Dealing with sparsity

BUILDING RECOMMENDATION ENGINES IN PYTHON

Rob O'Callaghan
Director of Data

Sparse matrices

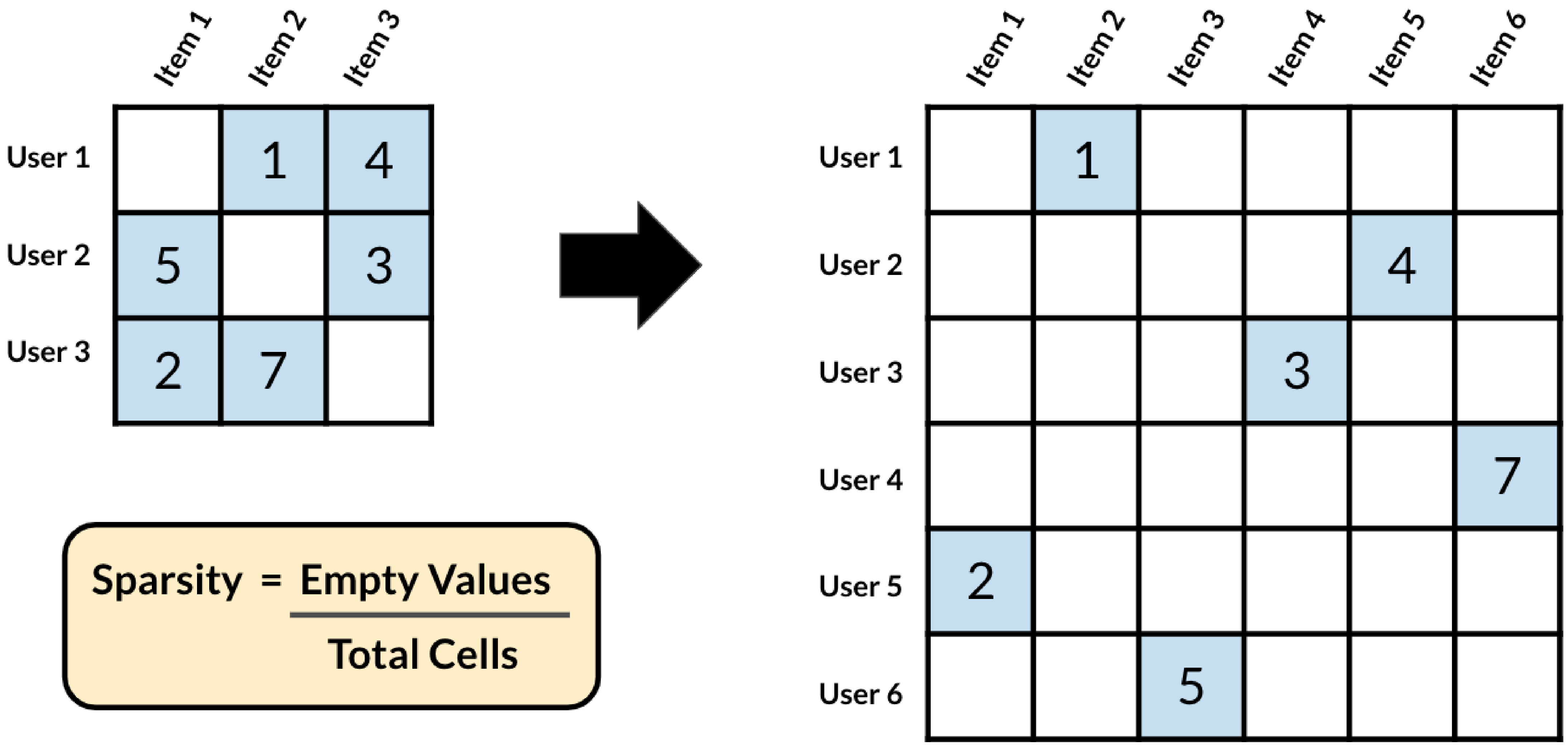
	Item 1	Item 2	Item 3
User 1		1	4
User 2	5		3
User 3	2	7	

