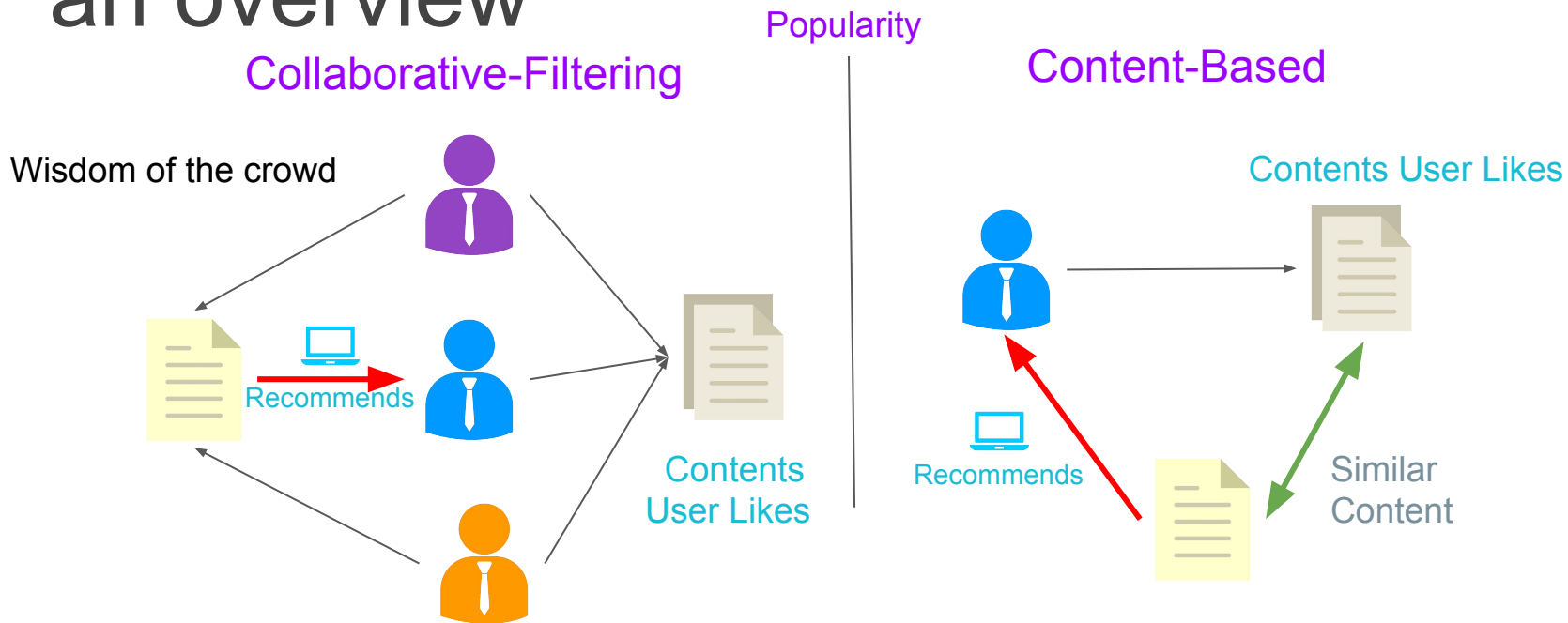


# Recommendation System: an overview



# Agenda

Overview

Content-based recommendation system

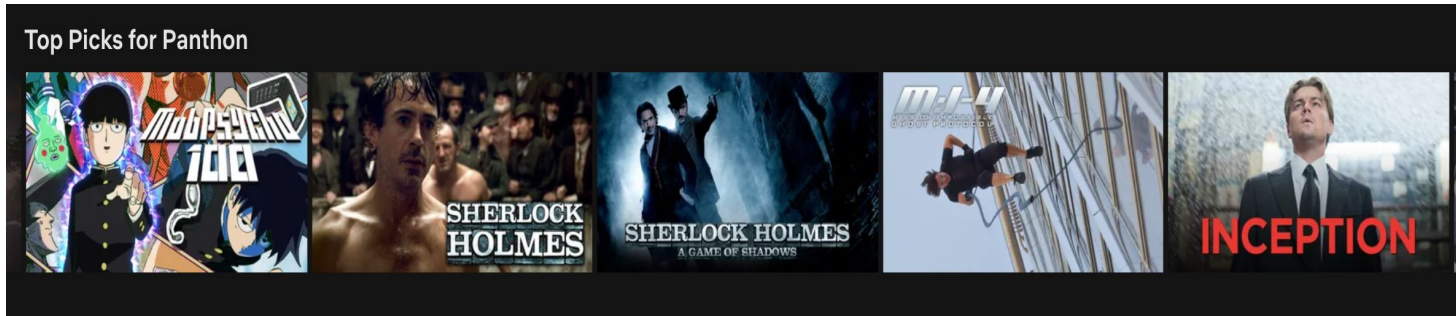
Collaborative filtering systems

Metrics

Lab

# What is the Goal of a Recommendation System?

- Predict the user's preference toward an item, and recommends it!
- Example: Netflix, Amazon



# What do We Know About Our Users?

## Explicit Data

- What the Users told us
- Example: Item feedbacks



Sapporo Ichiban Chow Mein, 3.6-Ounce Packages...

