How to Design and Build a Recommendation Pipeline in Python

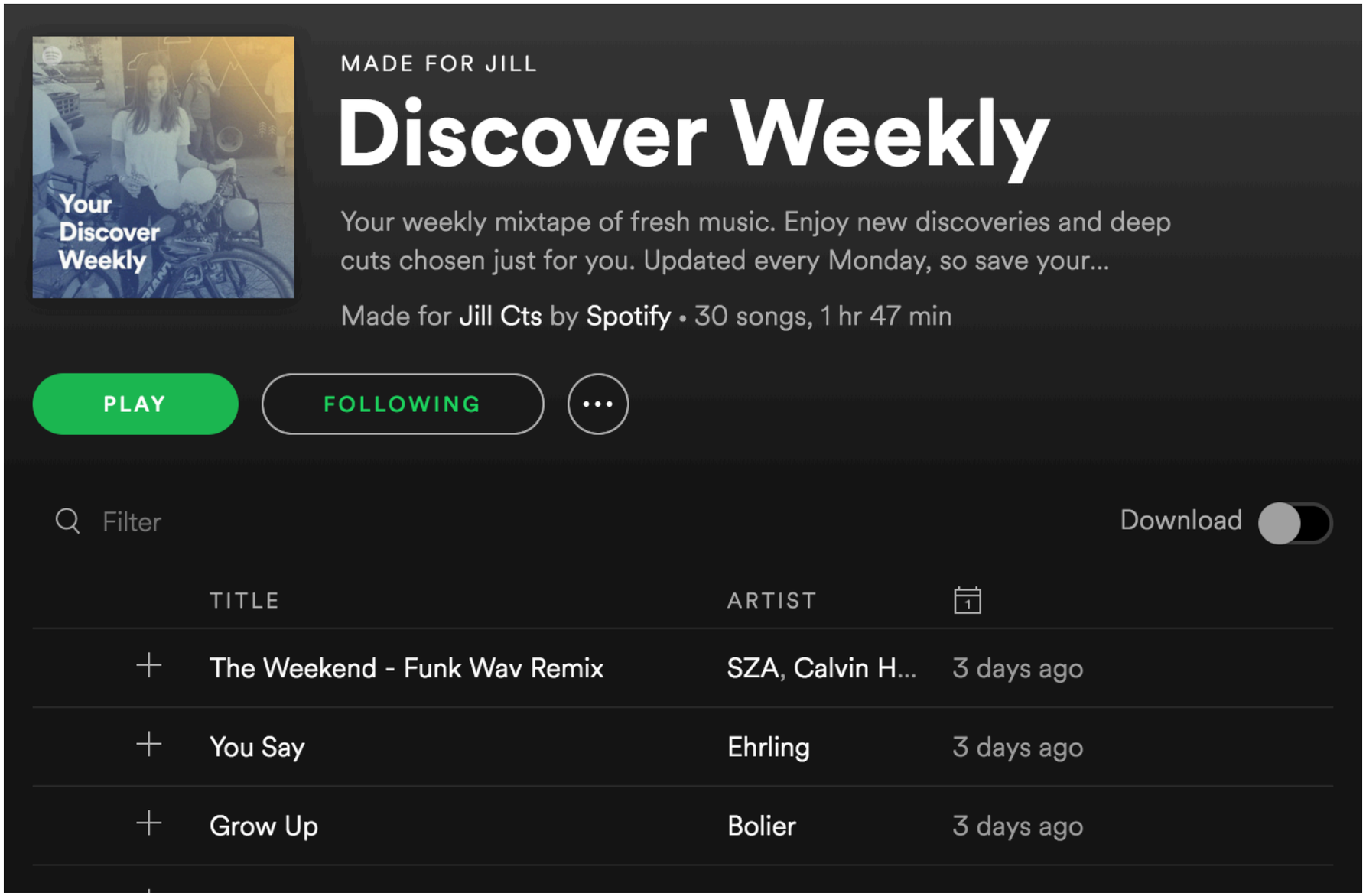
Jill Cates

November 10th, 2018

PyCon Canada

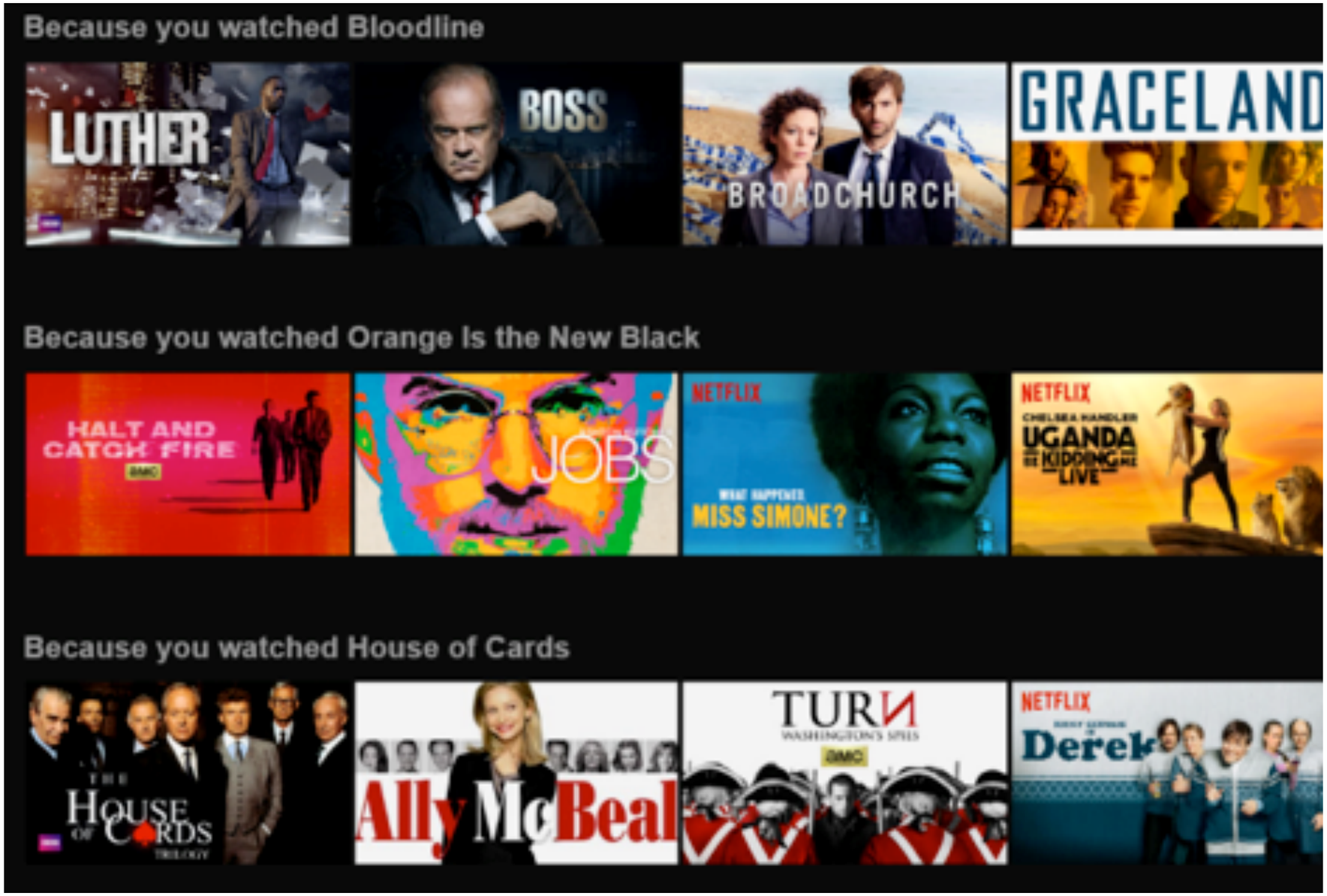
Spotify

Discover Weekly



Netflix

“Because you watched this TV show…”

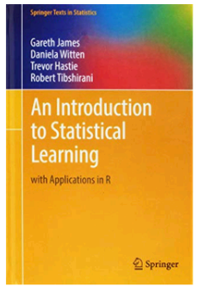


Amazon

"Frequently bought together"

"Customers who bought this item also bought"

Customers who bought this item also bought



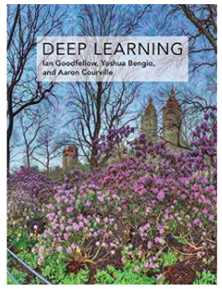
An Introduction to Statistical Learning: with Applications in R

Gareth James

13

Hardcover

CDN$ 77.35 prime



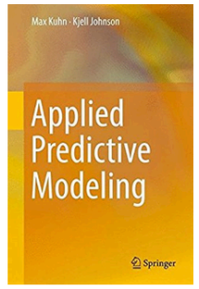
Deep Learning

Ian Goodfellow

26

Hardcover

CDN$ 81.36



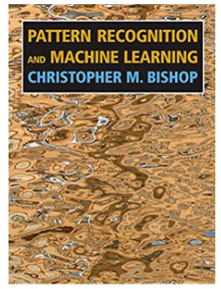
Applied Predictive Modeling

Max Kuhn

8

Hardcover

CDN$ 97.10 prime



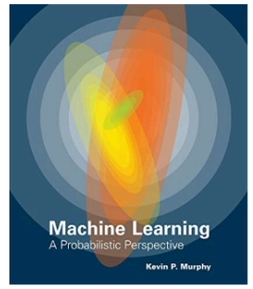
Pattern Recognition and Machine Learning

Christopher M. Bishop

8

Hardcover

CDN$ 86.14 prime



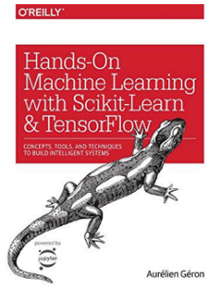
Machine Learning: A Probabilistic Perspective

Kevin P. Murphy

6

Hardcover

CDN$ 120.49 prime



Hands-On Machine Learning with Scikit-Learn and TensorFlow:...

