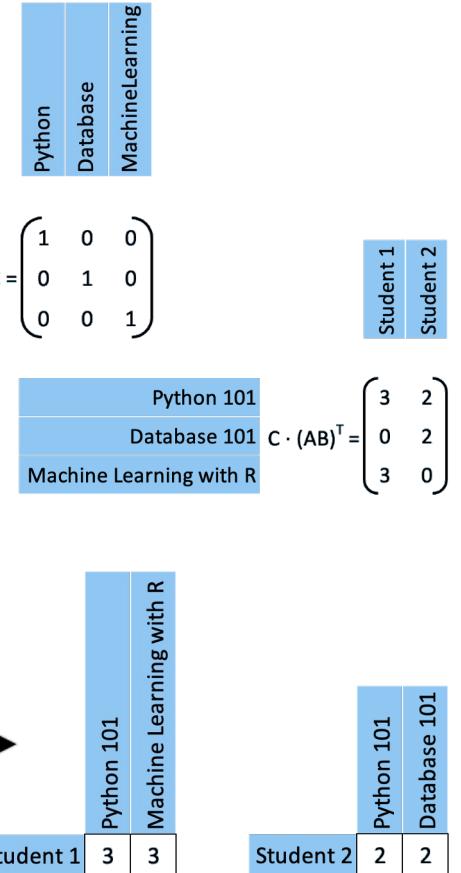
Let  $C$  be a  $p \times q$  matrix with  $m$  unseen or unselected courses and  $n$  genres where each element denotes that the unseen or unselected course  $i$  belongs to genre  $j$  if and only if the value is 1.

Then  $C \cdot (AB)^T$  is the product matrix of the size  $p \times m$  with  $p$  unseen or unselected courses and  $m$  students where each element denotes a rating of the unseen or unselected course  $i$  by the student  $j$  within the same interval as the elements of the matrix  $A$ .

A recommendation score threshold can be used to filter all unseen or unselected courses from the product matrix

$$C \cdot (AB)^T$$

to make appropriate recommendations to potential courses.



# Evaluation results of user profile-based recommender system

Using a `score_threshold = 10`,  
the average new courses  
recommended per student (in percentage) is

```
len(res_df) / len(res_df['USER'].unique())  
61.81828703703704
```

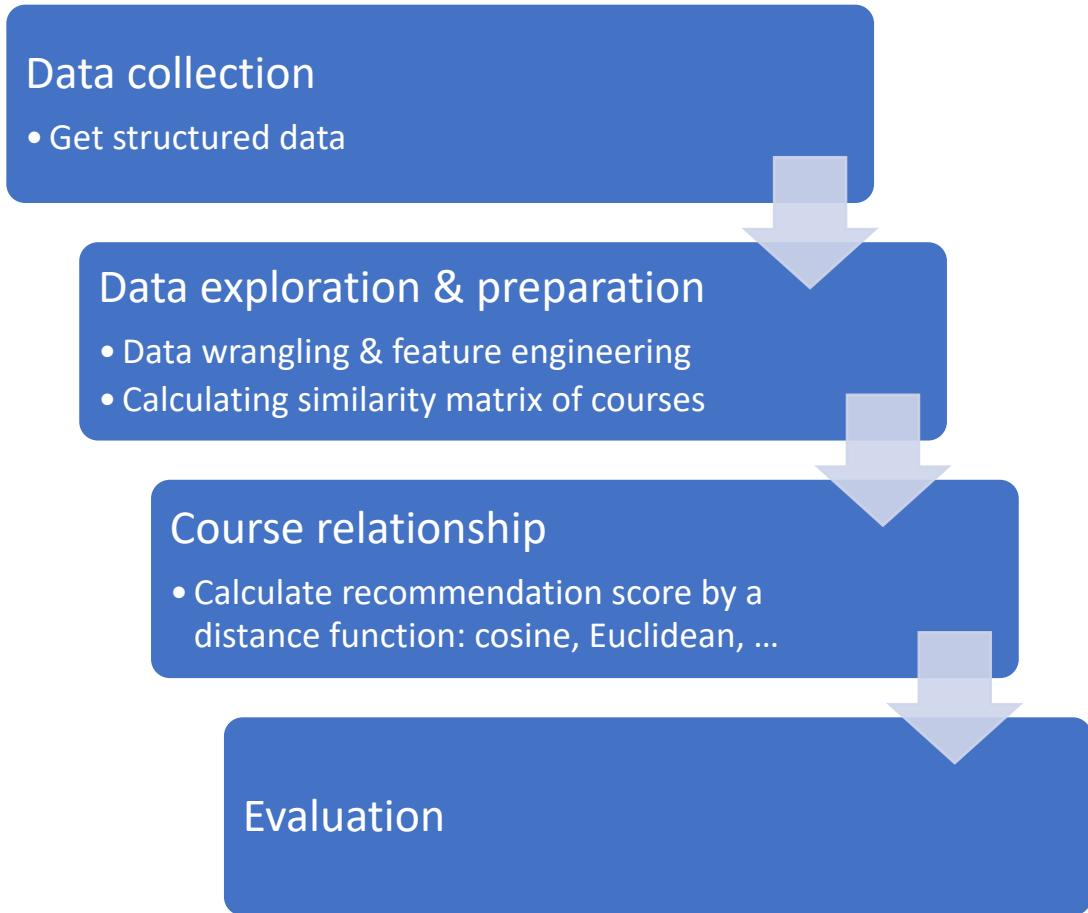
	USER	COURSE_ID	SCORE
0	37465	RP0105EN	27.0
1	37465	GPXX06RFEN	12.0
2	37465	CC0271EN	15.0
3	37465	BD0145EN	24.0
4	37465	DE0205EN	15.0
...	...	...	...
53406	2087663	excourse88	15.0
53407	2087663	excourse89	15.0
53408	2087663	excourse90	15.0
53409	2087663	excourse92	15.0
53410	2087663	excourse93	15.0

# Evaluation results of user profile-based recommender system

The top-10 commonly recommended courses across all test users.

	TITLE	COURSE_ID	SCORE
0	analyzing big data with sql	excourse73	14386.0
1	foundations for big data analysis with sql	excourse72	14386.0
2	getting started with the data apache spark ma...	TMP0105EN	14182.0
3	analyzing big data in r using apache spark	RP0105EN	13347.0
4	cloud computing applications part 2 big data...	excourse31	13036.0
5	text analytics at scale	TA0106EN	12910.0
6	spark overview for scala analytics	SC0103EN	12853.0
7	introduction to data science in python	excourse22	11875.0
8	applied machine learning in python	excourse21	11875.0
9	accelerating deep learning with gpu	ML0122EN	11785.0

# Flowchart of content-based recommender system using course similarity



# Course relationship

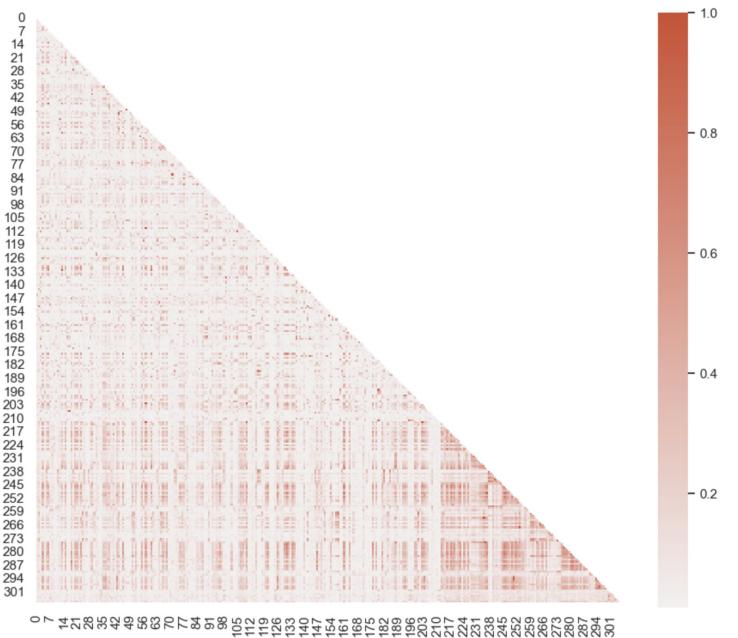
Let  $A$  be a  $m \times n$  matrix with  $m$  courses and  $n$  courses where each element denotes the similarity of the course  $i$  and the course  $j$  within the interval  $[0,1]$ . A zero means no similarity between both courses  $a_{ij}$  and a value one means there is similarity.

Similarity:  $a_{ij} \in [0,1]$

	Machine learning with R	
Machine learning with R		0.7 0.3
	Machine learning with Python	0.7 1 0.3
Predicting financial performance of a company		0.3 0.3 1

$$A = \begin{pmatrix} 1 & 0.7 & 0.3 \\ 0.7 & 1 & 0.3 \\ 0.3 & 0.3 & 1 \end{pmatrix}$$

Visualisation of a pre-made similarity matrix  $A$  that contains 306 courses. The similarities of the courses are highlighted by a heat map.



# Evaluation results of course similarity based recommender system

Using a `score_threshold = 10`,  
the average new/unseen courses  
recommended per student (in percentage) is

```
len(res_df) / len(res_df['USER'].unique())
```

20.733

The used distance function is **cosine**.