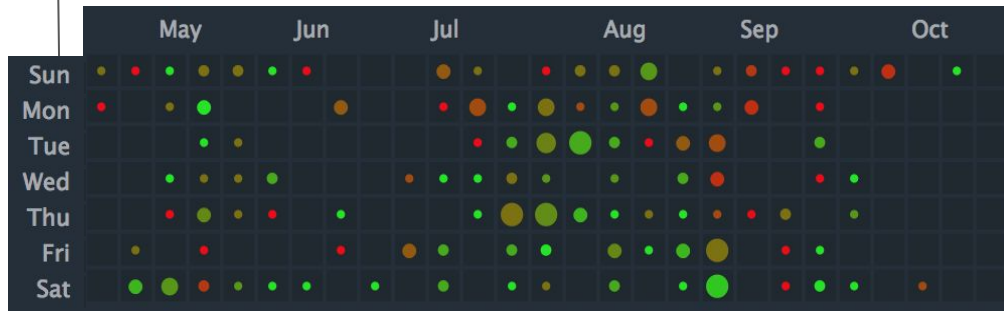
Sapporo



I love it

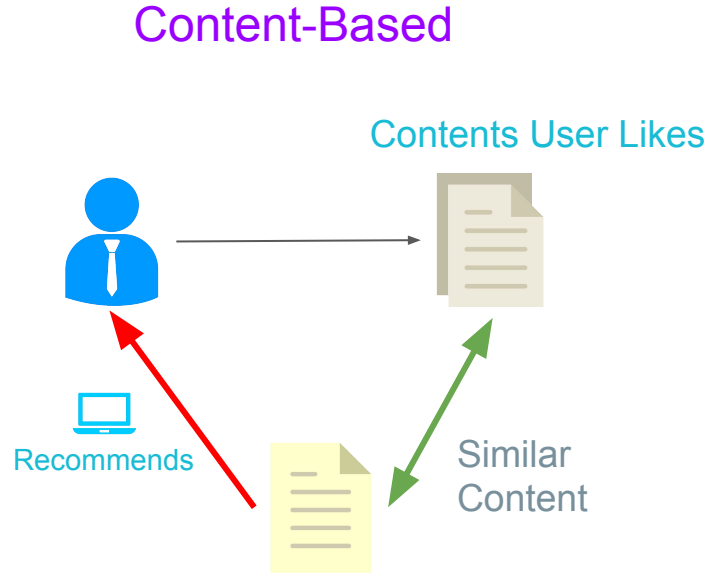
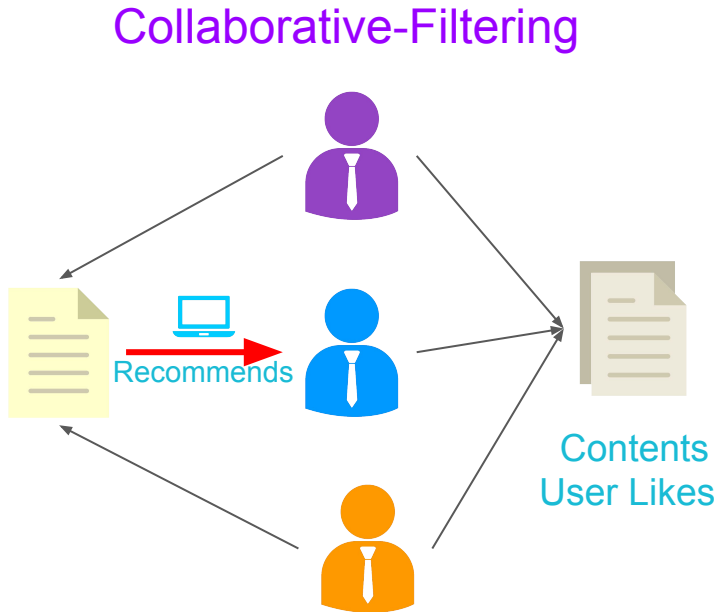
## Implicit Data