Aurélien Géron

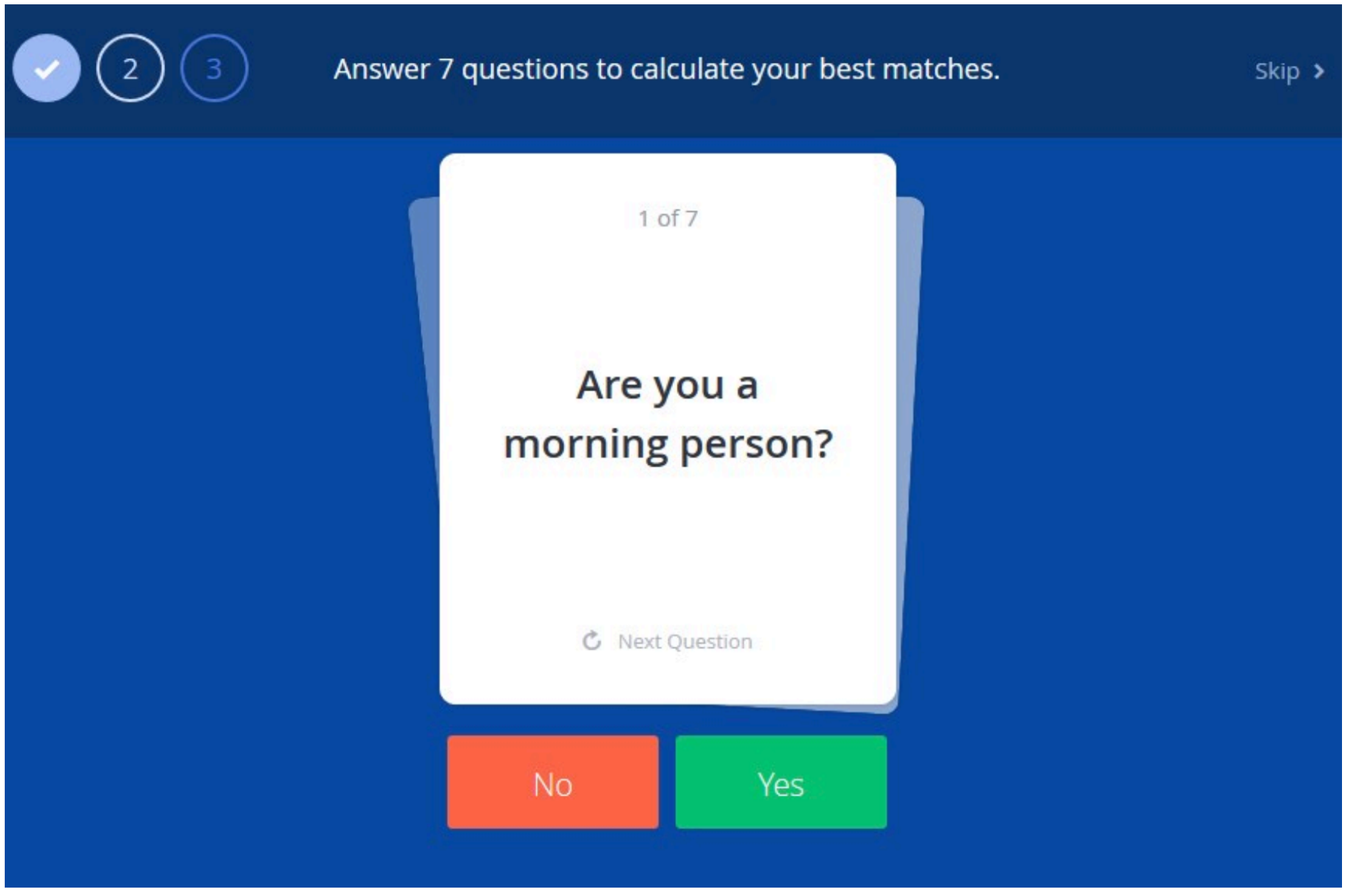
27

Paperback

CDN$ 45.06 prime

OkCupid

“Finding your best match”



Recommender Systems in the Wild

Spotify
Discover Weekly

Amazon
Customers who bought this item also bought

Netflix
Because you watched this show…

LinkedIn
Jobs recommended for you

OkCupid
Finding your best match

New York Times
Recommended Articles for You

GitHub
Repos “based on your interest”

Medicine
Facilitating clinical decision making

Before e-commerce

Things were sold exclusively in brick-and-mortar stores…

limited inventory

mainstream products

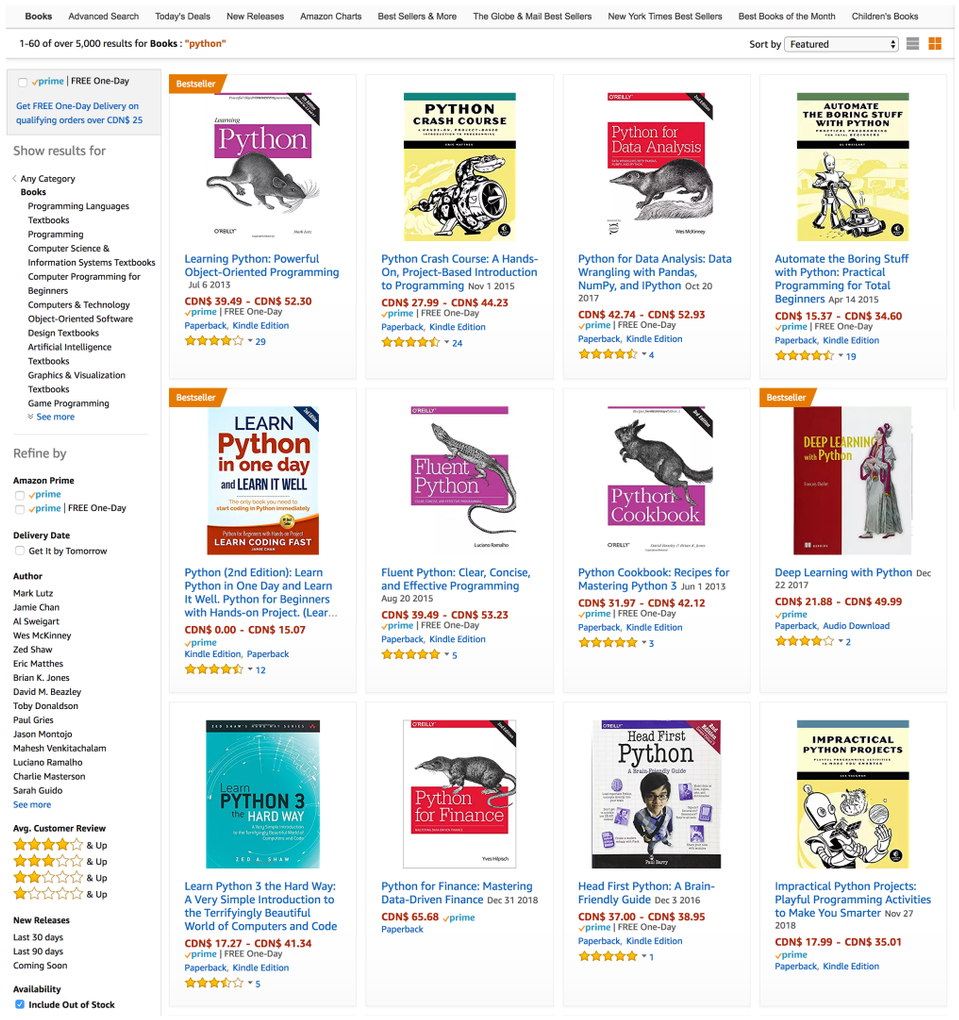
Before e-commerce

Things were sold exclusively in brick-and-mortar stores…

limited inventory

mainstream products

E-commerce



unlimited inventory

niche products

The Tasting Booth Experiment

When Choice is Demotivating: Can One Desire Too Much of a Good Thing?

Sheena S. Iyengar
Columbia University

Mark R. Lepper
Stanford University

6 jam samples

VS.