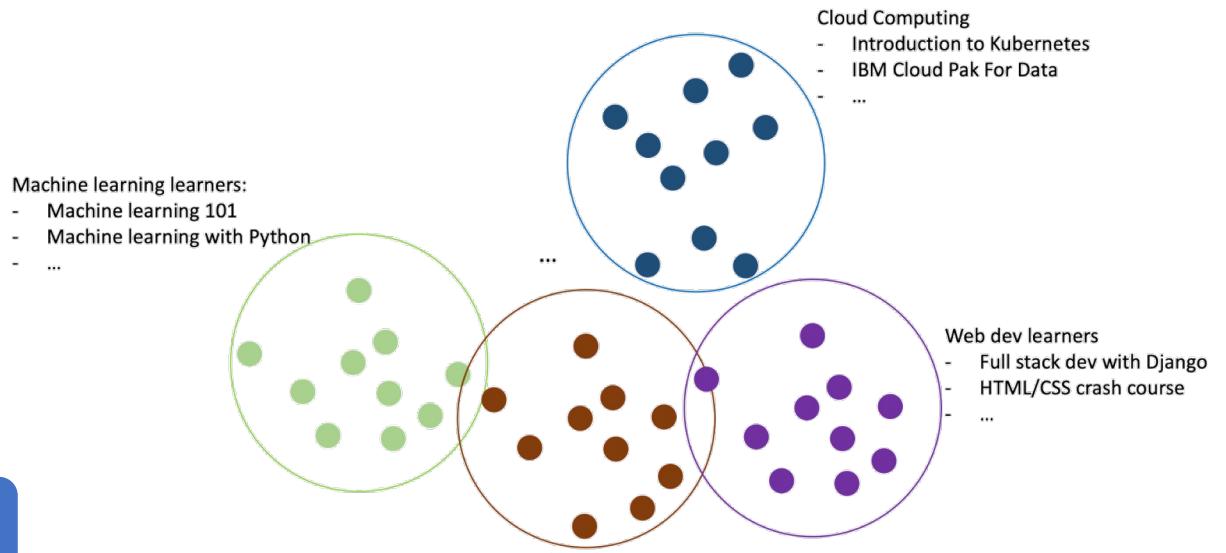
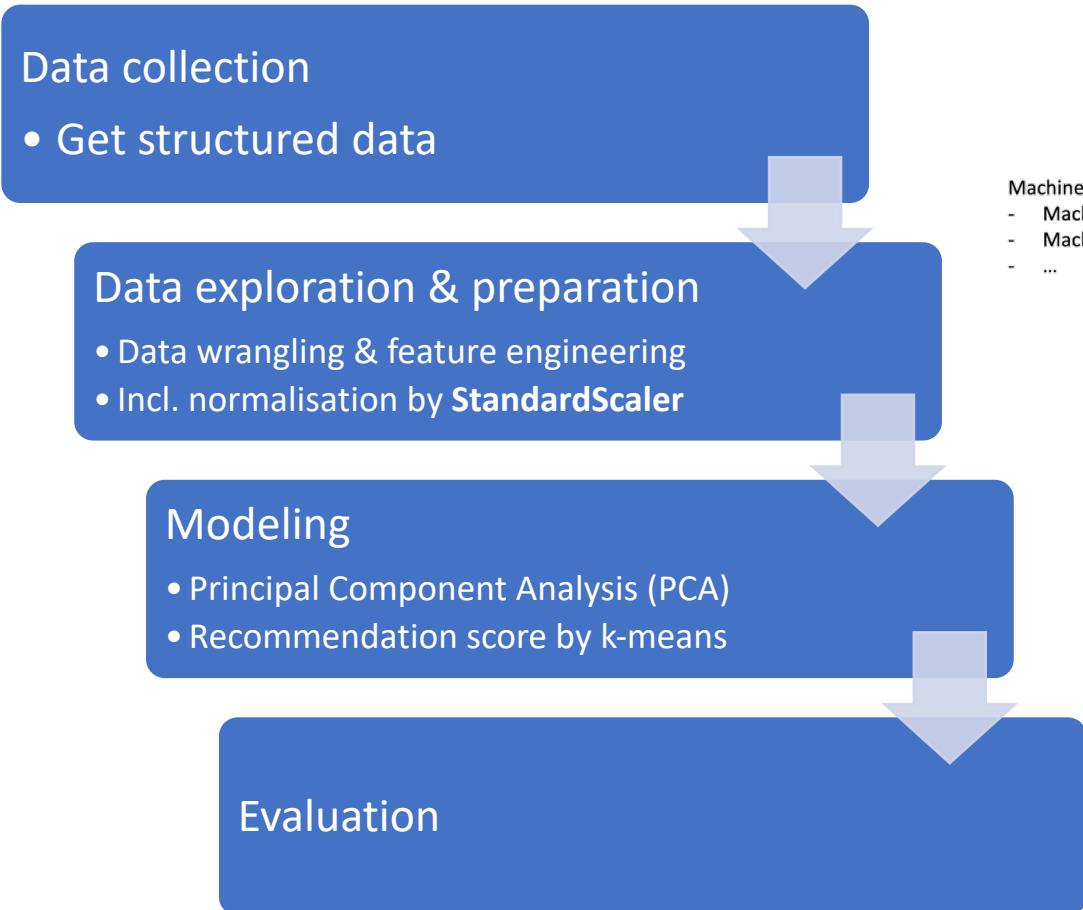
	USER	COURSE_ID	SCORE
0	37465	DS0101EN	1.000000
1	37465	BD0111EN	1.000000
2	37465	BD0101EN	1.000000
3	37465	BD0211EN	1.000000
4	37465	SC0101EN	1.000000
...	...	...	...
20728	2087663	excourse28	0.623754
20729	2087663	BD0145EN	0.623544
20730	2087663	excourse68	0.616759
20731	2087663	excourse60	0.615568
20732	2087663	excourse25	0.600535

# Evaluation results of course similarity based recommender system

The top-10 commonly recommended courses across all test users.

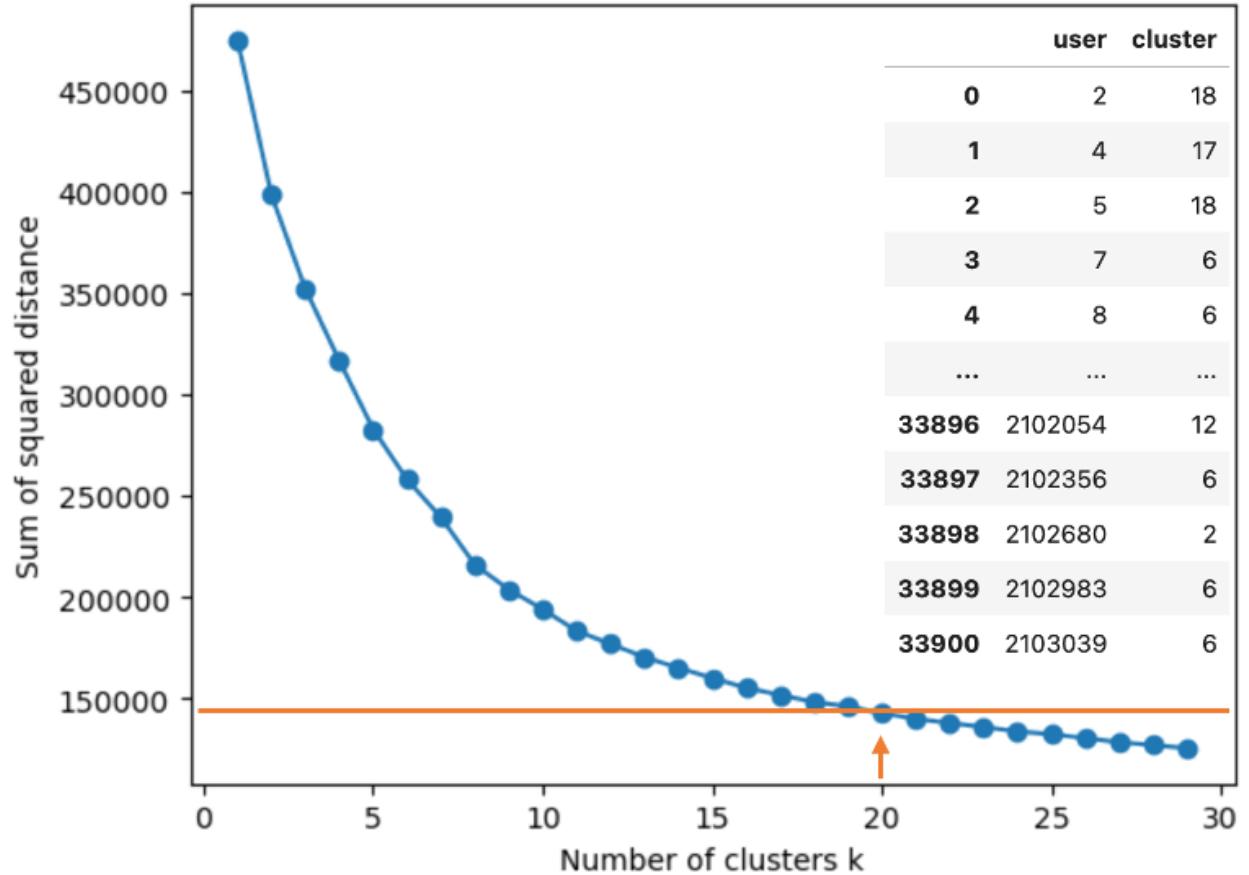
	TITLE	COURSE_ID	SCORE
0	python for data science	PY0101EN	579.000000
1	introduction to data science	DS0101EN	553.931764
2	big data 101	BD0101EN	546.482523
3	data science with open data	DS0110EN	429.676691
4	hadoop 101	BD0111EN	420.000000
5	a crash course in data science	excouse63	385.920850
6	introduction to big data	excouse67	381.727268
7	introduction to data science in python	excouse22	374.903425
8	introduction to data science in python	excouse62	374.903425
9	foundations for big data analysis with sql	excouse72	361.846936

# Flowchart of clustering-based recommender system



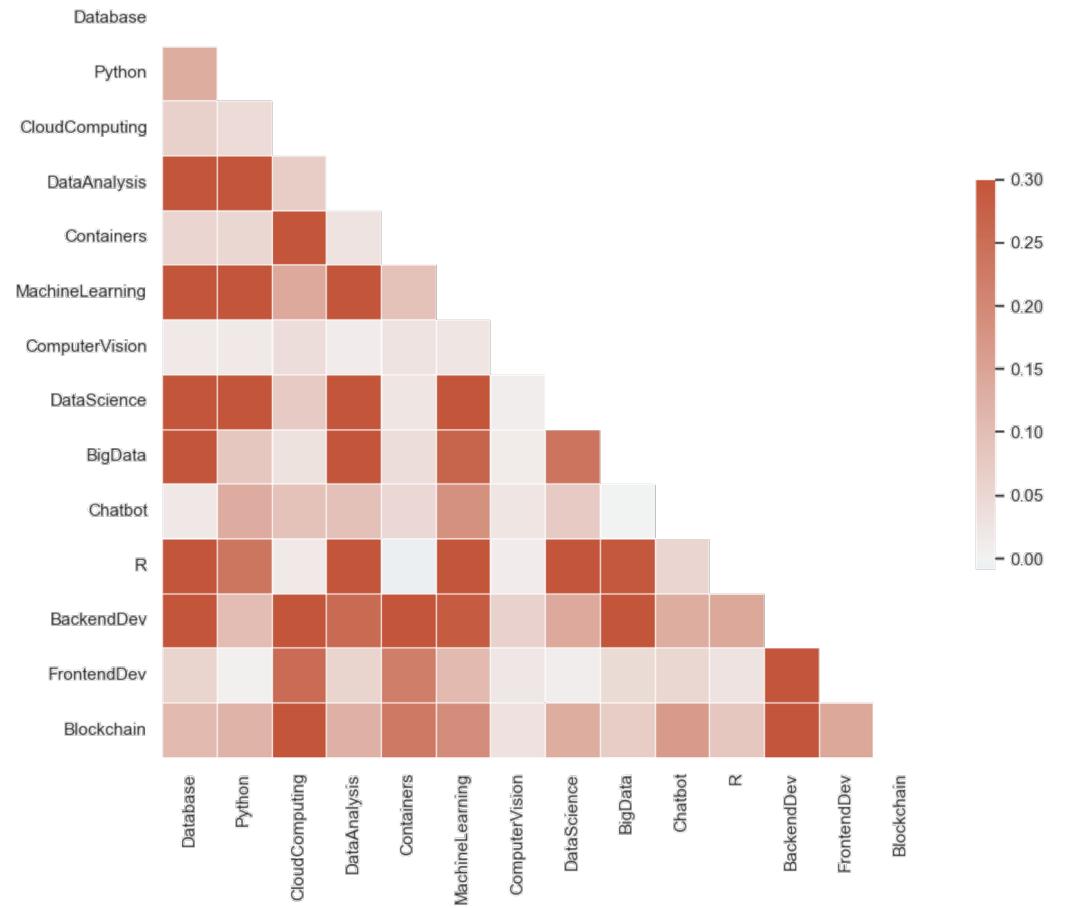
# Modeling

- On the right, an elbow plot after grid search to optimize the KMeans hyperparameter:  
`n_cluster`
- A value of twenty seems to be a good choice to train our model.