- What we infer from the user's activity
- Example: Time spent browsing on a page, number of times browsing a page



# Our Options:

## Collaborative vs Content-based



# Content-based Recommendation System

- Find an item similar to what the user likes
- Example: Netflix, Amazon

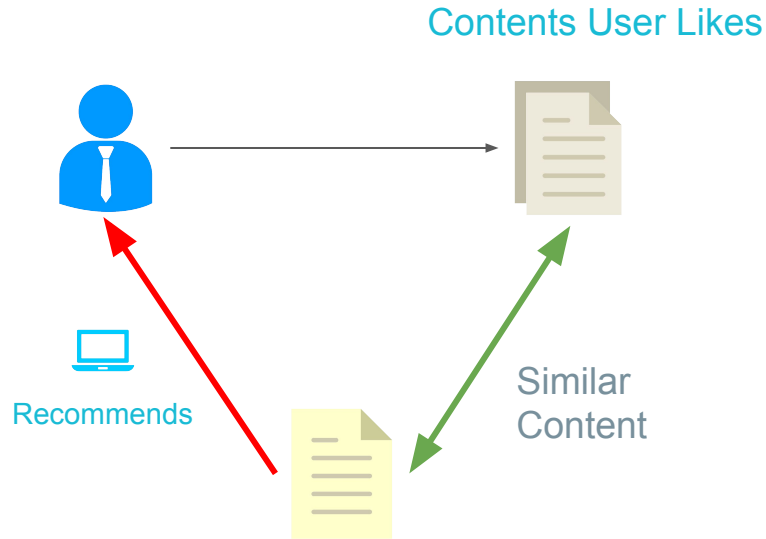
Related to items you've viewed [See more](#)



Because you watched Sherlock >

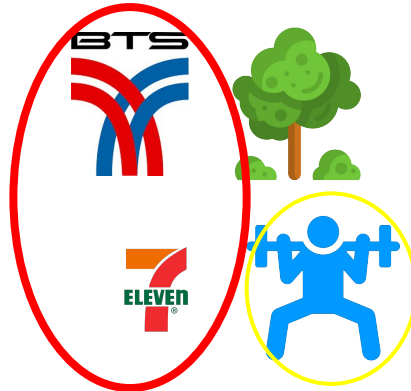
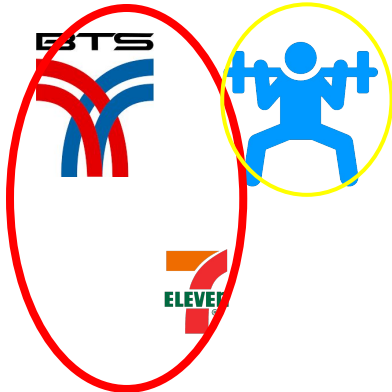
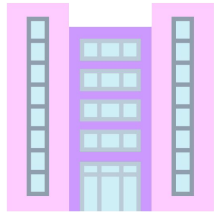


# What We Need To Do:



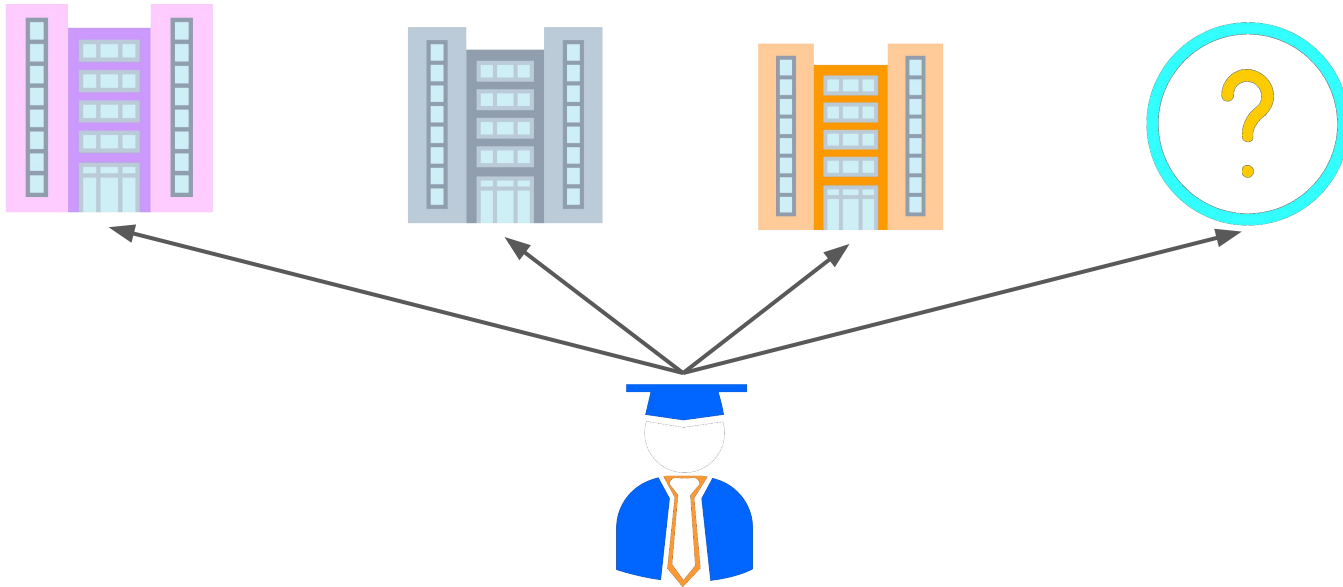
1. Find out what the user likes.
  - Which part of data do we want to use?
  - Rating? Browsing time? Browse count?
2. Construct a User Profile:
  - Which kinds of items does he like?
  - What is the similarity between the items the user likes
3. Match a new item to the User Profile
  - How much will the user like it?
4. Make Prediction!