24 jam samples

The Tasting Booth Experiment

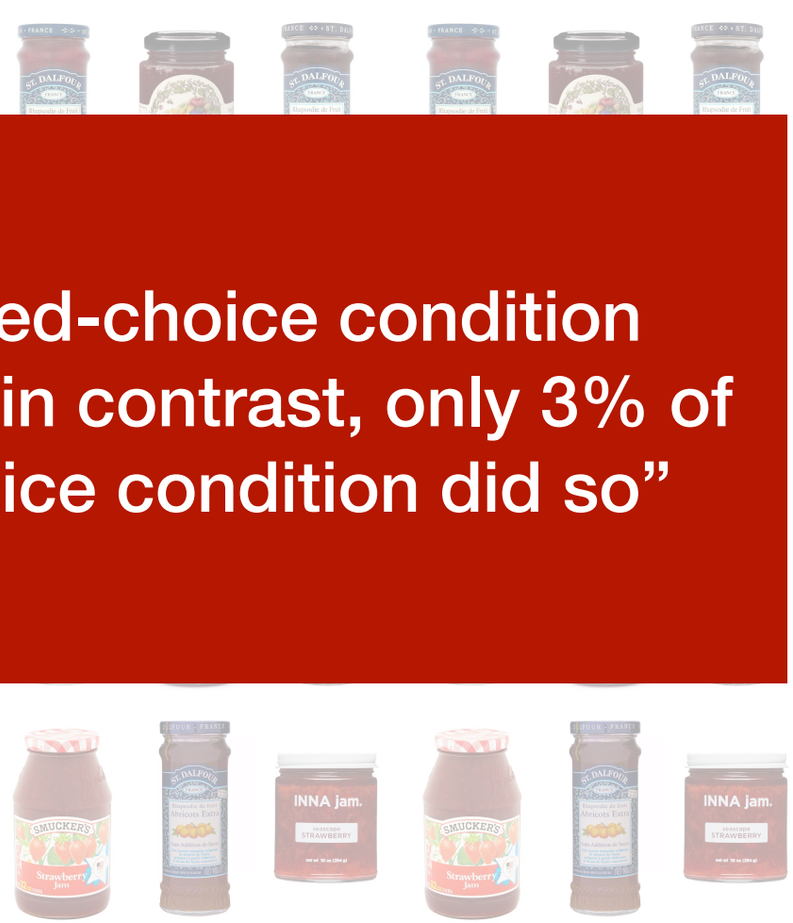
When Choice is Demotivating: Can One Desire Too Much of a Good Thing?

Sheena S. Iyengar
Columbia University

Mark R. Lepper
Stanford University

6 jam samples

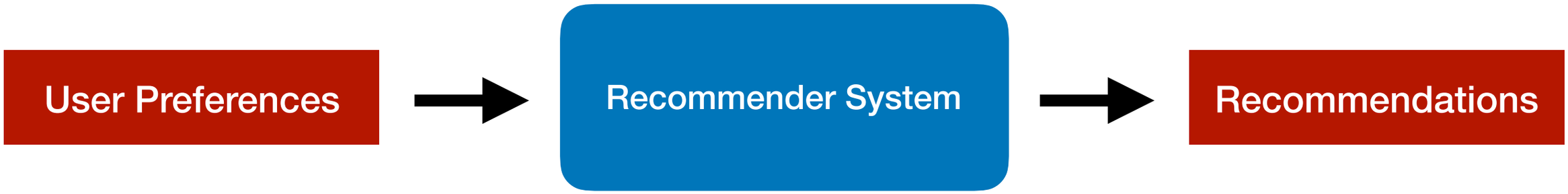
24 jam samples



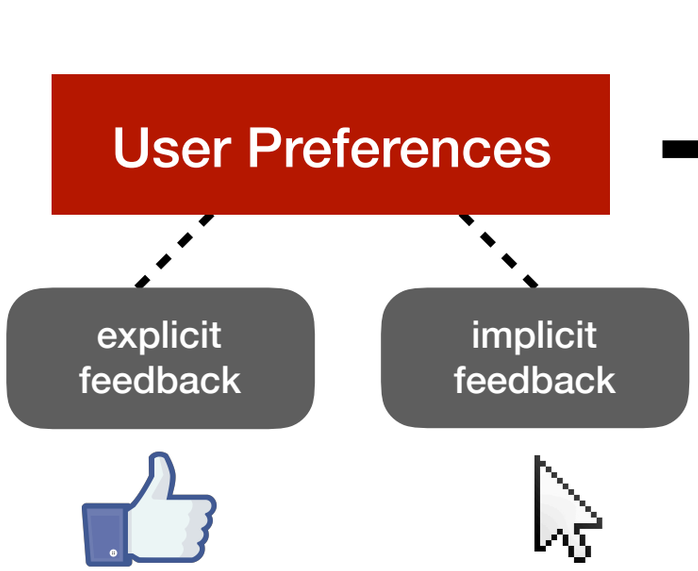
“30% of the consumers in the limited-choice condition subsequently purchased a jar of jam; in contrast, only 3% of the consumers in the extensive-choice condition did so”