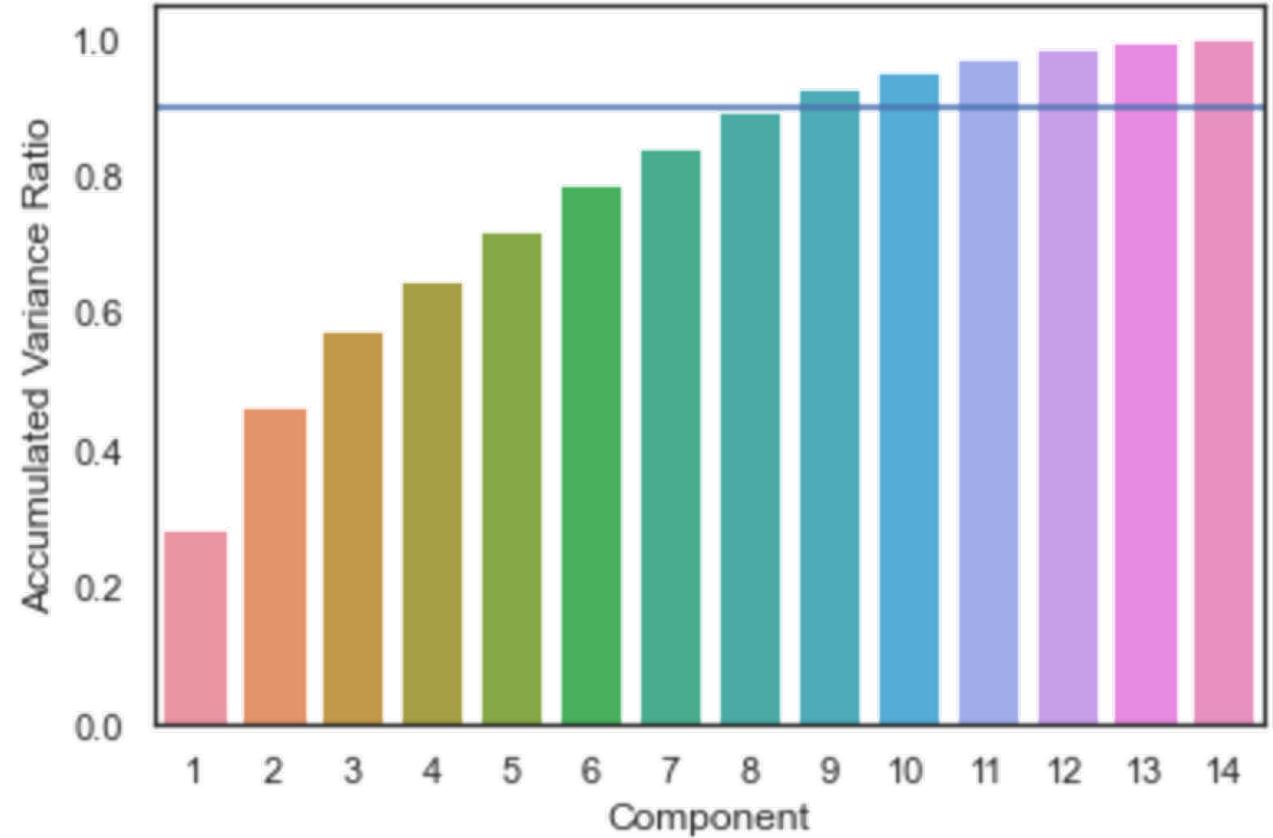
# Principal Component Analysis (PCA)

- A covariance matrix of the user profile feature vectors with 14 features to recognize which features are correlated.
- We can use PCA technique to reduce dimensionality at our 14 features to keep only only the independent main components.



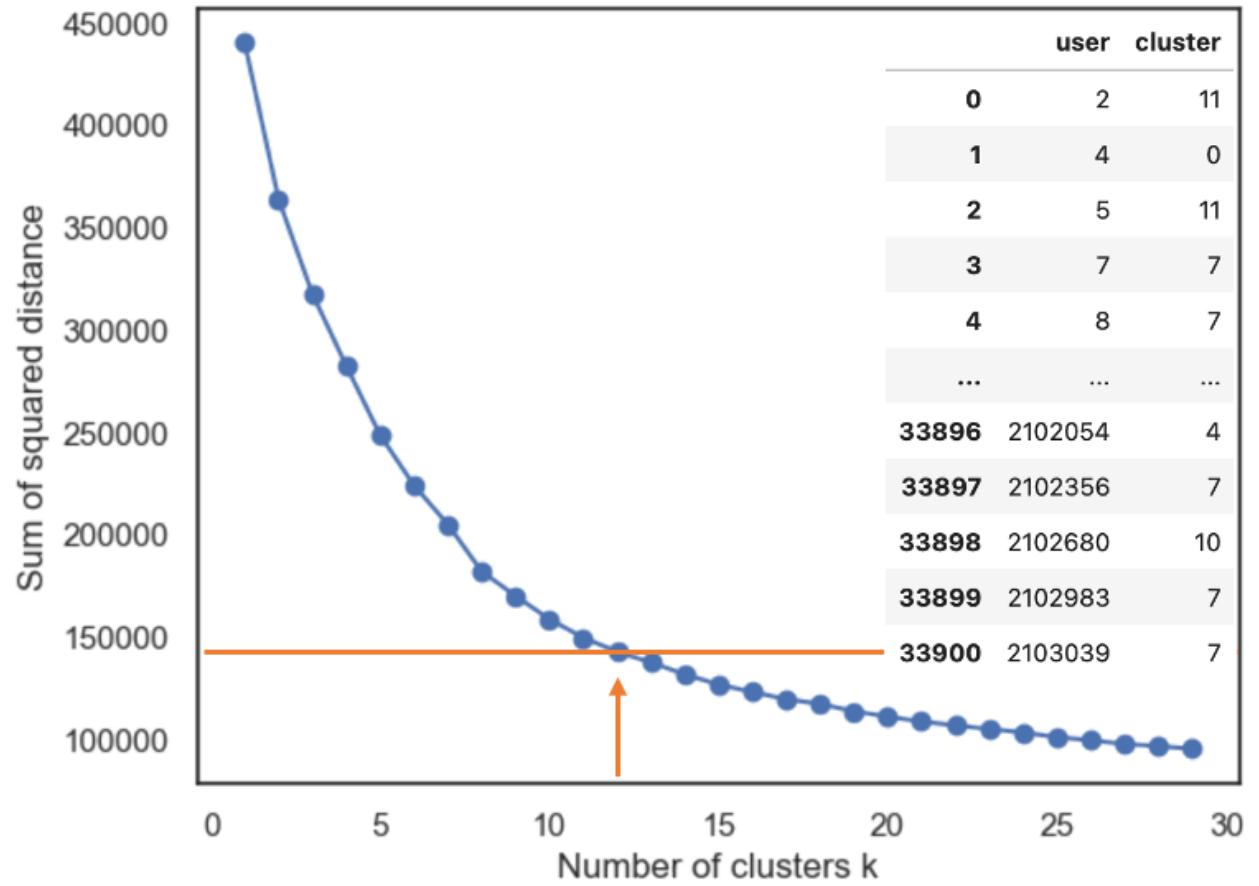
# Principal Component Analysis (PCA)

- On the right, a visualization after grid search to optimize the PCA hyperparameter:  
n\_components
- We see that eight features are below the accumulated variance ratio threshold of 90%.
- Hence, we can reduce the data's dimensionality to eight features.



# Modeling by PCA

- On the right, an elbow plot after grid search on the PCA transformed features to optimize the KMeans hyperparameter:  
n\_cluster
- PCA enabled us to reduce the value of the same sum square error (SSE) from twenty clusters to twelve clusters.



# Evaluation results of clustering-based recommender system

Using a `score_threshold = 10`,  
the average new/unseen courses  
recommended per student (in percentage) is

```
len(res_df) / len(res_df['USER'].unique())
```

```
22.21181262729124
```

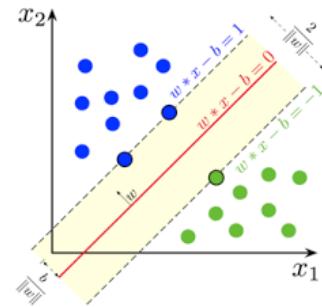
	USER	COURSE_ID
0	1502801	PY0101EN
1	1502801	DS0105EN
2	1502801	DS0103EN
3	1502801	DS0101EN
4	1502801	BD0143EN
...	...	...
11553	630511	ST0101EN
11554	630511	ML0103EN
11555	630511	WA0101EN
11556	630511	DA0101EN
11557	630511	ML0101ENv3

# Evaluation results of clustering-based recommender system

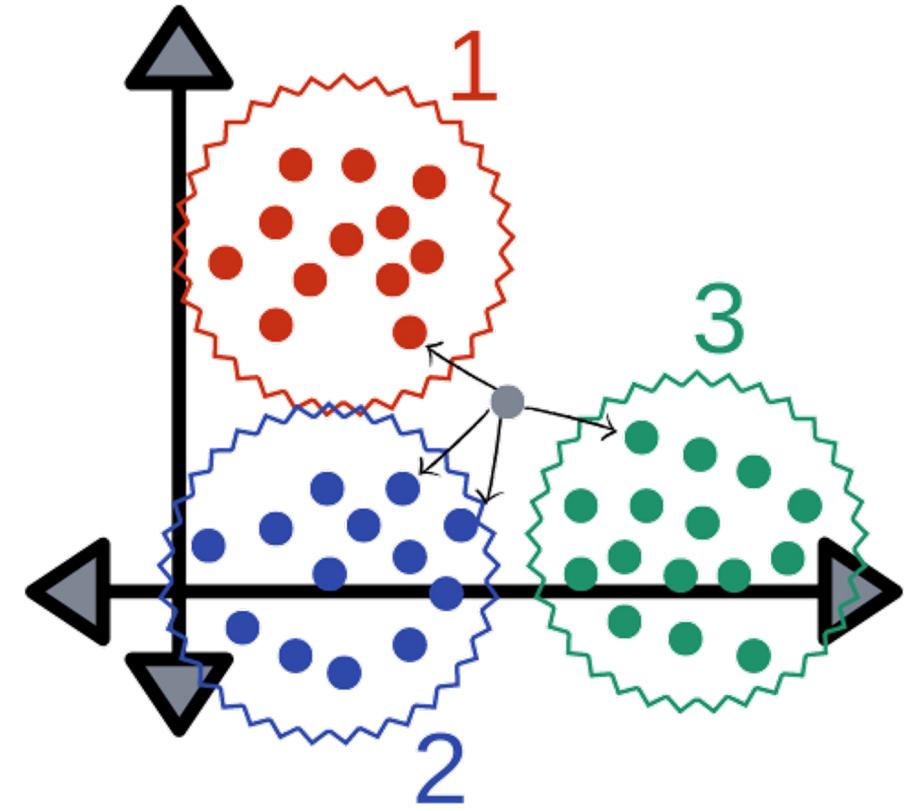
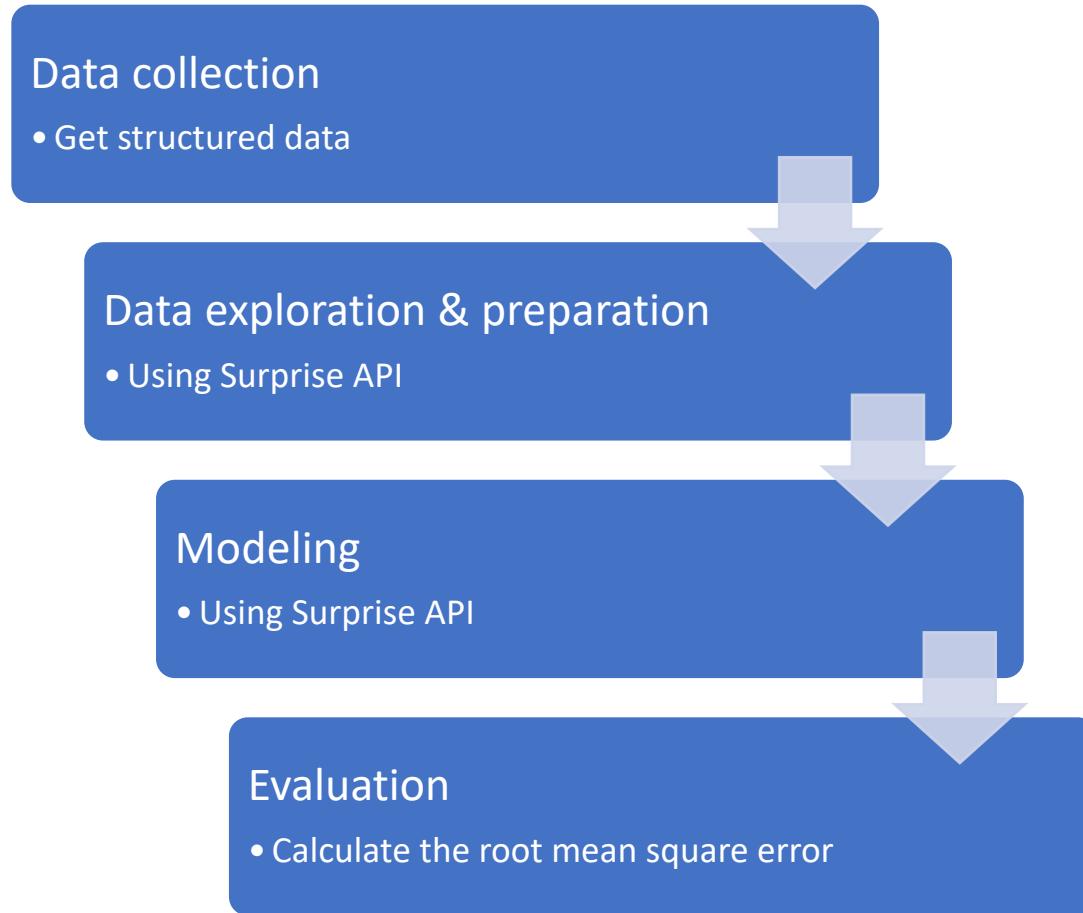
The top-10 commonly recommended courses across all test users.