# Eyeballing: What Do the Condos Have in Common?

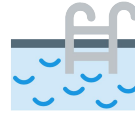
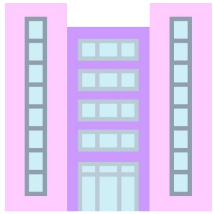




# An Intuitive Example: Home browsing

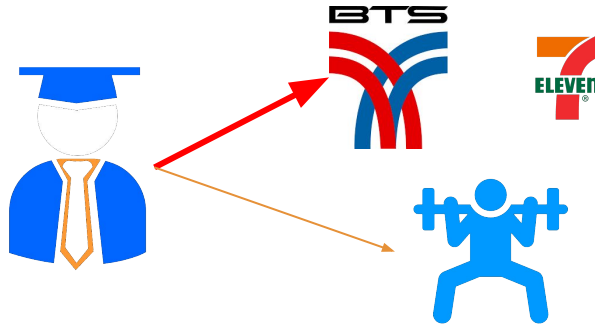


# Eyeballing: What Do the Condos Have in Common?

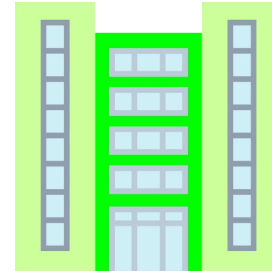
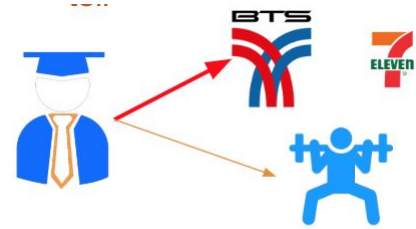


# Thus, our customer's profile is:

- Need to be close to the train system
- Need to be close to a convenience store
- Maybe prefer places with fitness center as well, we have too little data to tell

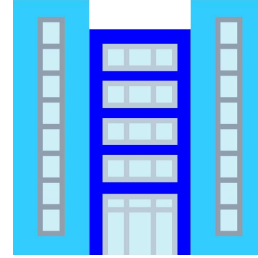
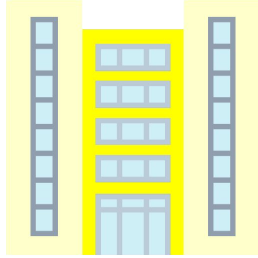
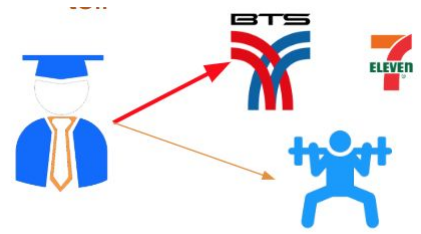


Thus, between



Which should we recommend?

# What about?






The answer is not as obvious..

Thus we need a systematic way to tell how well a place fits our model





# Item Profile

1 = have  
0 = don't have

					
	1	1	1	1	0
	0	1	0	1	0
	1	1	0	1	0
	0	0	1	0	1
	1	1	1	0	1

Simple Way: we know that the customer have browsed the first three home...









So we can create a user profile by a simple average of the item profile the user browse (Or we can also do weighted average or any more complexed algorithm if we want to!)

			
1	1	1	1
0	1	0	0.33
1	1	0	0.67
0	0	1	0.33
1	1	1	1

---

3

# Now we can compare which is more similar!

			
	1	1	0
	0.33	1	0
	0.67	1	0
	0.33	0	1
	1	0	1

It's still not obvious... but at least we have numbers to work with!