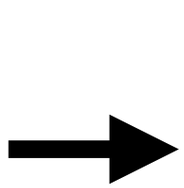
Recommender Crash Course

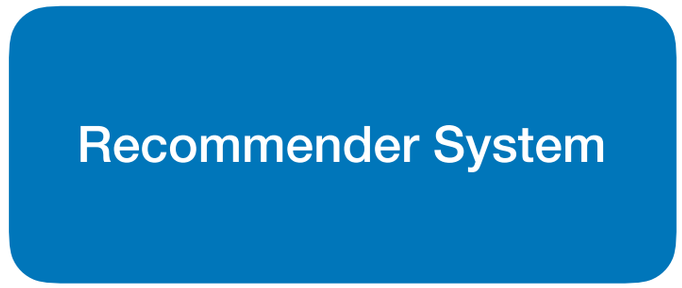
Recommender Crash Course

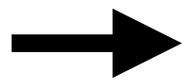


Recommender Crash Course

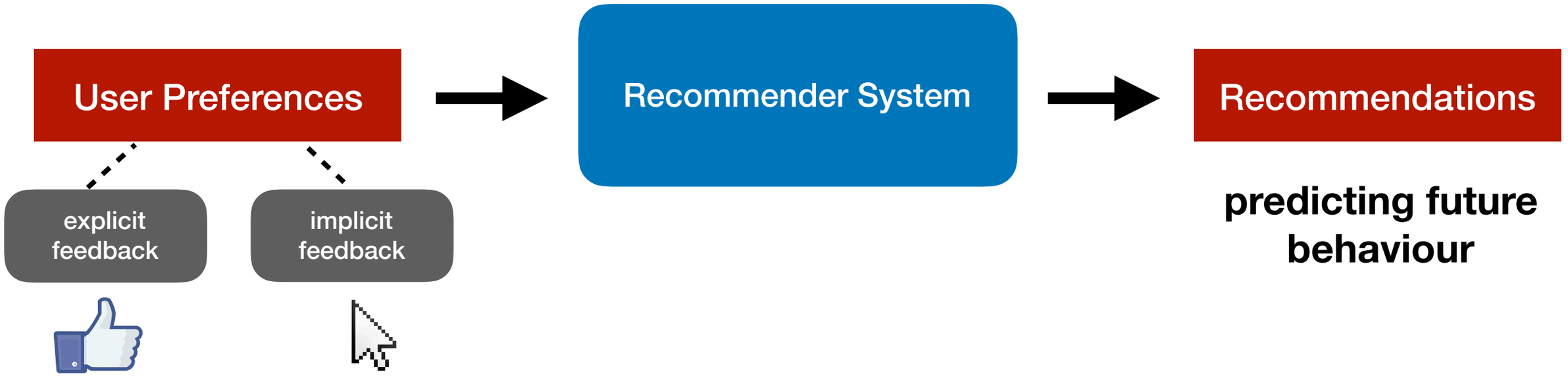




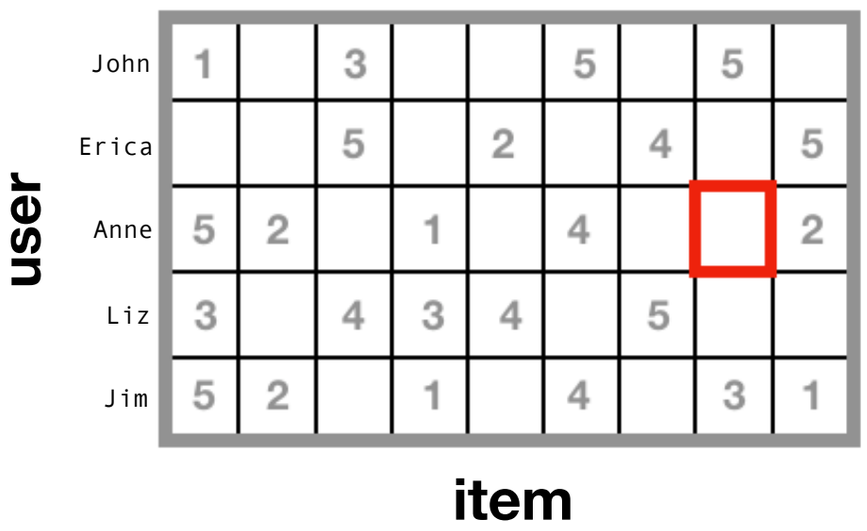




Recommender Crash Course

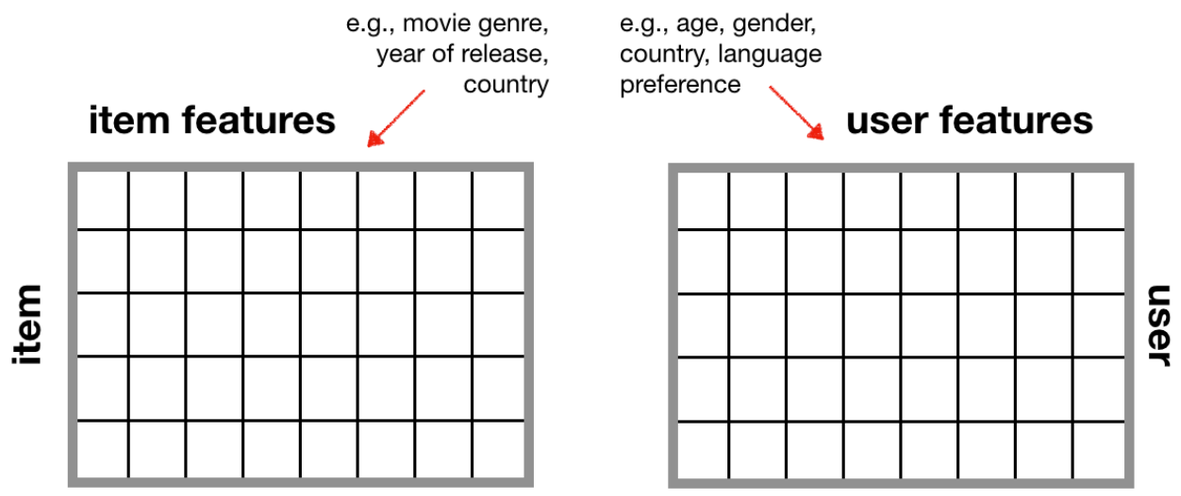


Collaborative filtering



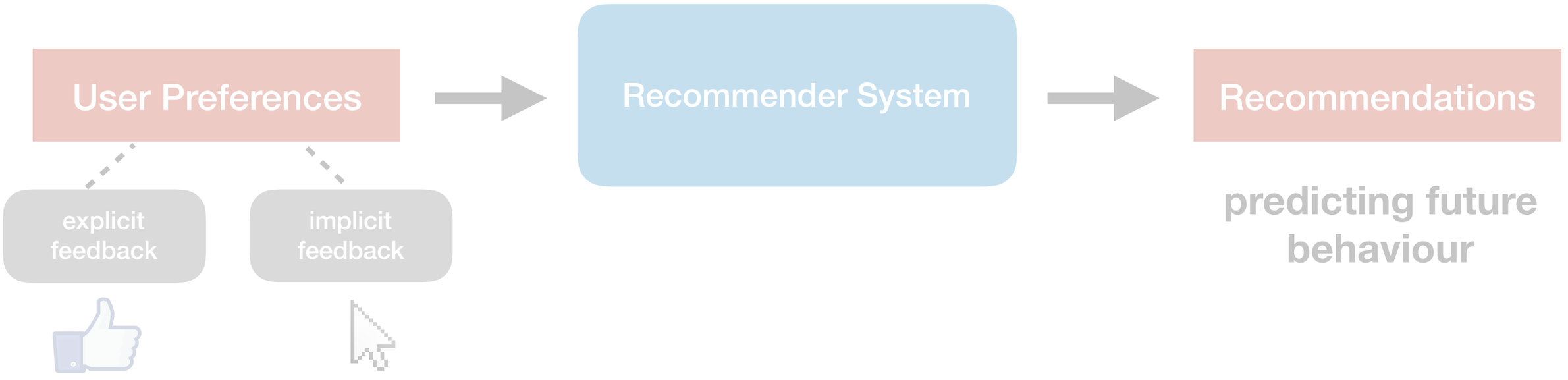
similar users like similar things

Content-based filtering

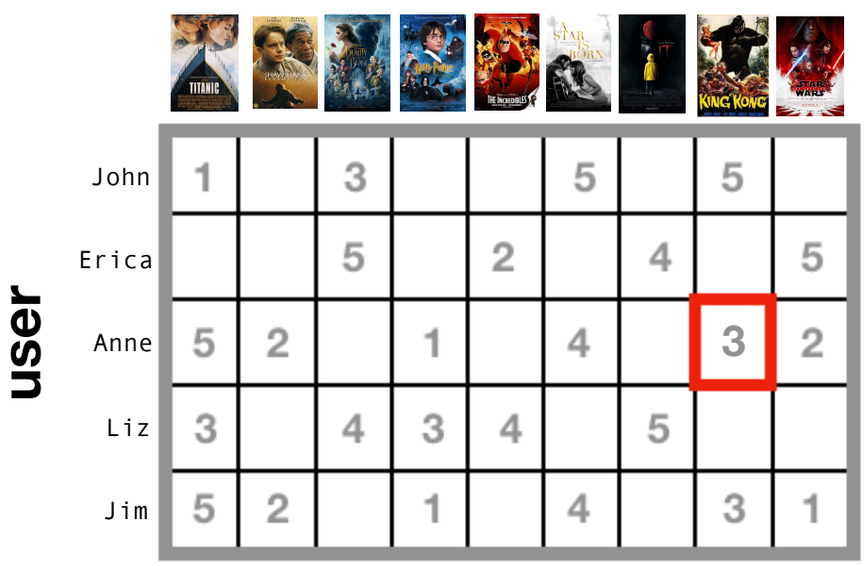


considers items/users features

Recommender Crash Course



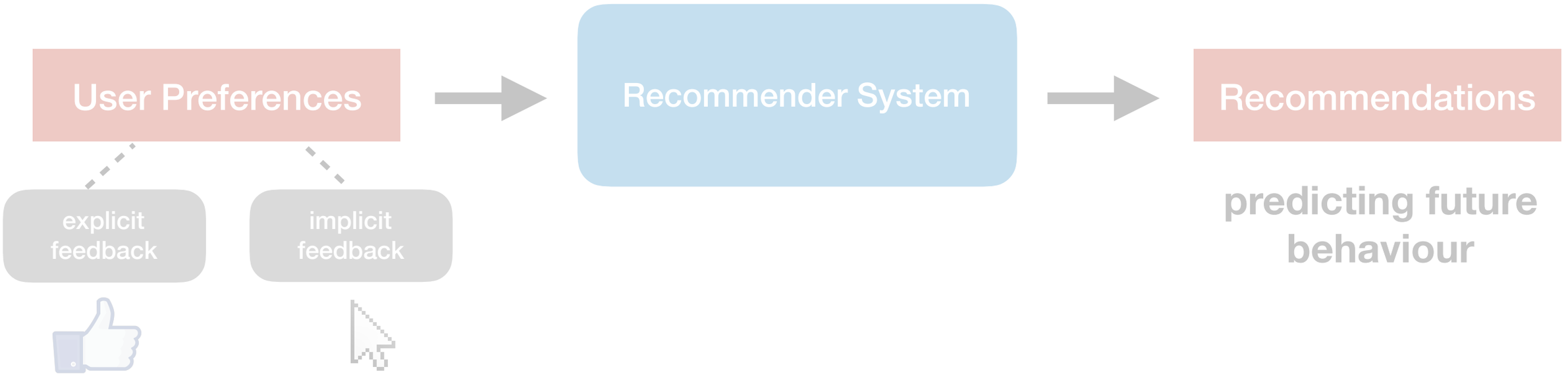
Collaborative filtering



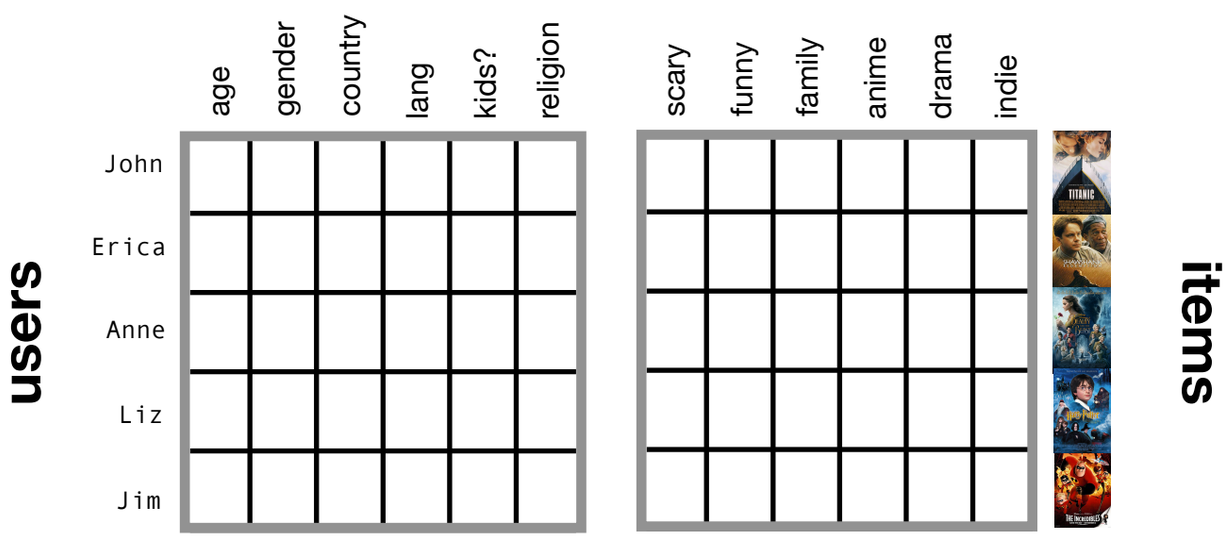
similar users like
similar things

- “Because you watched Movie X”
- “Customers who bought this item also bought”