	TITLE	COURSE_ID	USER
0	statistics 101	ST0101EN	760
1	sql and relational databases 101	DB0101EN	717
2	watson analytics 101	WA0101EN	703
3	deep learning 101	ML0115EN	694
4	docker essentials a developer introduction	CO0101EN	682
5	spark fundamentals i	BD0211EN	671
6	data science methodology	DS0103EN	660
7	data analysis with python	DA0101EN	644
8	ibm cloud essentials	CL0101EN	625
9	blockchain essentials	BC0101EN	621

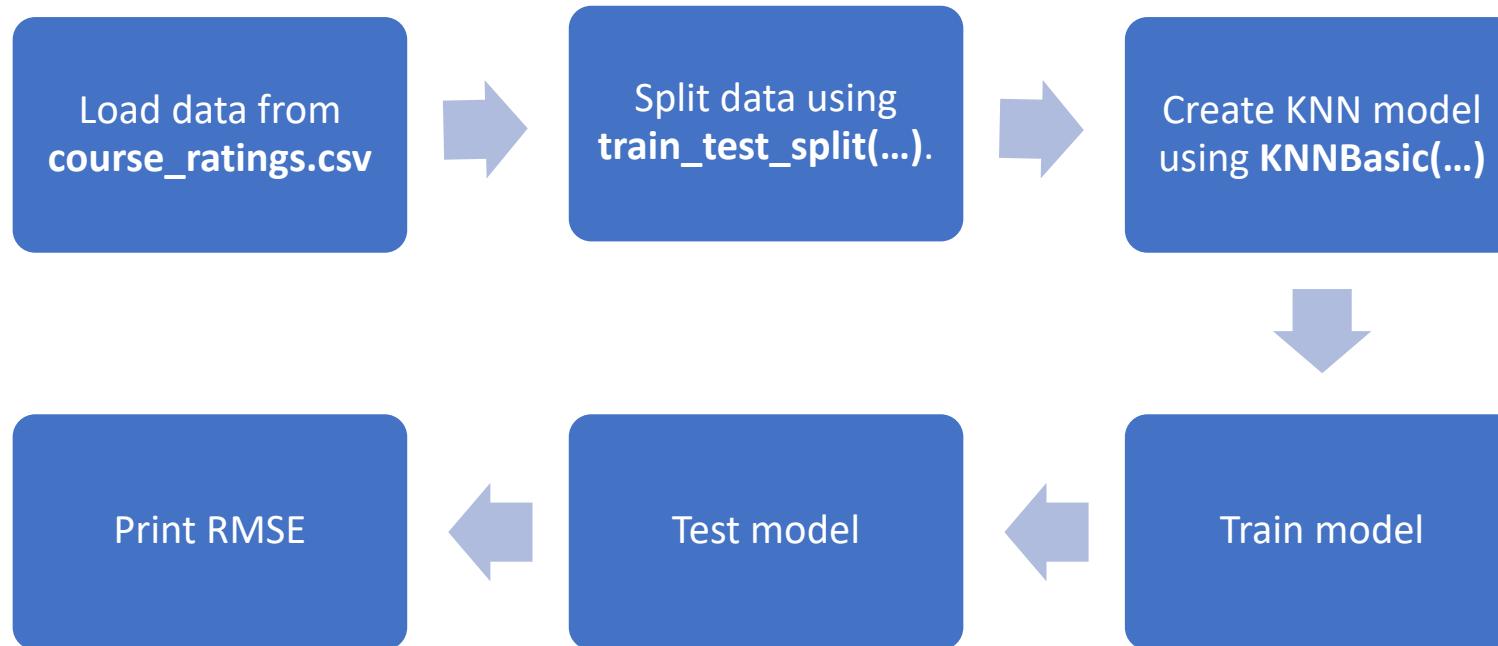
# Collaborative-filtering Recommender System using Supervised Learning



# Flowchart of KNN based recommender system



# Surprise API



	user	item	rating
0	1889878	CC0101EN	3.0
1	1342067	CL0101EN	3.0
2	1990814	ML0120ENv3	3.0
3	380098	BD0211EN	3.0
4	779563	DS0101EN	3.0

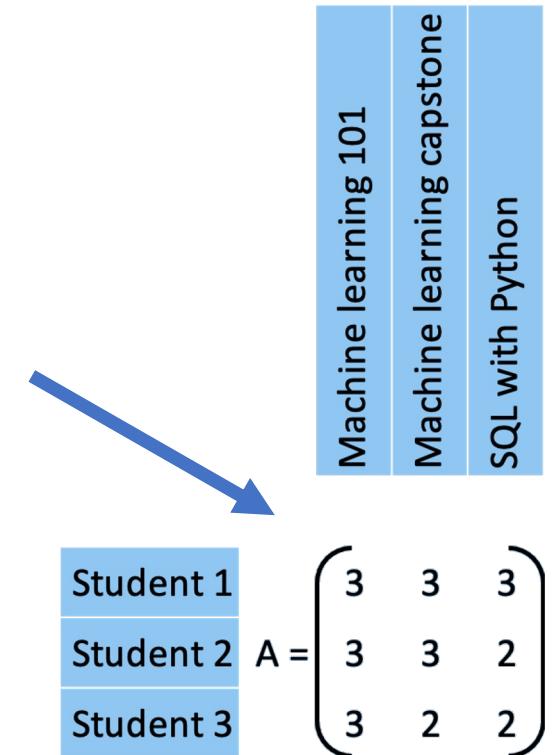
KNN Hyperparameters:

- `min_k = 3`
- `min_k = 12`
- `cosine` similarity

# Sparse matrix

Notice that the Surprise API does the transformaton job for our us. On the right, you see an example how the data is transformed from a dense matrix to a sparse matrix.

user	item	rating
Student 1	Machine learning 101	3
Student 1	Machine learning capstone	3
Student 1	SQL with Python	3
Student 2	Machine learning 101	3
Student 2	Machine learning capstone	3
Student 2	SQL with Python	2
Student 3	Machine learning 101	3
Student 3	Machine learning capstone	2
Student 3	SQL with Python	2



# Evaluation results of KNN based recommender system

The result of the user-based collaborative filtering is a root mean square error (RMSE) of 0.2042. The predicted rating of user  $u$  to item  $i$  is given by the formula

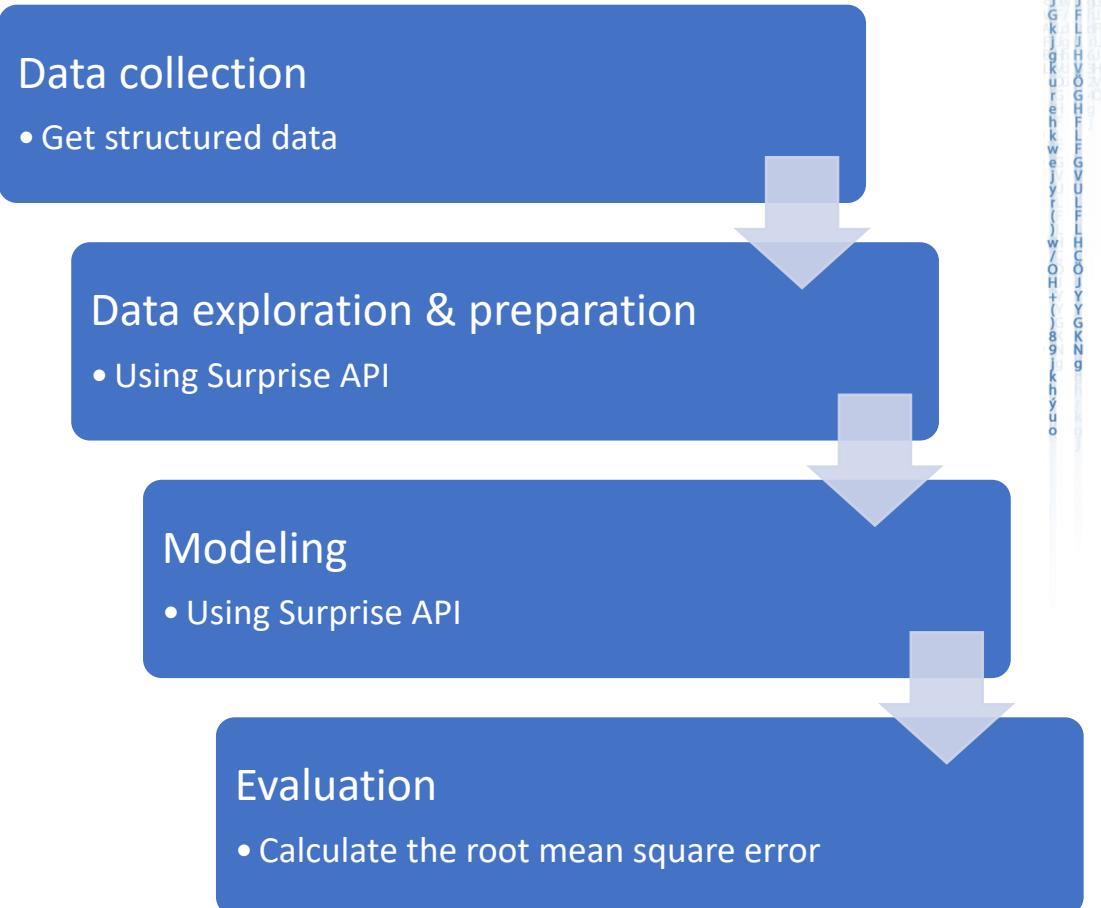
$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} \text{similarity}(u, v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} \text{similarity}(u, v)}$$