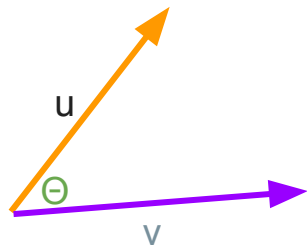


# Cosine Similarity

- Our profiles could be viewed as vectors
- We can use the Cosine Similarity to normalize and compare how similar the vectors are

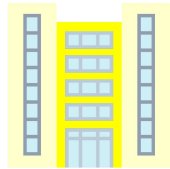
Recall that  $u \cdot v = |u||v|\cos\theta$

$$\cos\theta = \frac{u \cdot v}{|u||v|}$$

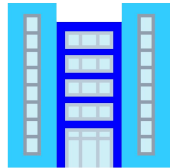


The value of  $\cos \Theta$  is called the Cosine Similarity, with value closer to one indicating that the vectors point roughly in the same direction.

## Example: Cosine Similarity



$$\cos\theta = \frac{1 \cdot 1 + 0.33 \cdot 1 + 0.67 \cdot 1}{|1.633||\sqrt{3}|} = 0.707$$



$$\cos\theta = \frac{0.33 \cdot 1 + 1 \cdot 1}{|1.633||\sqrt{3}|} = 0.470$$

The yellow Condo is more similar to what our user has browsed!

Other distance/similarity functions: Euclidean, Jaccard, Earth Mover, etc.

# Closest?

We need some kind of **distance** or **similarity** measures

$$F(\mathbf{X}_1, \mathbf{X}_2) = d$$
$$\mathbf{X}_1 = [x_{1,1}, x_{1,2}, \dots, x_{1,n}]$$
$$\mathbf{X}_2 = [x_{2,1}, x_{2,2}, \dots, x_{2,n}]$$

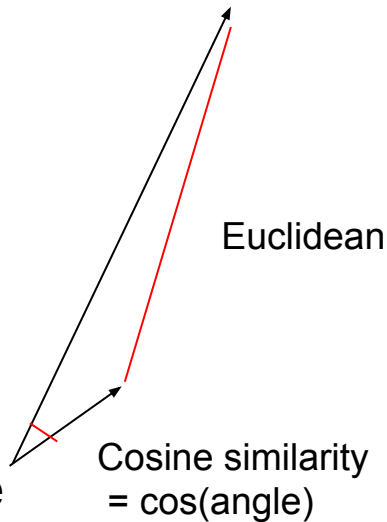
Euclidean distance

$$\sqrt{\sum_i (x_{1,i} - x_{2,i})^2}$$

Cosine similarity

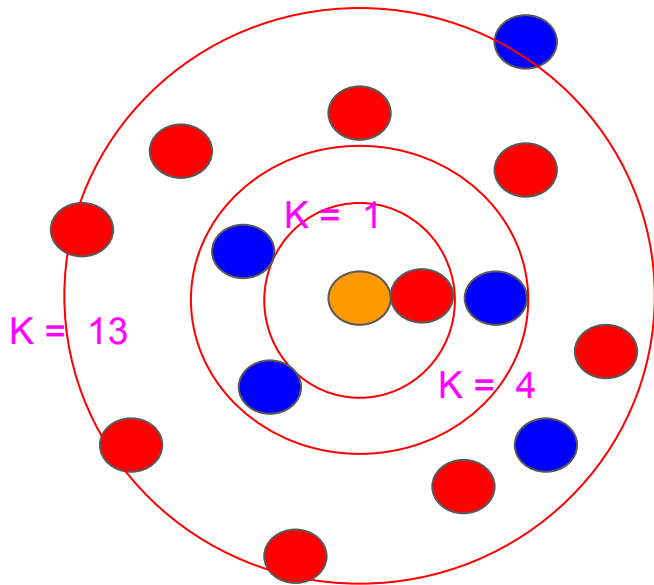
$$\frac{\mathbf{X}_1 \cdot \mathbf{X}_2}{|\mathbf{X}_1| |\mathbf{X}_2|} = \frac{\sum_i x_{1,i} x_{2,i}}{\sqrt{\sum_i x_{1,i}^2} \sqrt{\sum_i x_{2,i}^2}}$$

Many more distances, Jaccard distance, Earth mover distance



# K-Nearest Neighbor

- Classify something based on  $k$  items that are most similar to it.