Recommender Crash Course



Content-based filtering



user and item features

- user features: age, gender, spoken language
- item features: movie genre, year of release, cast

Overview of the Recommender Pipeline

1. Pre-processing
2. Hyperparameter Tuning
3. Model Training and Prediction
4. Post-processing
5. Evaluation

Step 1: Data Pre-processing

user_id	movie_id	rating
2	439	4.0
10	368	4.5
14	114	5.0
19	371	1.0
2	371	3.0
19	114	4.5
3	439	3.5
54	421	2.0
32	114	3.0
10	369	1.0

Step 1: Data Pre-processing

user_id	movie_id	rating
2	439	4.0
10	368	4.5
14	114	5.0
19	371	1.0
2	371	3.0
19	114	4.5
3	439	3.5
54	421	2.0
32	114	3.0
10	369	1.0

items

users

		5.0			4.5		3.0
		3.0		4.0	2.5		3.0
2.0						1.0	
	3.0	3.5		5.0	4.5		
1.5	2.0			4.5		2.0	

Transform original data to user-item (utility) matrix

Step 1: Data Pre-processing

user_id	movie_id	rating
2	439	4.0
10	368	4.5
14	114	5.0
19	371	1.0
2	371	3.0
19	114	4.5
3	439	3.5
54	421	2.0
32	114	3.0
10	369	1.0

items

users

		5.0			4.5		3.0
		3.0		4.0	2.5		3.0
2.0						1.0	
	3.0	3.5		5.0	4.5		
1.5	2.0			4.5		2.0	

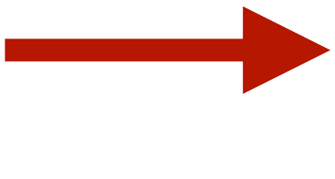
`scipy.sparse.csr_matrix`

SciPy.org

Pre-processing | Hyperparameter Tuning | Model Training | Post-processing | Evaluation

Step 1: Data Pre-processing

user_id	movie_id	rating
2	439	4.0
10	368	4.5
14	114	5.0
19	371	1.0
2	371	3.0
19	114	4.5
3	439	3.5
54	421	2.0
32	114	3.0
10	369	1.0



items

users

		5.0			4.5		3.0
		3.0		4.0	2.5		3.0
2.0						1.0	
	3.0	3.5		5.0	4.5		
1.5	2.0			4.5		2.0	

Calculate Matrix Sparsity

$$\text{sparsity} = \frac{\text{\# ratings}}{\text{total \# elements}}$$

Step 1: Data Pre-processing

Normalization

- Optimists → rate everything 4 or 5
- Pessimists → rate everything 1 or 2
- Need to normalize ratings by accounting for user and item bias
- Mean normalization

  - subtract b_i from each user's rating for given item i

$$b_{ui} = \mu + b_i + b_u$$

user-item rating bias

global avg

item's avg rating

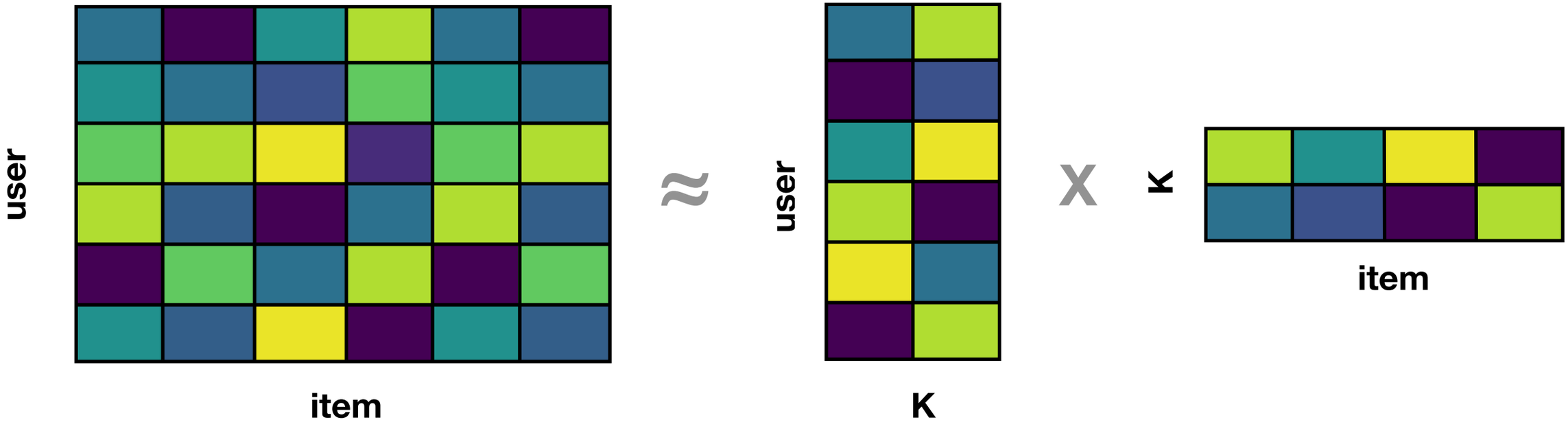
user's avg rating

Pick a Model

Matrix Factorization

- factorize the user-item matrix to get 2 latent factor matrices:
  - user-factor matrix
  - item-factor matrix
- missing ratings are predicted from the inner product of these two factor matrices

$$X_{mn} \approx P_{mk} \times Q_{nk}^{T} = \hat{X}$$

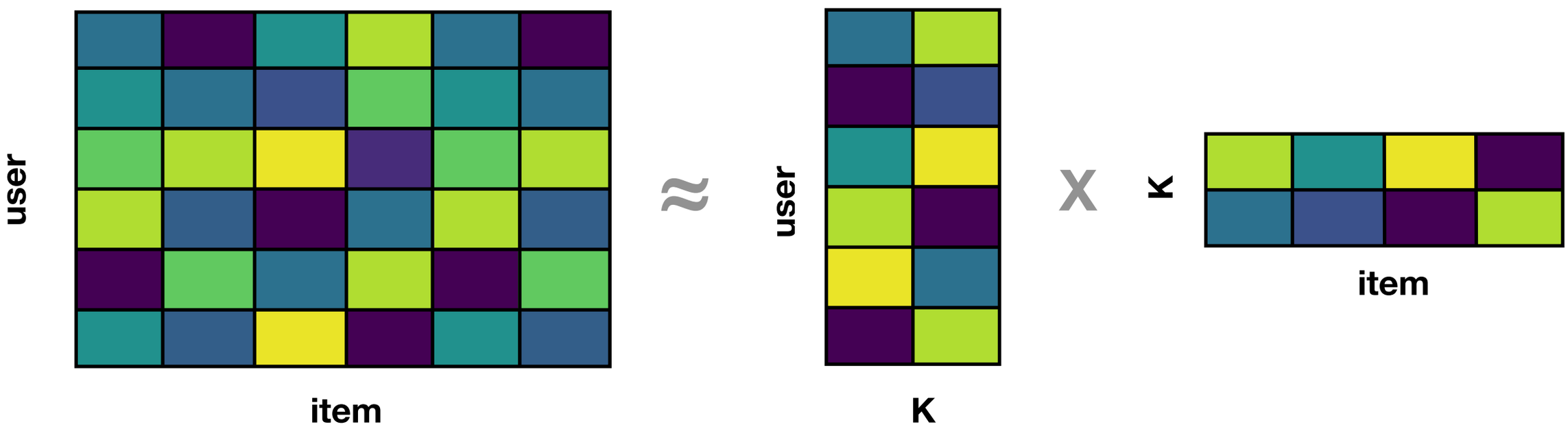


Pick a Model

Matrix Factorization

- Algorithms that perform matrix factorization:
  - Alternating Least Squares (ALS)
  - Stochastic Gradient Descent (SGD)
  - Singular Value Decomposition (SVD)