The result of item-based collaborative filtering is a root mean square error (RMSE) of 0.2048. The predicted rating of user  $u$  to item  $i$  is given by the formula

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

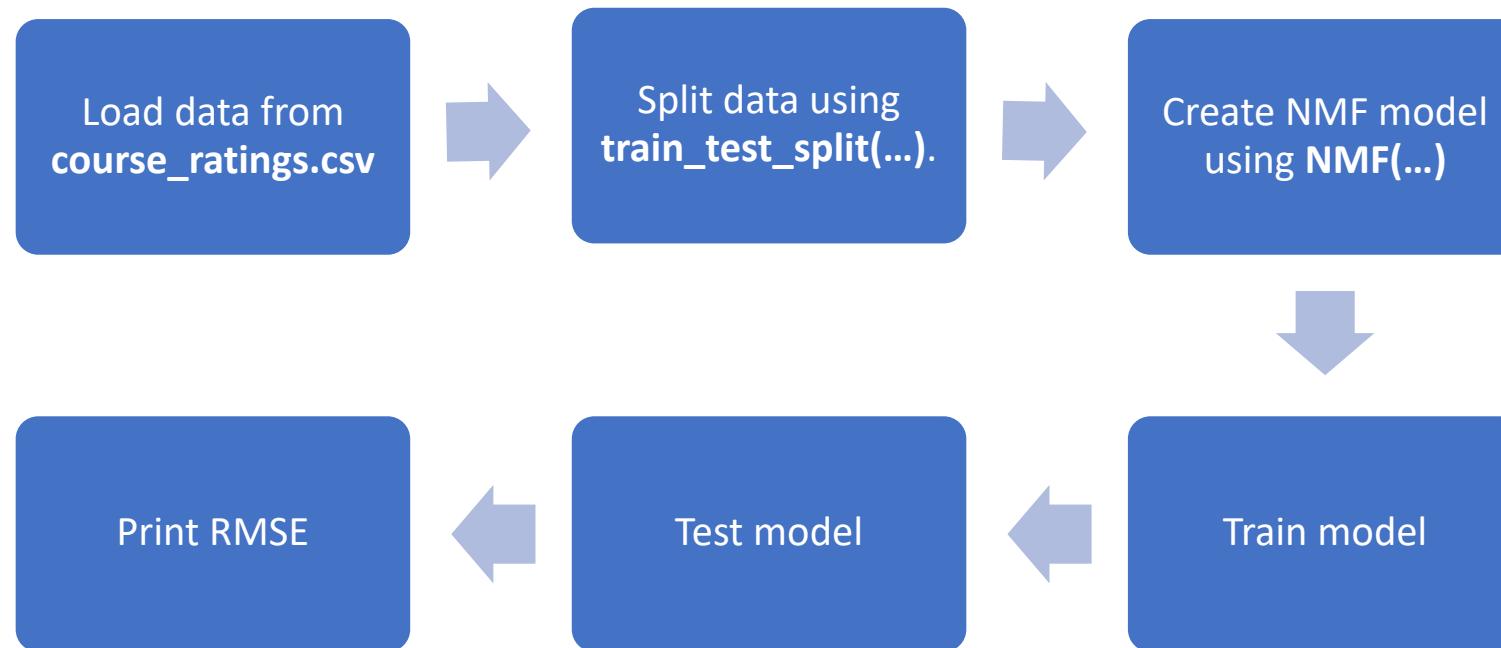
The user-based collaborative filtering has a little better accuracy.

# Flowchart of NMF based recommender system



A large, semi-transparent watermark or background image is visible across the entire slide. It features a dense grid of letters and numbers in various sizes and orientations, resembling a barcode or a complex matrix. The letters are primarily in a monospace font, with some variations in style and color. The overall effect is a technical or data-oriented aesthetic.

# Surprise API



	user	item	rating
0	1889878	CC0101EN	3.0
1	1342067	CL0101EN	3.0
2	1990814	ML0120ENv3	3.0
3	380098	BD0211EN	3.0
4	779563	DS0101EN	3.0

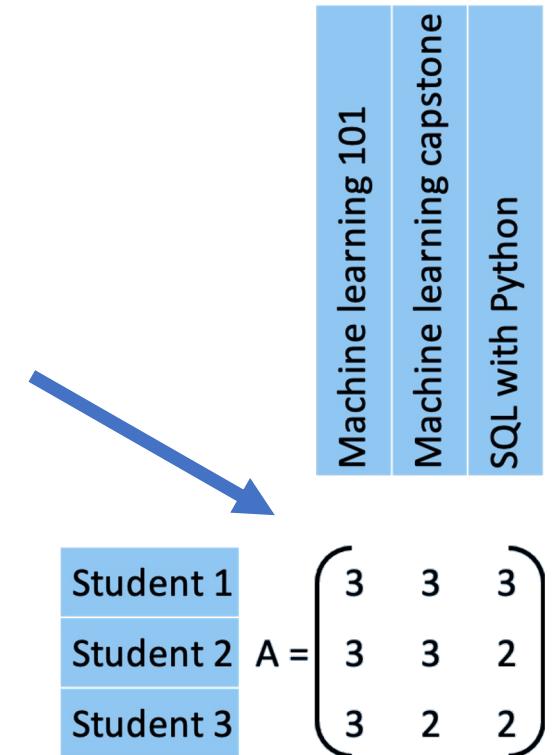
## NMF Hyperparameters:

- `init_low = ½`
- `init_high= 5`
- `factors = 32`

# Sparse matrix

Notice that the Surprise API does the transformaton job for our us. On the right, you see an example how the data is transformed from a dense matrix to a sparse matrix.

user	item	rating
Student 1	Machine learning 101	3
Student 1	Machine learning capstone	3
Student 1	SQL with Python	3
Student 2	Machine learning 101	3
Student 2	Machine learning capstone	3
Student 2	SQL with Python	2
Student 3	Machine learning 101	3
Student 3	Machine learning capstone	2
Student 3	SQL with Python	2



# Non-negative matrix factorization

For dimensionality reduction, the sparse matrix  $A$  is decomposed into two smaller and dense matrices,  $B$  and  $C$ .

Machine learning 101	Machine learning capstone	SQL with Python
----------------------	---------------------------	-----------------

Feature 1	Feature 2	Feature 3
-----------	-----------	-----------

Machine learning 101	Machine learning capstone	SQL with Python
----------------------	---------------------------	-----------------

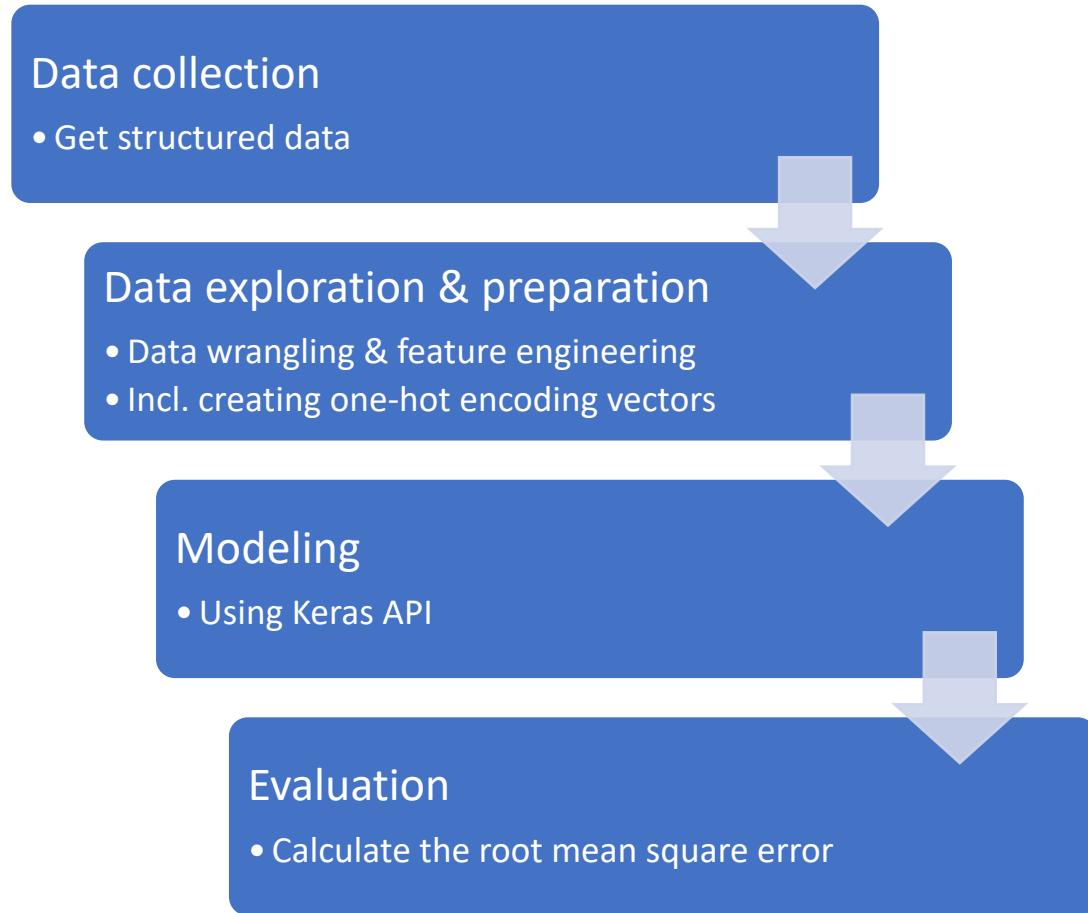
$$\begin{array}{c|ccc} \text{Student 1} & 3 & 3 & 3 \\ \text{Student 2} & 3 & 3 & 2 \\ \text{Student 3} & 3 & 2 & 2 \end{array} A = \begin{pmatrix} 3 & 3 & 3 \\ 3 & 3 & 2 \\ 3 & 2 & 2 \end{pmatrix} = \begin{array}{c|ccc} \text{Student 1} & f & f & f \\ \text{Student 2} & f & f & f \\ \text{Student 3} & f & f & f \end{array} B = \begin{pmatrix} f & f & f \\ f & f & f \\ f & f & f \end{pmatrix} \cdot \begin{array}{c|ccc} \text{Feature 1} & f & f & f \\ \text{Feature 2} & f & f & f \\ \text{Feature 3} & f & f & f \end{array} C = \begin{pmatrix} f & f & f \\ f & f & f \\ f & f & f \end{pmatrix}$$