# Nearest Neighbour classification

Find the closest training data, assign the same label as the training data

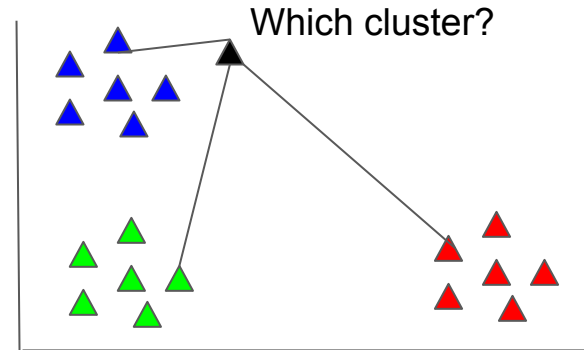
Given query data

For every point in the training data

Compute the distance with the query

Assign label of the smallest distance

Brand royalty



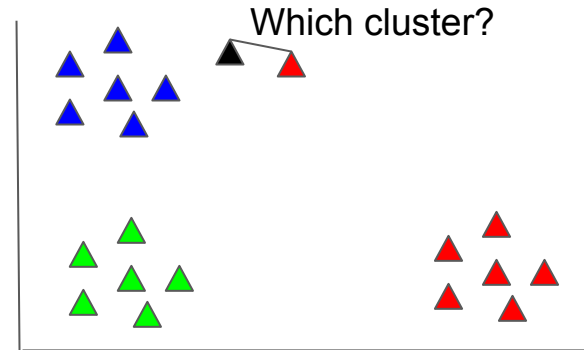
Price sensitivity

# K-Nearest Neighbour (kNN) classification

Nearest Neighbour is susceptible to noise in the training data

Use a voting scheme instead

Brand royalty



Price sensitivity

# K-Nearest Neighbour (kNN) classification

Nearest Neighbour is susceptible to noise in the training data

Use a voting scheme instead

Given query data

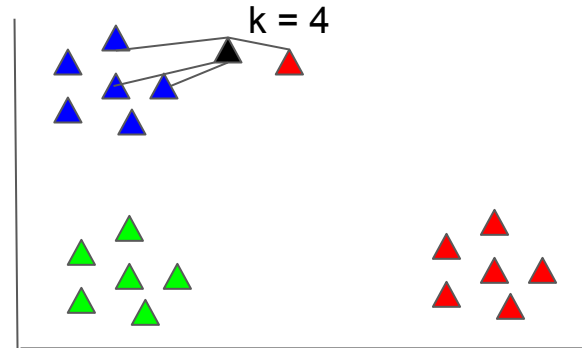
For every point in the training data

Compute the distance with the query

Find the K closest data points

Assign label by voting

Brand royalty



Price sensitivity

# K-Nearest Neighbour (kNN) classification

Nearest Neighbour is susceptible to noise in the training data

Use a voting scheme instead

Given query data

For every point in the training data

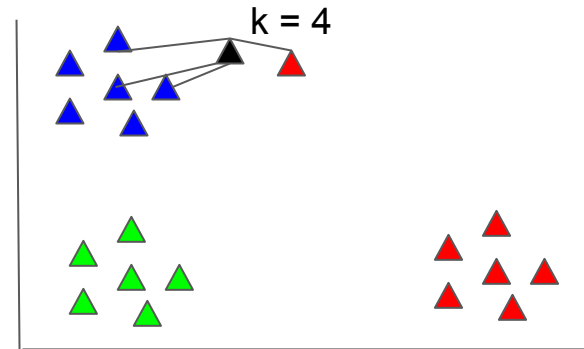
- Compute the distance with the query

- Find the K closest data points

- Assign label by voting

The votes can be weighted by the  
inverse distance (weighted k-NN)