$$X_{mn} \approx P_{mk} \times Q_{nk}^{T} = \hat{X}$$

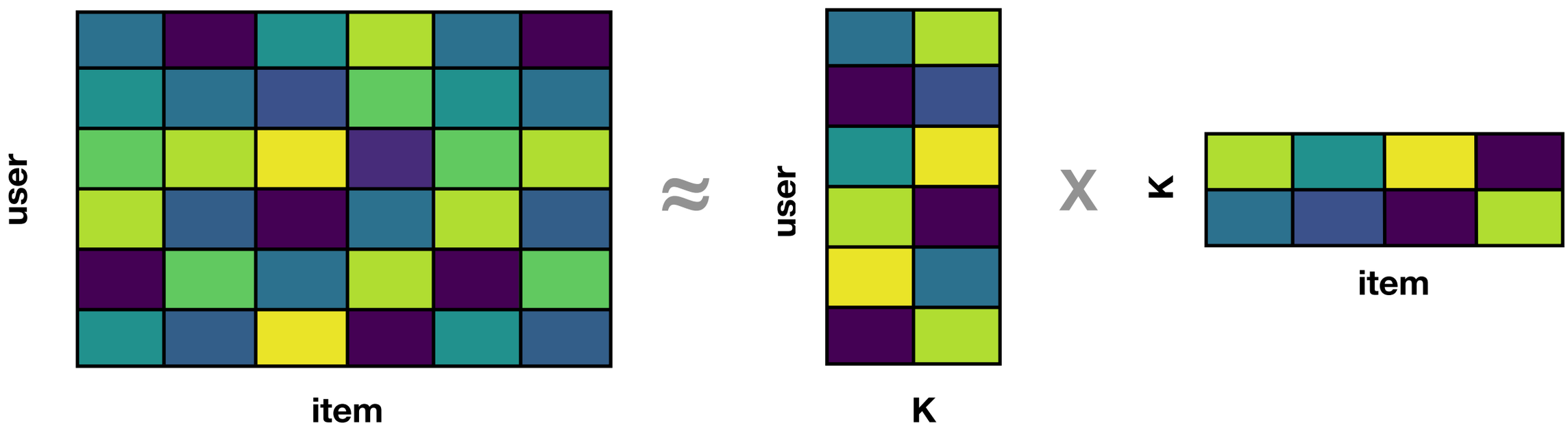


Pick a Model

Matrix Factorization

- Algorithms that perform matrix factorization:
  - Alternating Least Squares (ALS)
  - Stochastic Gradient Descent (SGD)
  - Singular Value Decomposition (SVD)

$$X_{mn} \approx P_{mk} \times Q_{nk}^{T} = \hat{X}$$



Pick an Evaluation Metric

Precision@K

- Of the top K recommendations, what proportion are relevant to the user?

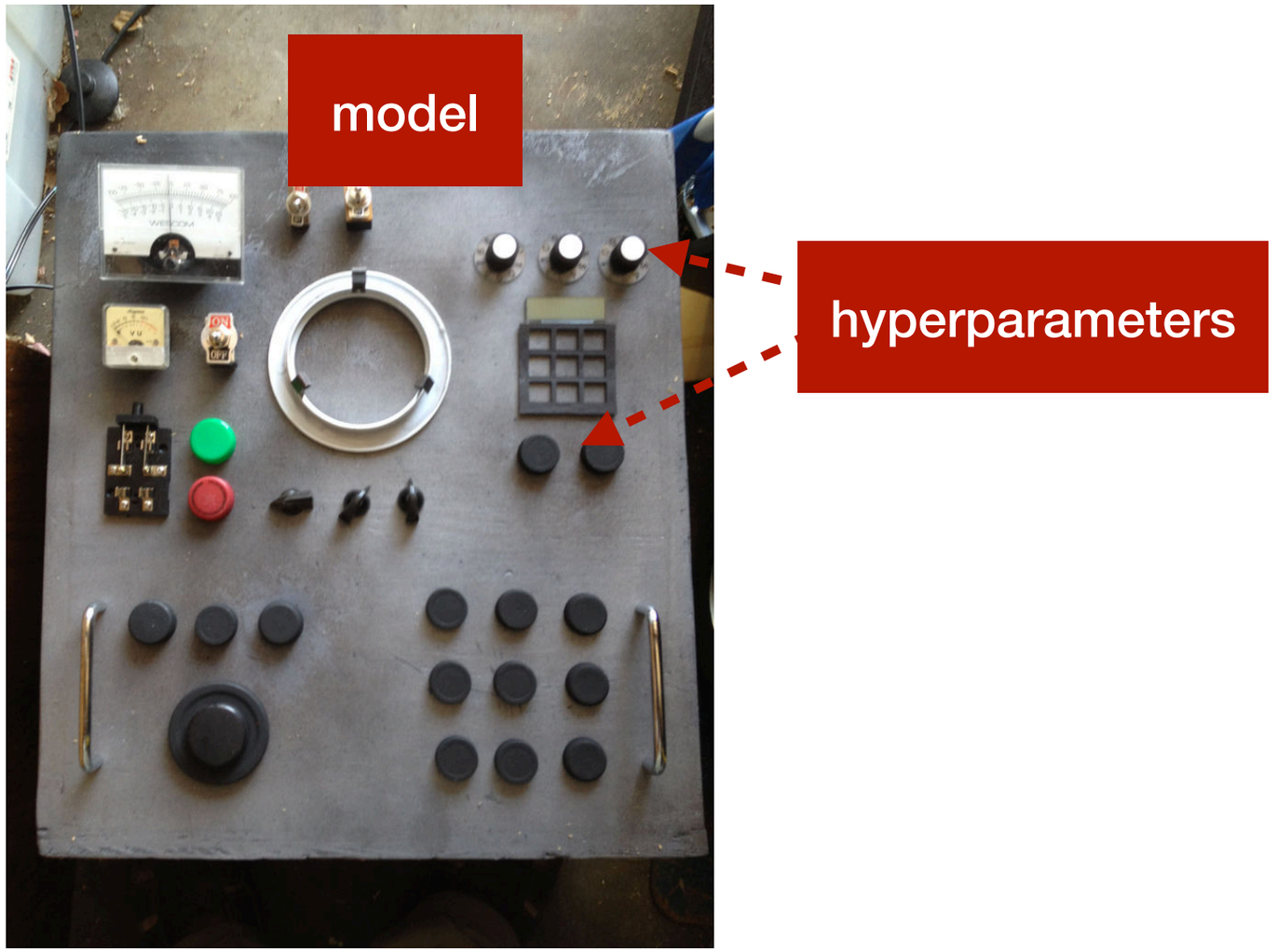
Pick an Evaluation Metric

Precision@10

- Of the top 10 recommendations, what proportion are relevant to the user?

Step 2: Hyperparameter Tuning

What is a hyperparameter?



configuration that is external to the model

Pre-processing	Hyperparameter Tuning	Model Training	Post-processing	Evaluation

Step 2: Hyperparameter Tuning

Alternating Least Square's Hyperparameters

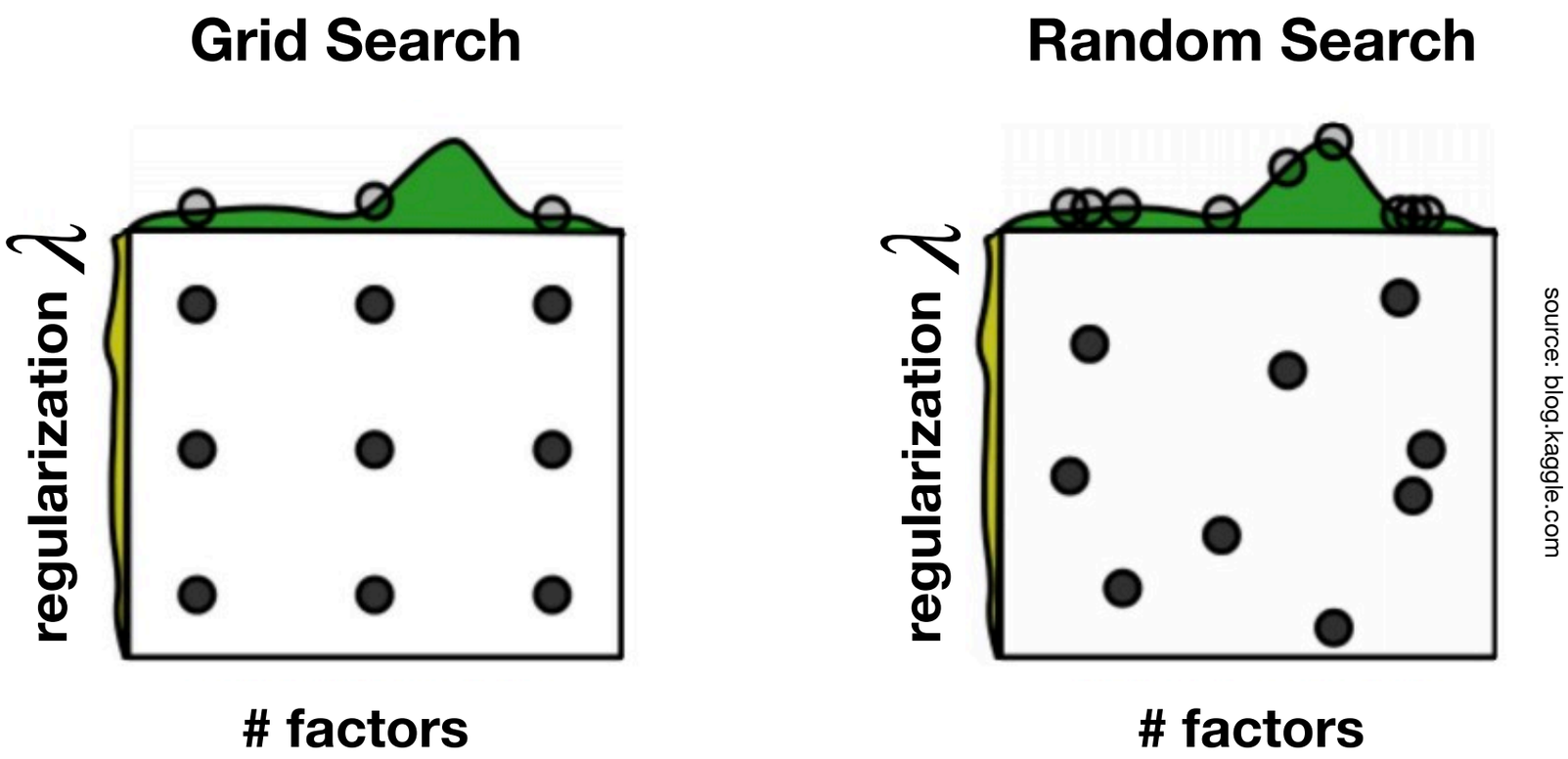
- k (# of factors)
- λ (regularization parameter)