# Evaluation results of NMF based recommender system

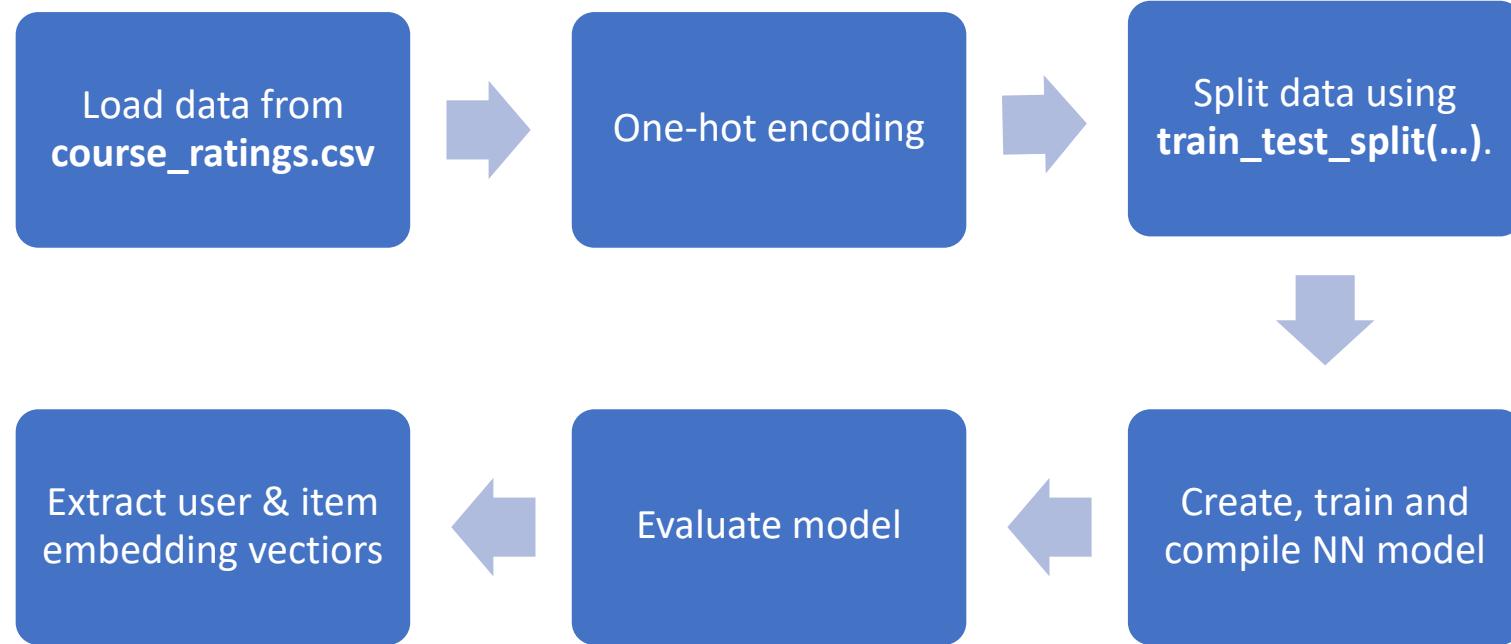
The result of the NMF-based collaborative filtering  
is a root mean square error (RMSE) of 0.1912.

The accuracy is a little better than the KNN models.

# Flowchart of Neural Network Embedding based recommender system



# Details



	user	item	rating
0	1889878	CC0101EN	3.0
1	1342067	CL0101EN	3.0
2	1990814	ML0120ENv3	3.0
3	380098	BD0211EN	3.0
4	779563	DS0101EN	3.0

# Modeling

Our neuronal network (NN) model is build by the customized function:

```
class RecommenderNet(keras.Model):

    def __init__(self, num_users, num_items, embedding_size=16, **kwargs):
        """
        Constructor
        :param int num_users: number of users
        :param int num_items: number of items
        :param int embedding_size: the size of embedding vector
        """
        super(RecommenderNet, self).__init__(**kwargs)
        self.num_users = num_users
        self.num_items = num_items
        self.embedding_size = embedding_size

        # Define a user_embedding vector
        # Input dimension is the num_users
        # Output dimension is the embedding size
        self.user_embedding_layer = layers.Embedding(
            input_dim=num_users,
            output_dim=embedding_size,
            name='user_embedding_layer',
            embeddings_initializer="he_normal",
            embeddings_regularizer=keras.regularizers.l2(1e-6),
        )
        # Define a user_bias layer
        self.user_bias = layers.Embedding(
            input_dim=num_users,
            output_dim=1,
            name="user_bias")

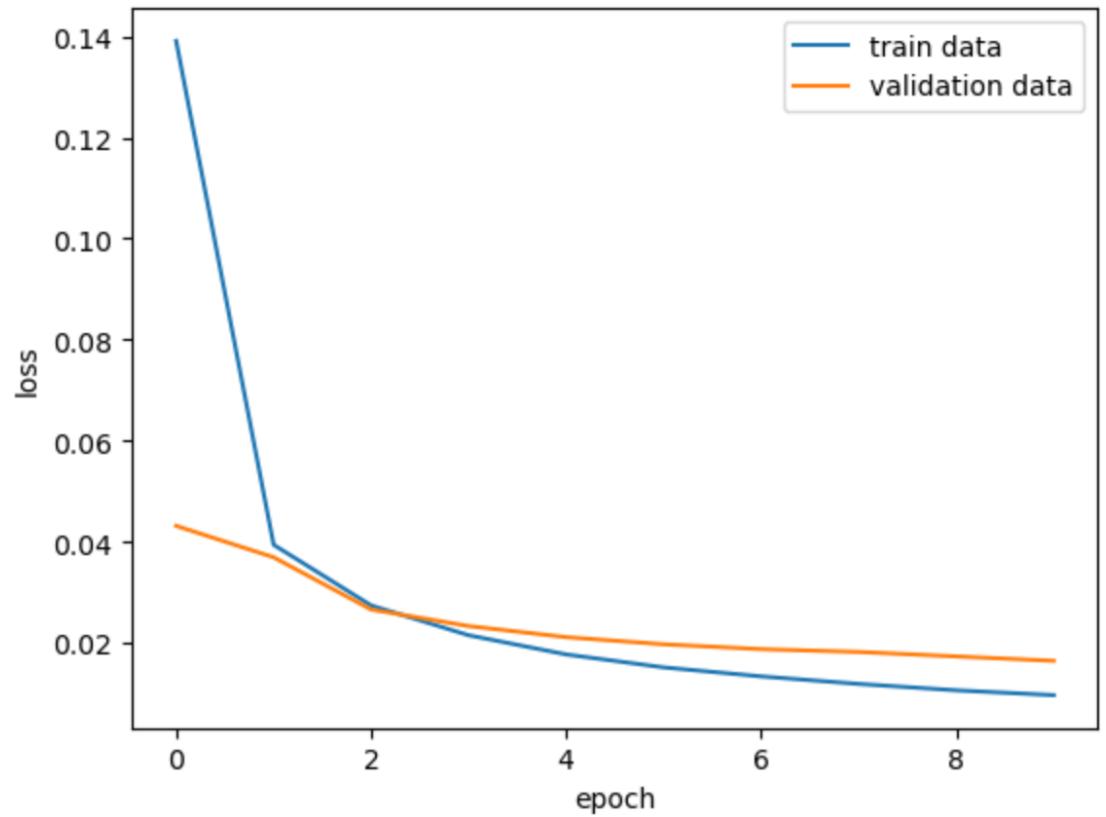
    def call(self, inputs):
        """
        method to be called during model fitting
        :param inputs: user and item one-hot vectors
        """
        # Compute the user embedding vector
        user_vector = self.user_embedding_layer(inputs[:, 0])
        user_bias = self.user_bias(inputs[:, 0])
        item_vector = self.item_embedding_layer(inputs[:, 1])
        item_bias = self.item_bias(inputs[:, 1])
        dot_user_item = tf.tensordot(user_vector, item_vector, 2)
        # Add all the components (including bias)
        x = dot_user_item + user_bias + item_bias
        # Sigmoid output layer to output the probability
        return tf.nn.relu(x)
```

The compile parameters are

- Optimizer = Adam
- loss = MeanSquaredError
- metrics = RootMeanSquaredError

# Modeling training

- At the first epochs, the validation data optimized better than the train data.
- After epoch two, the trainset outperformed.
- Finally, the spread between the sets kept stable but the optimization rate decelerated.



# Evaluation results of Neural Network Embedding based recommender system

At the end, we evaluated our model on the testset and found:

- loss is 0.0159
- RMSE is 0.1190

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

$$RMSE = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}}$$

# Extract user & item embedding vectors

A summary of the model gives us the number of parameters on

- `user_embedding_layer`
- `item_embedding_layer`

```
model.summary()
```

```
Model: "recommender_net"
```

Layer (type)	Output Shape	Param #
<hr/>		
user_embedding_layer (Embedding)	multiple	542416
user_bias (Embedding)	multiple	33901
item_embedding_layer (Embedding)	multiple	2016
item_bias (Embedding)	multiple	126
<hr/>		
Total params: 578,459		
Trainable params: 578,459		
Non-trainable params: 0		

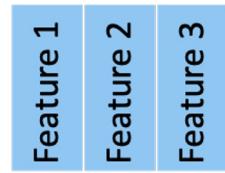
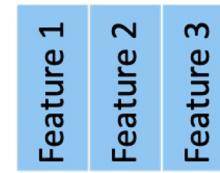
# Extract user & item embedding vectors

Furthermore, the function `get_weights(...)` gives us the trained weights on both layers:

- user features shape: (33901,16)
- Item features shape: (126,16)

Each user or item has been transformed into a 16x1 latent feature vector used for prediction.

Illustration of two latent feature vectors of the size 3x1:

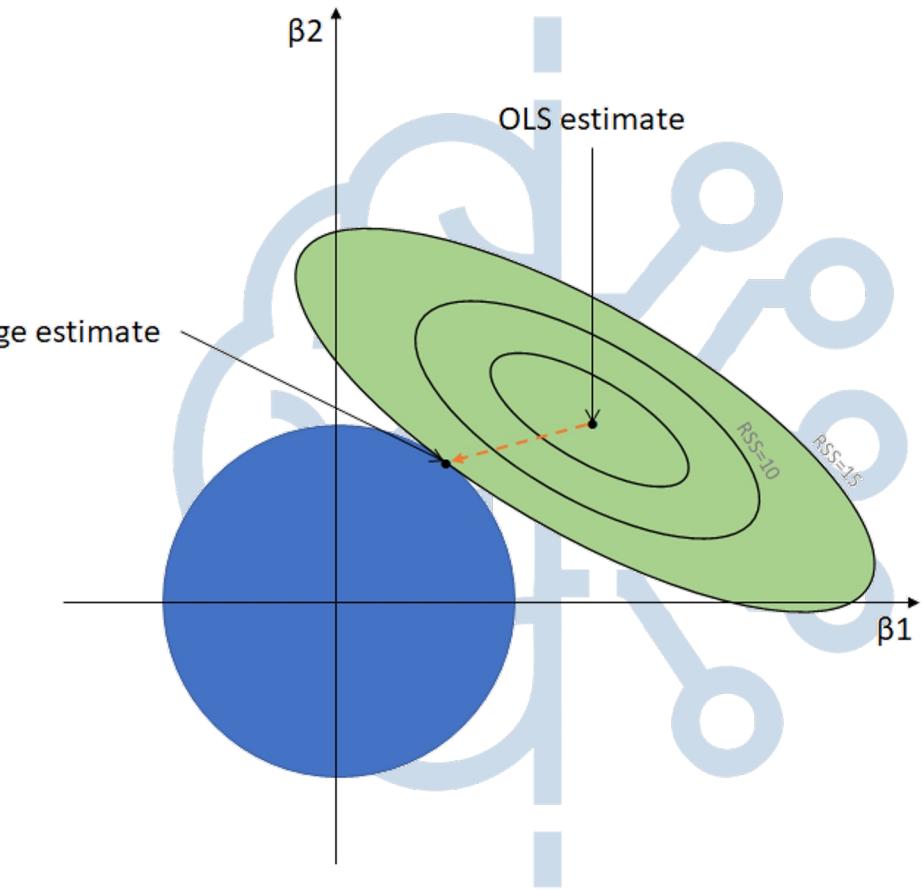
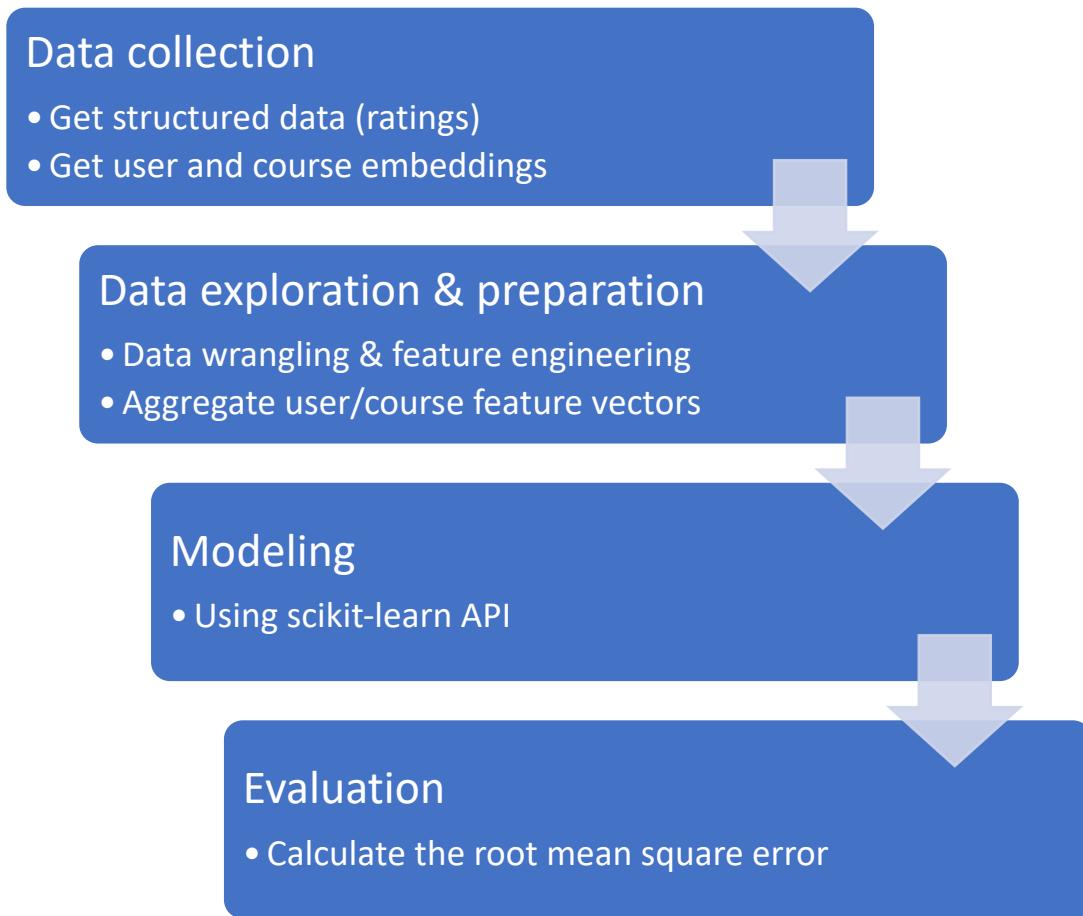


$$\text{Student 1 } A = \begin{pmatrix} f & f & f \end{pmatrix}$$

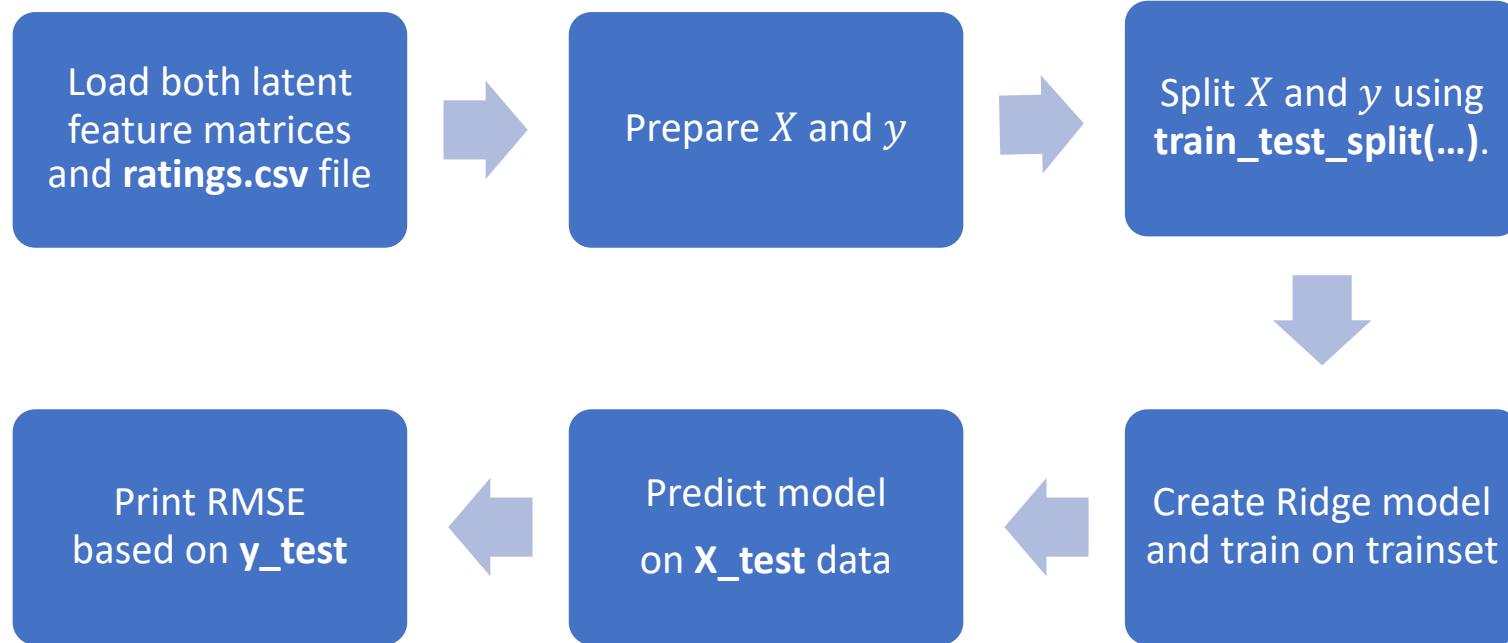
$$\text{Item 1 } B = \begin{pmatrix} f & f & f \end{pmatrix}$$

Next, we will use both latent feature matrices for prediction.

# Flowchart of Regression Embedding based recommender system



# Details



	user	item	rating
0	1889878	CC0101EN	3.0
1	1342067	CL0101EN	3.0
2	1990814	ML0120ENv3	3.0
3	380098	BD0211EN	3.0
4	779563	DS0101EN	3.0

# Prepare $X$ and $y$

We use the `ratings.csv` table and merge with both latent feature matrices,  $A$  and  $B$ .

	user	item	rating		Feature 1	Feature 2	Feature 3		Feature 1	Feature 2	Feature 3
0	1889878	CC0101EN	3.0	Student 1	$f$	$f$	$f$	Item 1	$f$	$f$	$f$
1	1342067	CL0101EN	3.0	Student 2	$f$	$f$	$f$	Item 2	$f$	$f$	$f$
2	1990814	ML0120ENv3	3.0	Student 3	$f$	$f$	$f$	Item 3	$f$	$f$	$f$

$A = \begin{pmatrix} f & f & f \\ f & f & f \\ f & f & f \end{pmatrix}$

$B = \begin{pmatrix} f & f & f \\ f & f & f \\ f & f & f \end{pmatrix}$

# Extract user & item embedding vectors

Next, we assign the sum of both latent feature matrices, joined in the merged dataset, to  $X$ .

Student 1 / Item 1	Feature 1	Feature 2	Feature 3
Student 2 / Item 2	f	f	f
Student 3 / Item 3	f	f	f

$$A + B = \begin{pmatrix} f & f & f \\ f & f & f \\ f & f & f \end{pmatrix} = X$$

Finally, we assign the rating column from merged dataset to  $y$ .

Feature 1	Feature 2	Feature 3
-----------	-----------	-----------

$$\text{Rating } Y = \begin{pmatrix} f & f & f \end{pmatrix}$$

The  $X$  and  $y$  matrices can now be splitted to a trainset and a testset.

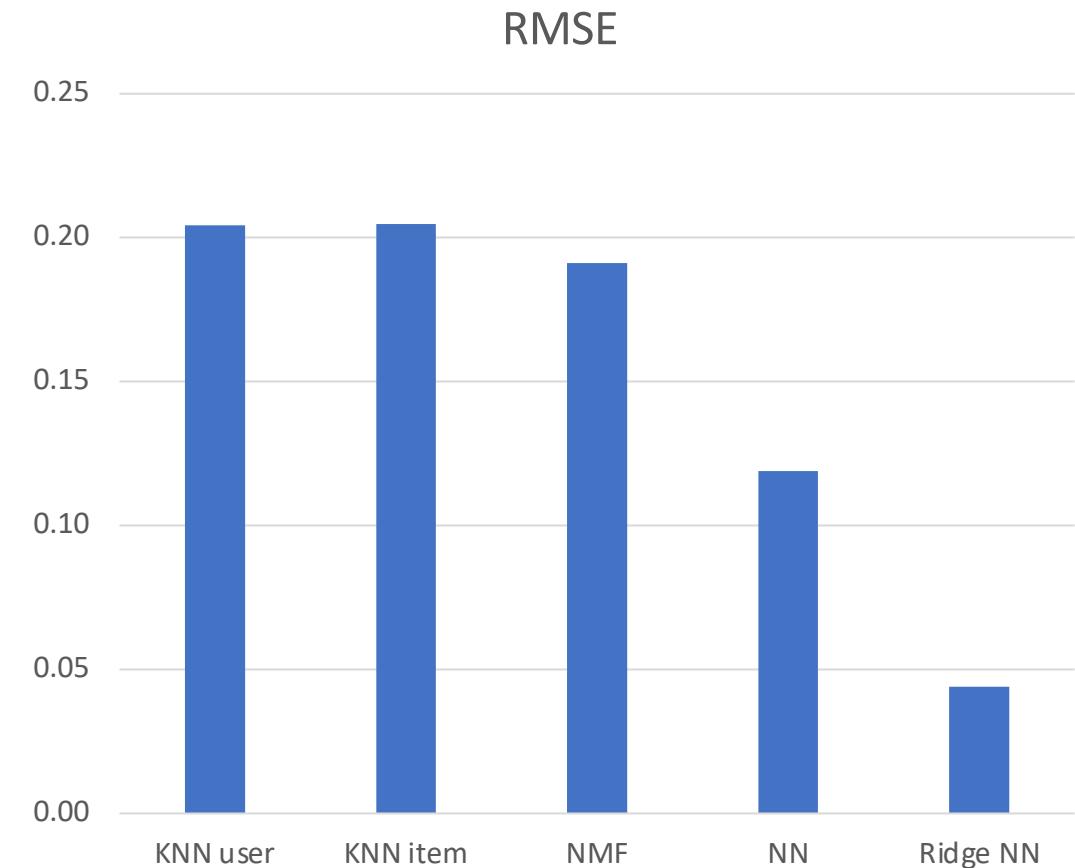
# Evaluation results of Regression Embedding based recommender system

After train and prediction,  
we found that our regression model based on Ridge regression,  
using latent feature matrices trained by neural network,  
has a root mean square error (RMSE) of 0.0440.