Brand royalty



Price sensitivity



# KNN runtime

For every point in the training data

- Compute the distance with the query

- Find the K closest data points

- Assign label by voting

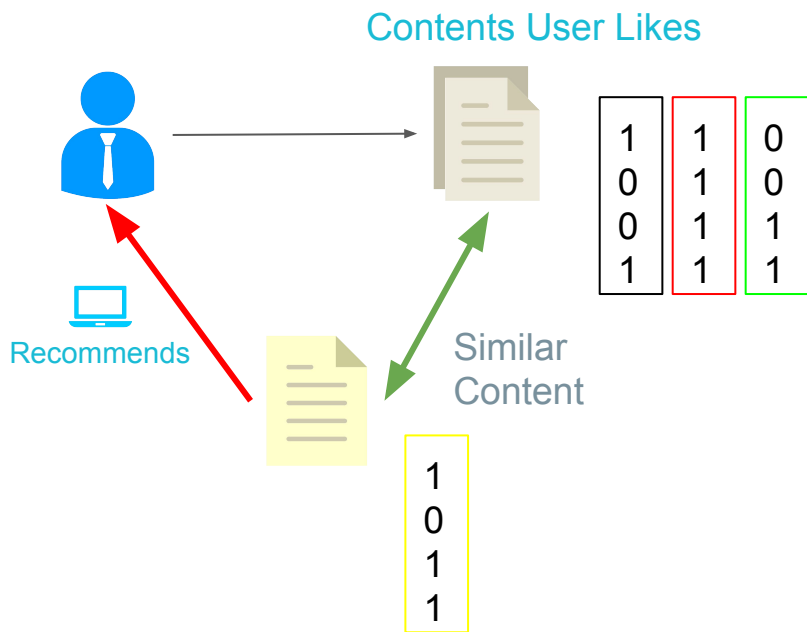
$O(N)$

$O(JN)$  - If we have J queries

Ways to make it faster: Kernelized KNN, locally Sensitive Hashing (LSH), use centroids

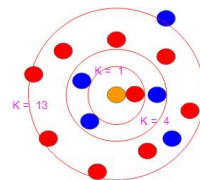
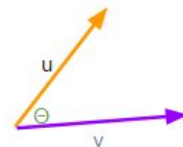
# Let's review: Content-based

## Content-Based



- Represent items as **features**
- Create a **user profile** based on the items the user interacts with
- **Find** items based on the user profile

1	1	0
0.33	1	0
0.67	1	0
0.33	0	1
1	0	1



# Context information

There's many information available

Ratings/views

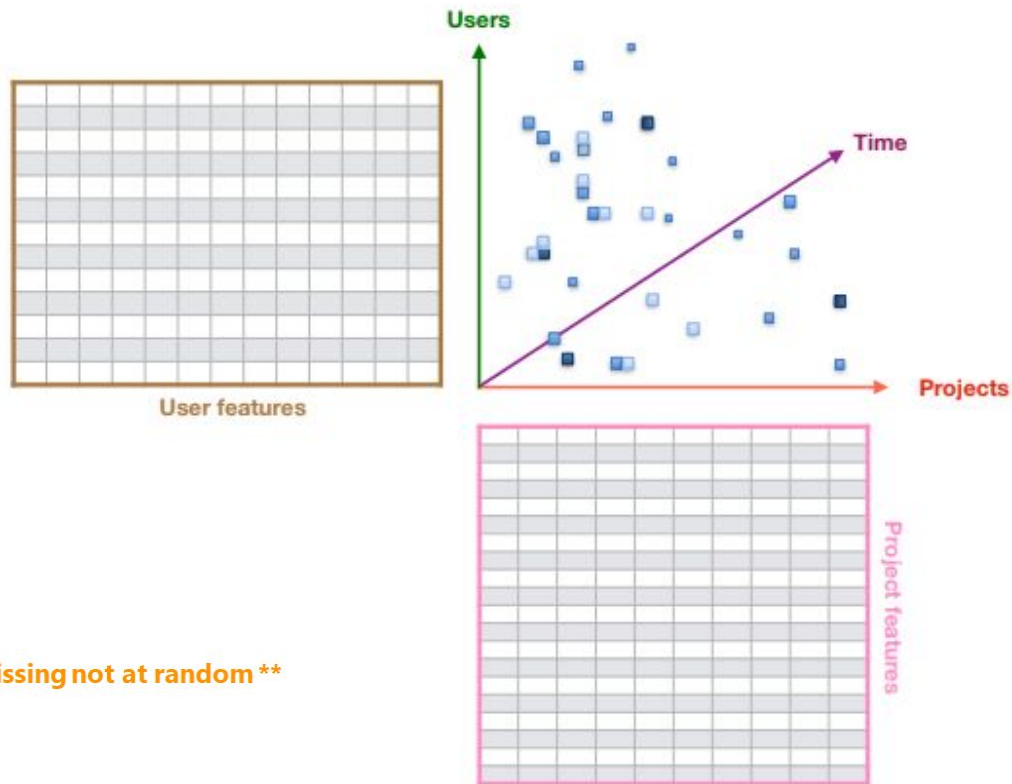
Time

Missing ratings

Side information

User information

Item information



# Matrix view (ignoring time)

Conventional collaborative filtering  
tries to fill in the blanks of the  
view/rating matrices

0/1 matrix

Rating matrix

Implicit info matrix

How to fill?

							5	
3				10				
		0						
						1		
		5						
						7		
2								
					8			
				6			5	
		7						
					4			
3								

# Evaluation metrics

Recommendation system commonly used metrics

Best metric is via A/B testing: click through rate, conversion rate, time spent on content, etc.