Goal: find the hyperparameters that give the best precision@10

* (or any other evaluation metric that you want to optimize)

Pre-processing	Hyperparameter Tuning	Model Training	Post-processing	Evaluation

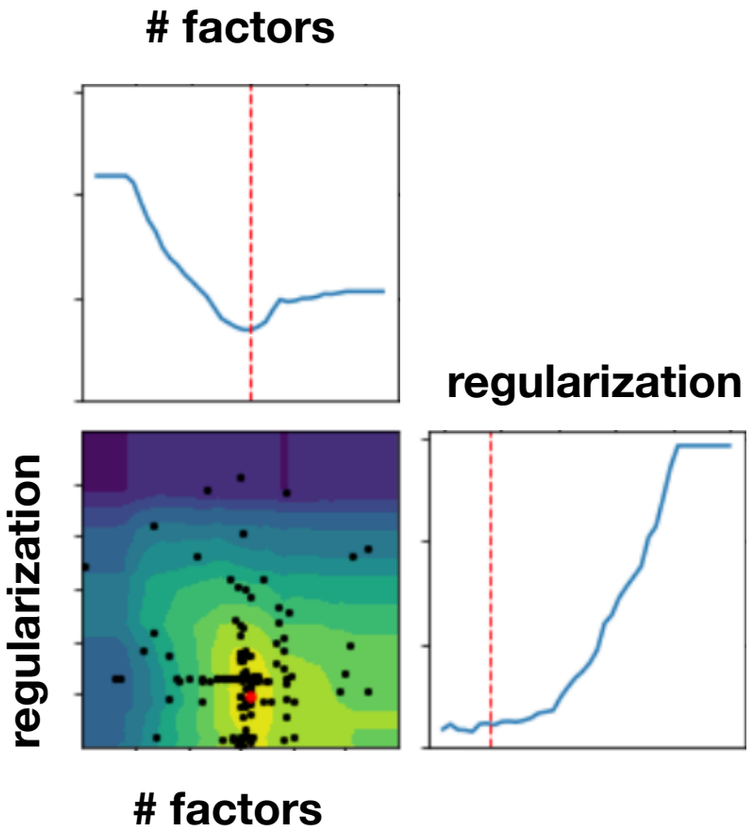
Step 2: Hyperparameter Tuning




```
sklearn.model_selection.GridSearchCV
sklearn.model_selection.RandomizedSearchCV
```

Step 2: Hyperparameter Tuning

Sequential Model-Based Optimization



scikit-optimize (skopt)
hyperopt
Metric Optimization Engine (MOE)

Pre-processing	Hyperparameter Tuning	Model Training	Post-processing	Evaluation

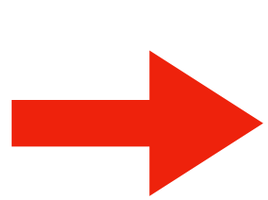
Step 3: Model Training

```
AlternatingLeastSquares(k=8,
    regularization=0.001)
```

items

users

		5.0			4.5		3.0
		3.0		4.0	2.5		3.0
2.0						1.0	
	3.0	3.5		5.0	4.5		
1.5	2.0			4.5		2.0	



items

users

3.12	4.74	5.0	1.22	4.39	4.5	2.75	3.0
0	0	0	0	0	0	0	0
3.99	1.73	3.0	2.93	4.0	2.5	-2.15	3.0
2.0	3.28	5.61	0.82	-1.27	4.79	1.0	4.95
5.10	3.0	3.5	4.77	5.0	4.5	-0.21	5.19
1.5	2.0	-1.84	3.08	4.5	3.27	2.0	-2.46

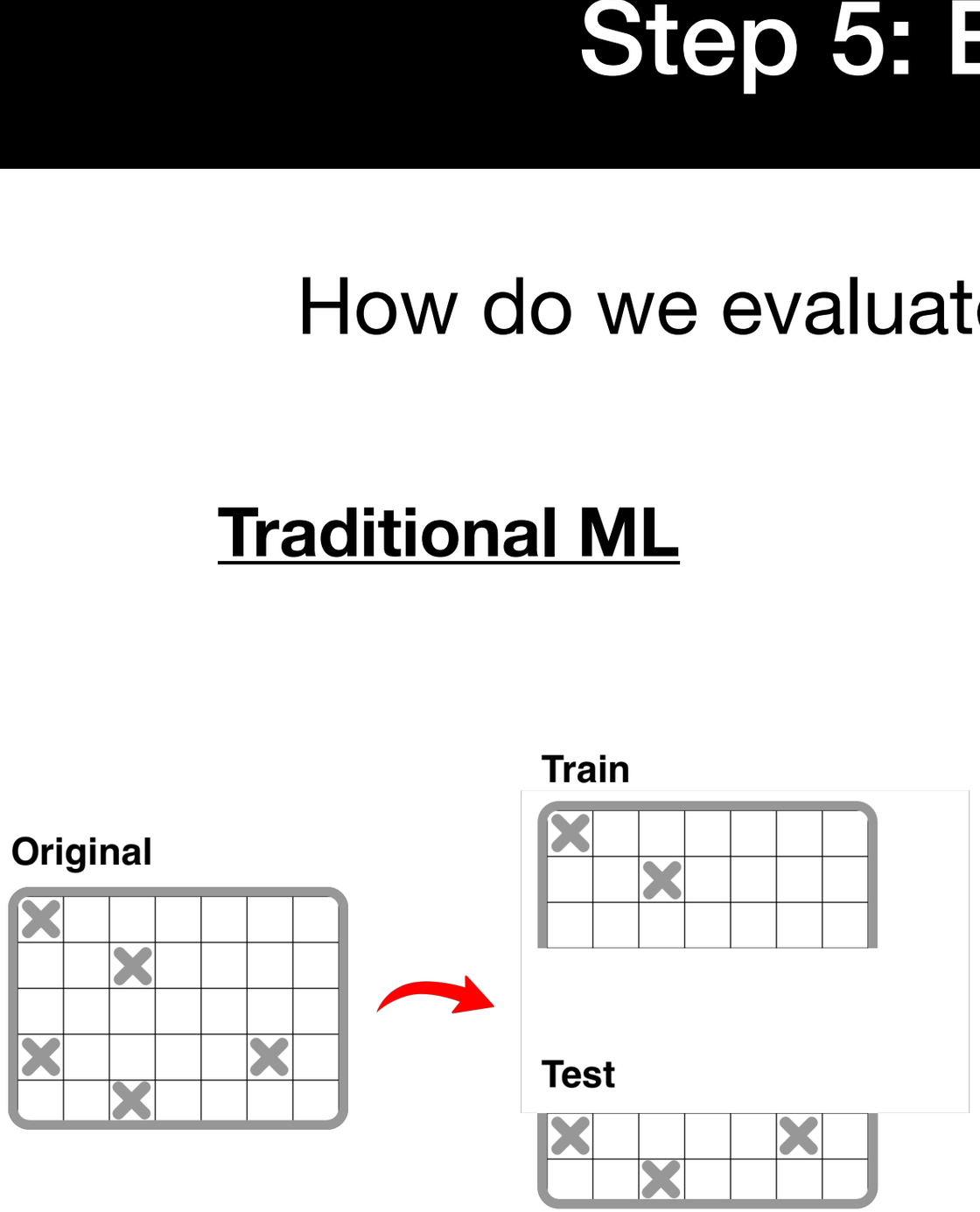
Step 4: Post-processing

- Sort recommendations and get top N
- Filter out items that a user has already purchased, watched, interacted with
- Increase diversity of recommendations
- Item-item recommendations
  - Use a similarity metric (e.g., cosine similarity)
  - “Because you watched Movie X”