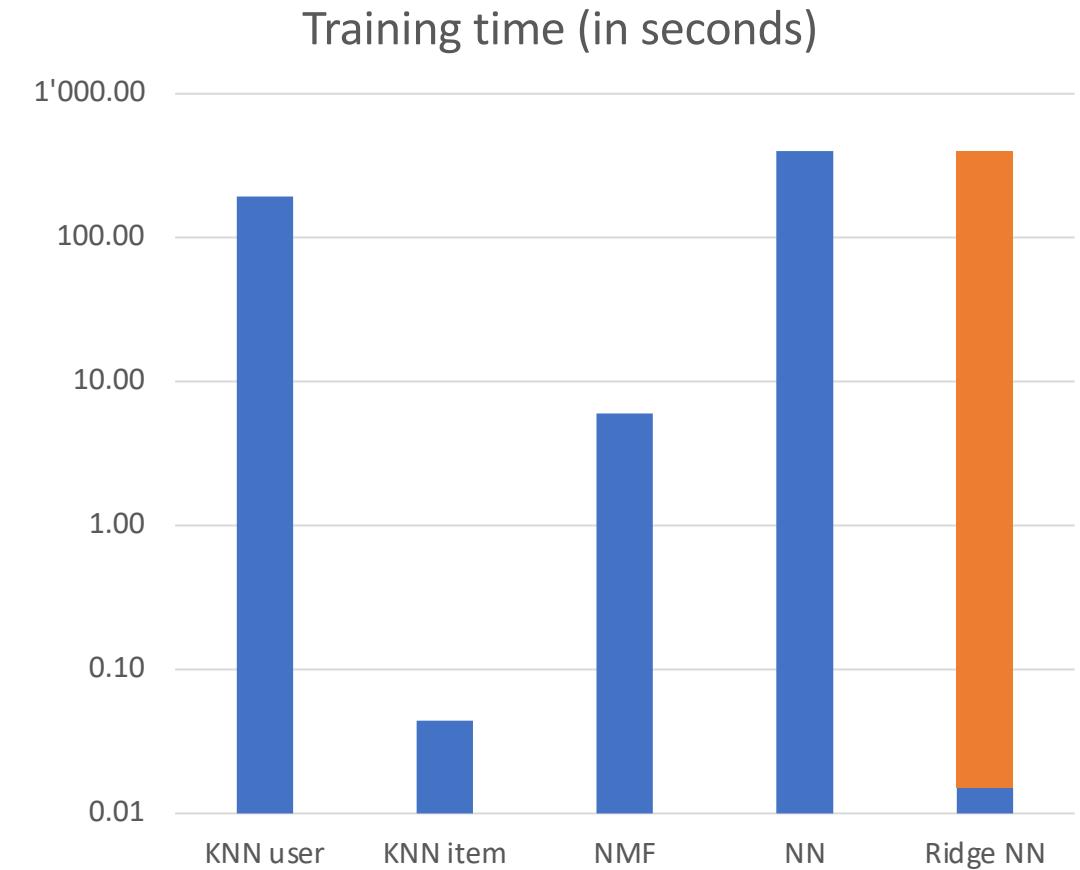
# Compare the performance of collaborative-filtering models

Model	RMSE
KNN user-based	0.2042
KNN item-based	0.2048
Non-negative Matrix Factorization (NMF)	0.1912
Neuronal network (NN)	0.1190
<b>Ridge using embedding features</b>	<b>0.0440</b>



# Compare the performance of collaborative-filtering models

Model	Seconds
KNN user-based	192.926
KNN item-based	0.044
Non-negative Matrix Factorization (NMF)	5.986
Neuronal network (NN)	399.978
Ridge using embedding features	0.015 +399.978



# Build a course recommender system app with Streamlit

Personalized Learning Recommender

1. Select recommendation models  
Select model:  
Course Similarity

2. Tune Hyper-parameters:  
Top courses  
3  
0 100  
Course Similarity Threshold %  
60  
0 100

3. Training:  
Train Model

4. Prediction  
Recommend New Courses

Datasets loaded successfully...

Select courses that you have audited or completed:

COURSE_ID	TITLE ↑	DESCRIPTION
excuse69	Machine Learning With Big Data	want to make sens
ML0101EN	Machine Learning With Python	are the phrases , it
ML0101ENV3	Machine Learning With Python	machine learning c
ML0151EN	Machine Learning With R	this machine learni
GPXX0G3KEN	Managing And Injecting Dependencies Into Java Microservices Using Contexts And Depen...	learn how to use cr
BD0015EN	Mapreduce And Yarn	string together you
CP0101EN	Mathematical Optimization For Business Problems	mathematical optimi
GPXX05RDEN	Medical Appointment Data Analysis	in this notebook we
EE0101EN	Modernizing Java Ee Applications	learn about the tec
GPXXQOS6EN	Monitoring The Metrics Of Java Microservices Using Eclipse Microprofile Metrics	you will explore ho
BD00131EN	Moving Data Into Hadoop	this course descri
excuse77	Natural Language Processing With Attention Models	in course 4 of the r
excuse78	Natural Language Processing With Probabilistic Models	in course 4 of the r

Your courses:

COURSE_ID	TITLE
0	ML0101EN
1	ML0101ENV3

Recommendations generated!

SCORE	TITLE	DESCRIPTION
0 0.6626	Machine Learning With R	this machine learning with r course dives into the basics of machine learning using an approachable and well known programming language you ll learn about supervised vs unsupervised learning look into how statistical modeling relates to machine learning and do a comparison of each
1 0.6348	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 stories that herald new breakthroughs in facial recognition technology self driving cars or computers that can have a conversation just like a real person machine learning technology is set to revolutionise almost any area of human life and work and so will affect all our lives and so you are likely to want to find out more about it machine learning has a reputation for being one of the most complex areas of computer science requiring advanced mathematics and engineering skills to understand it while it is true that working as a machine learning engineer does involve a lot of mathematics and programming we believe that anyone can understand the basic concepts of machine learning and given the importance of this technology everyone should the big ai breakthroughs sound like science fiction but they come down to a simple idea the use of data to train statistical algorithms in this course you will learn to understand the basic idea of machine learning even if you don t have any background in math or programming not only that you will get hands on and use user friendly tools developed at goldsmiths university of london to actually do a machine learning project training a computer to recognise images this course is for a lot of different people it could be a good first step into a technical career in machine learning after all it is always better to start with the high level concepts before the technical details but it is also great if your role is non technical you might be a manager or other non technical role in a company that is considering using machine learning you really need to understand this technology and this course is a great place to get that understanding or you might just be following the news reports about ai and interested in finding out more about the hottest new technology of the moment whoever you are we are looking forward to guiding you through you first machine learning project nb this course is designed to introduce you to machine learning without needing any programming that means that we don t cover the programming based machine learning tools like python and tensorflow introduction to machine learning this course will provide you a foundational understanding of machine learning models logistic regression multilayer perceptrons convolutional neural networks natural language processing etc as well as demonstrate how these models can solve complex problems in a variety of industries from medical diagnostics to image recognition to text prediction in addition we have designed practice exercises that will give you hands on experience implementing these data science models on data sets these practice exercises will teach you how to implement machine learning algorithms with pytorch open source libraries used by leading tech companies in the machine learning field e g google nvidia cocacola ebay snapchat uber and many more
2 0.6121	Machine Learning	machine learning is the science of getting computers to act without being explicitly programmed in the past decade machine learning has given us self driving cars practical speech recognition effective web search and a vastly improved understanding of the human genome machine learning is so pervasive today that you probably use it dozens of times a day without knowing it many researchers also think it is the best way to make progress towards human level ai in this class you will learn about the most effective machine learning techniques and gain practice implementing them and getting them to work for yourself more importantly you ll learn about not only the theoretical underpinnings of learning but also gain the practical know how needed to quickly and powerfully apply these techniques to new problems finally you ll learn about some of silicon valley s best practices in innovation as it pertains to machine learning and ai this course provides a broad introduction to machine learning datamining and statistical pattern recognition topics include i supervised learning parametric non parametric algorithms support vector machines kernels neural networks ii unsupervised learning clustering dimensionality reduction recommender systems deep learning iii best practices in machine learning bias variance theory innovation process in machine learning and at the course will also draw from numerous case studies and applications so that you ll also learn how to apply learning algorithms to building smart robots perception control text understanding web search anti spam computer vision medical informatics audio database mining and other areas

```
$ python -m venv $APP_PATH
```

```
$ streamlit run recommender_app.py
```

50

# Conclusions

---

- There is not much significant RMSE differences between KNN and NMF models.
- Time measurement between the non-neuronal networks revealed that the KNN item-based training is the fastest and KNN user-based the slowest.
- Ridge regression using the embedding features, trained by a neuronal network, has the best predictability
- RMSE is 2.7 times lower than the second best (NN) and 4.6 times lower than the worst model (KNN user-based).
- Drawback is the higher computing costs to get the best model. In addition, the workload is higher, especially for neuronal networks.
- A pre-trained neuronal network reduces the computing costs insofar as the Ridge regression becomes the fastest trained model overall with the best predictability.



# Appendix

---

- Python snippets
- Github repository: <https://github.com/starfile/machine-learning-capstone>

# Python snippets

Calculate the top-10 commonly recommended courses across all test users for profile-based recommender system:

```
most_freq_recommendations = res_df.groupby('COURSE_ID').sum() \
    .sort_values('SCORE', ascending = False) \
    .head(10) \
    .reset_index()

filtered_courses = course_genres_df.iloc[
    pd.Index(course_genres_df['COURSE_ID']) \
    .get_indexer(most_freq_recommendations['COURSE_ID'])] \
    .reset_index()

most_freq_recommendations = filtered_courses.join(
    most_freq_recommendations['SCORE'])[['TITLE', 'COURSE_ID', 'SCORE']]

most_freq_recommendations
```

Calculate the top-10 commonly recommended courses across all test users for similarity based recommender system:

```
most_freq_recommendations = res_df.groupby('COURSE_ID').sum() \
    .sort_values('SCORE', ascending = False) \
    .head(10) \
    .reset_index()

filtered_courses = course_df.iloc[
    pd.Index(course_df['COURSE_ID']) \
    .get_indexer(most_freq_recommendations['COURSE_ID'])] \
    .reset_index()

most_freq_recommendations = filtered_courses.join(
    most_freq_recommendations['SCORE'])[['TITLE', 'COURSE_ID', 'SCORE']]

most_freq_recommendations
```

# Python snippets

Calculate the top-10 commonly recommended courses across all test users for clustering-based recommender system:

```
course_url = "https://cf-courses-data.s3.us.cloud-object-storage" + \
    ".appdomain.cloud/IBM-ML321EN-SkillsNetwork/labs/datasets/" + \
    "course_processed.csv"

course_df = pd.read_csv(course_url)

most_freq_recommendations = res_df.groupby('COURSE_ID').count() \
    .sort_values('USER', ascending = False) \
    .head(10) \
    .reset_index()

filtered_courses = course_df.iloc[
    pd.Index(course_df['COURSE_ID']) \
    .get_indexer(most_freq_recommendations['COURSE_ID'])] \
    .reset_index()

most_freq_recommendations = filtered_courses.join(
    most_freq_recommendations['USER'])[[ 'TITLE', 'COURSE_ID', 'USER']]

most_freq_recommendations
```

Thank you!