Mean Square Error (MSE)

Mean Average Precision (MAP@K)

normalized Discounted Cumulative Gain (nDCG)

# Evaluation: Precision and Recall

- **Precision** is the **correctness** of our recommendation

Precision = Number of Correct Recommendation / Number Recommended

- **Recall** is the ability of our recommendation to obtain the correct recommendation

Recall = Number of correct answer that get recommended / number of correct answers

# Precision and Recall Example

We recommend [H,F,T,A,E],

the correct answer is [A,B,C,D]

- We got 1 item(A) correct out of 5 recommended items, so our **precision** is

$$\frac{1}{5} = 0.2$$

- We got 1 item(A) out of 4 correct answers, so our **recall** is  $\frac{1}{4} = 0.25$

# Evaluation: Average Precision

- Average Precision = the Sum of (precision at i) x (the change in recall)
- Example: We recommend [E,K,A,P,O,L], while the correct answer is [A,B,C,D,E]
- We get two items right at position 1 and 3
- $\text{Precision@1} = 1/1 = 1$        $\text{Precision@3} = 2/3 = 0.67$
- Note: Change in recall is always equal to  $1/(\text{number of correct items}) = 1/5$
- Average Precision = sum of precision over correct items/number of correct items =  $(1+0.67)/5 = 1.67/5 = 0.334$
  
- What if Instead of [E,K,A,P,O,L] we recommend [E,A,K,P,O,L]
- $\text{AP} = (1/1+2/2)/5 = 0.4$  **AP rewards correct rankings!**

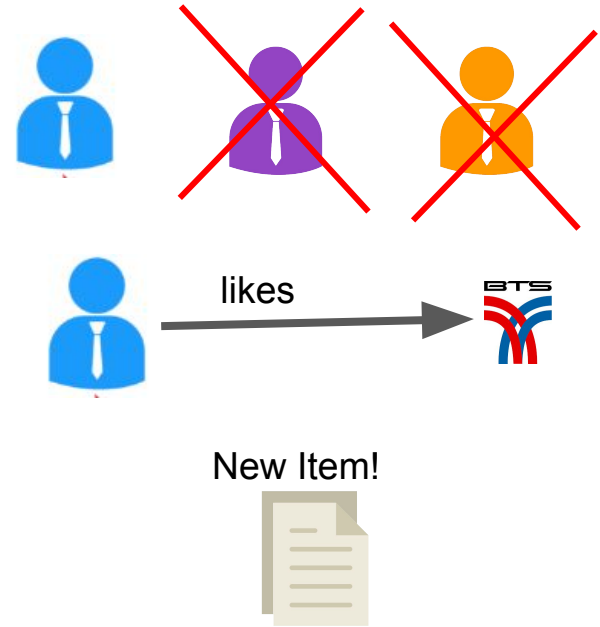


# Evaluation: Mean Average Precision at K (MAP@K)

- Mean Average Precision is simply the mean of AP for multiple users
- In MAP@K, we only care about the first **K** recommendations
  - You are not penalized for incorrect answers in the K recommendations. Send K answers!
  - Change in recall =  $1/\min(K, \text{number correct answer})$
- MAP value both **correctness** of recommendation and **correct rankings**.

# Why should we use Content-based over collaborative filtering?

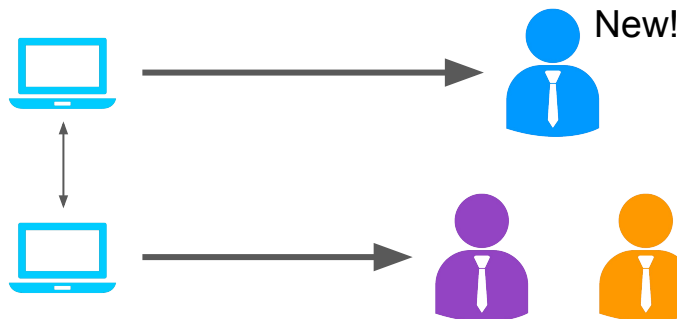
- We don't need to know about any other users to make a prediction
- Can explain why an item is recommended (What attribute in the item that the user likes)
- Works well on new items with no reviews from other users



# Cold Start Problem

- Some recommenders (content-based and collaborative filtering) build user profiles based on past histories
  - We do not have any history for a first time user!
  - Or too little history
- Potential Solution: Hybrid Recommender
  - Some recommenders do not rely on user history(non-personalized recommender etc.)

New!



# Lab overview

1. Data exploration and preprocessing
2. k-NN recommender using user profile
  - a. MAP@K
3. k-NN recommender using collaborative matrix (binary and implicit rating)