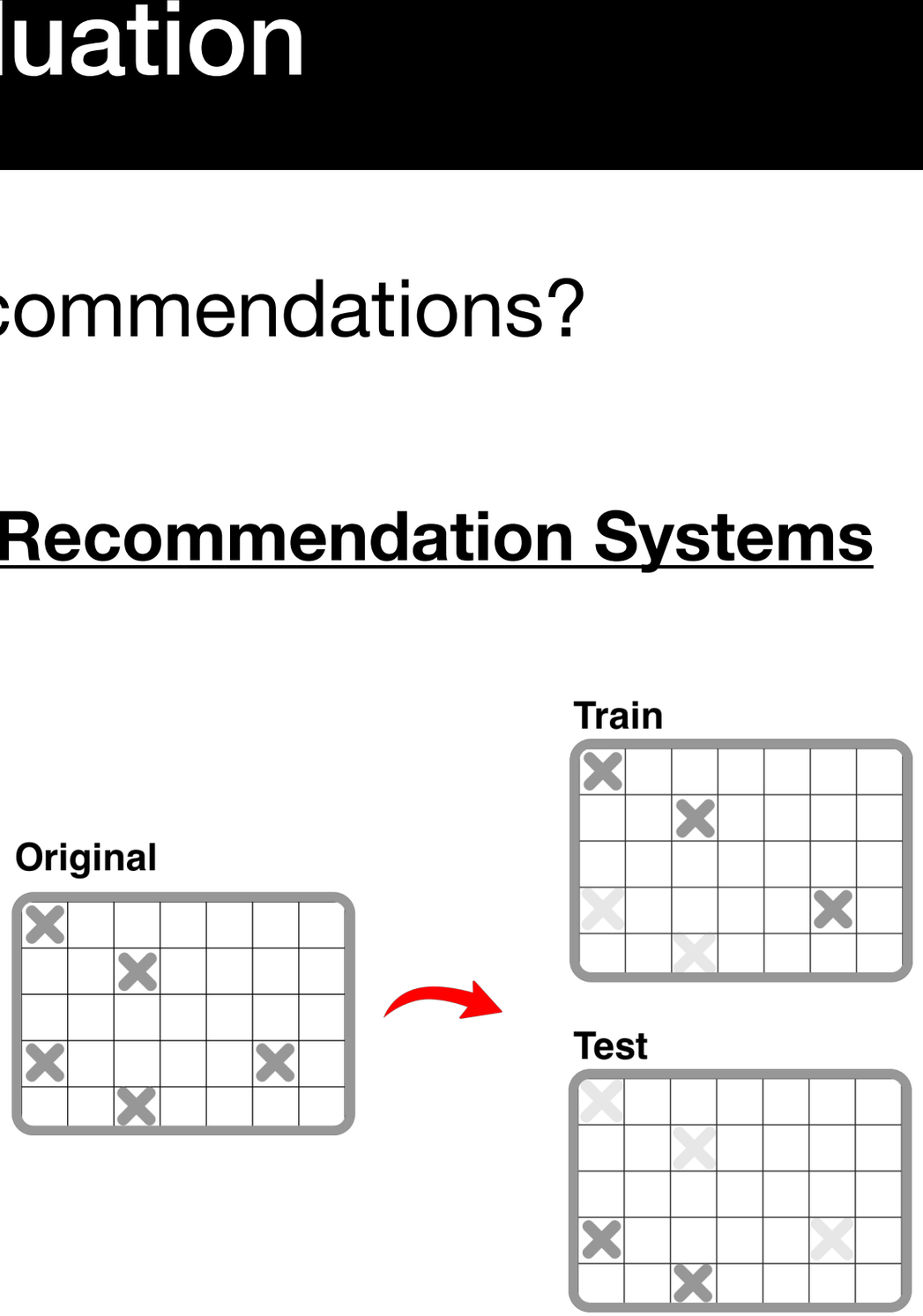
Step 5: Evaluation

How do we evaluate recommendations?

Traditional ML



Recommendation Systems



Pre-processing | Hyperparameter Tuning | Model Training | Post-processing | Evaluation

Step 5: Evaluation

Metrics

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{N}(y - \hat{y})^2}{N}}$$

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Step 5: Evaluation

Precision@K
Of the top k recommendations, what proportion are actually "relevant"?

Recall@K
Proportion of items that were found in the top k recommendations.

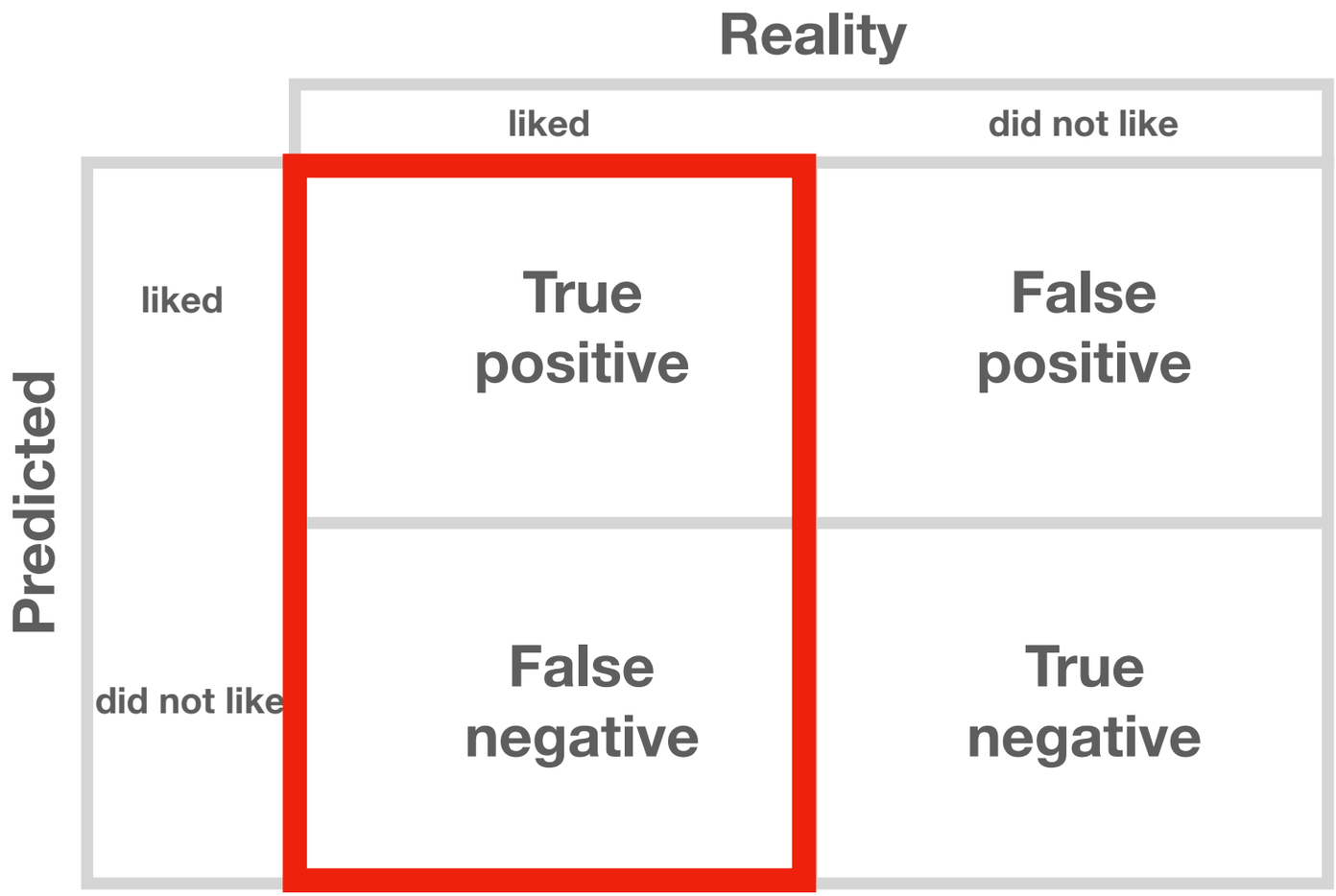
Predicted \ Reality	liked	did not like
liked	True positive	False positive
did not like	False negative	True negative

$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

Step 5: Evaluation

Precision@K
Of the top k recommendations, what proportion are actually "relevant"?

Recall@K
Proportion of items that were found in the top k recommendations.



$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

Important Considerations

- Interpretability
- Efficiency and scalability
- Diversity
- Serendipity

Python Tools

- `import surprise (@NicolasHug)`
- `import implicit (@benfred)`
- `import LightFM  (@lyst)`
- `import pyspark.mlib.recommendation`

Thank you!

Jill Cates
Data Scientist at BioSymetrics
github: @topspinj
cates.jill@gmail.com

Cross-validation

- Prevent overfitting with k-fold cross-validation