Sparse matrices

	Item 1	Item 2	Item 3
User 1		1	4
User 2	5		3
User 3	2	7	

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1		1				
User 2					4	
User 3				3		
User 4						7
User 5	2					
User 6			5			

Sparse matrices



Measuring sparsity

```
print(book_rating_df)
```

```
title       The Great Gatsby    The Catcher in the Rye    Fifty Shades of Grey
User
User_233                 3.0                       NaN                     NaN
User_651                 NaN                       5.0                     4.0
User_965                 4.0                       3.0                     NaN
     ...                 ...                       ...                     ...
```

Measuring sparsity

```
number_of_empty = book_ratings_df.isnull().values.sum()
total_number = user_ratings_df.size
sparsity = number_of_empty/total_number
print(sparsity)
```

```
0.0114
```

Why sparsity matters

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1		1				
User 2					5	
User 3				3		
User 4						4
User 5	2					
User 6			5		1	

Why sparsity matters

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1		1			?	
User 2					5	
User 3				3		
User 4						4
User 5	2					
User 6			5		1	

Why sparsity matters

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1		1			?	
User 2					5	
User 3				3		
User 4						4
User 5	2					
User 6			5		1	

The only other ratings

Why sparsity matters

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1		1			3	
User 2					5	
User 3				3		
User 4						4
User 5	2					
User 6			5		1	

The only other ratings **mean = 3**

Measuring sparsity per column

```
user_ratings_df.notnull().sum()
```

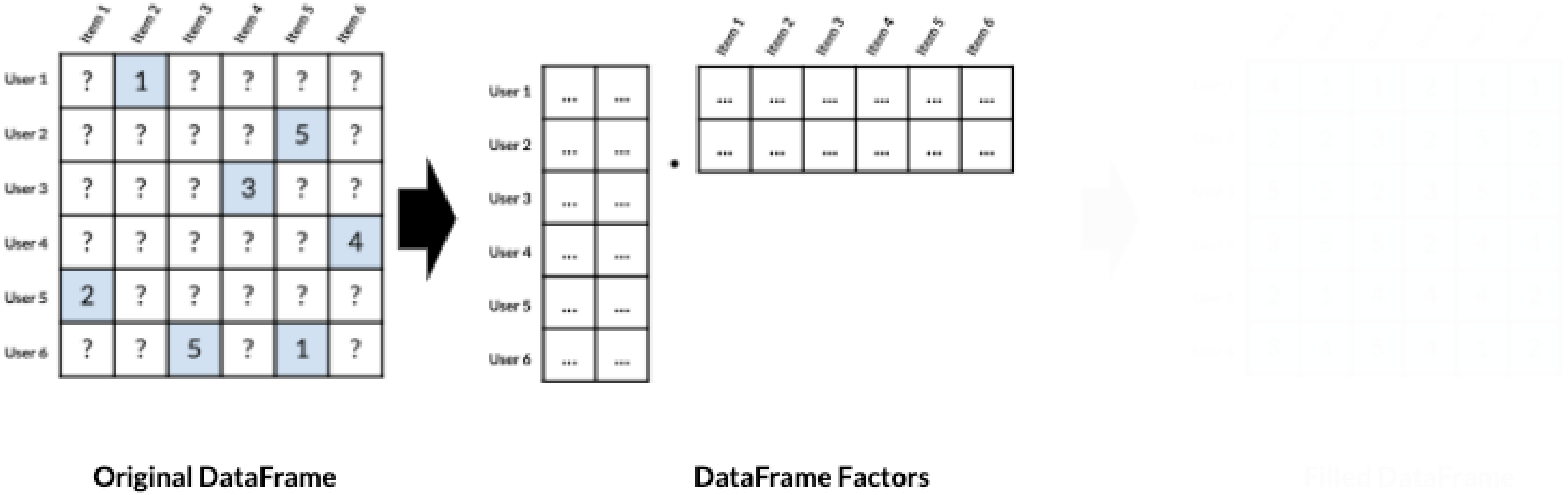
```
The Pelican Brief                          1
Snow Crash                                 1
The Great Gatsby                          12
Fifty Shades of Grey                       9
Leviathan                                  1
                                          ..
```

Matrix factorization

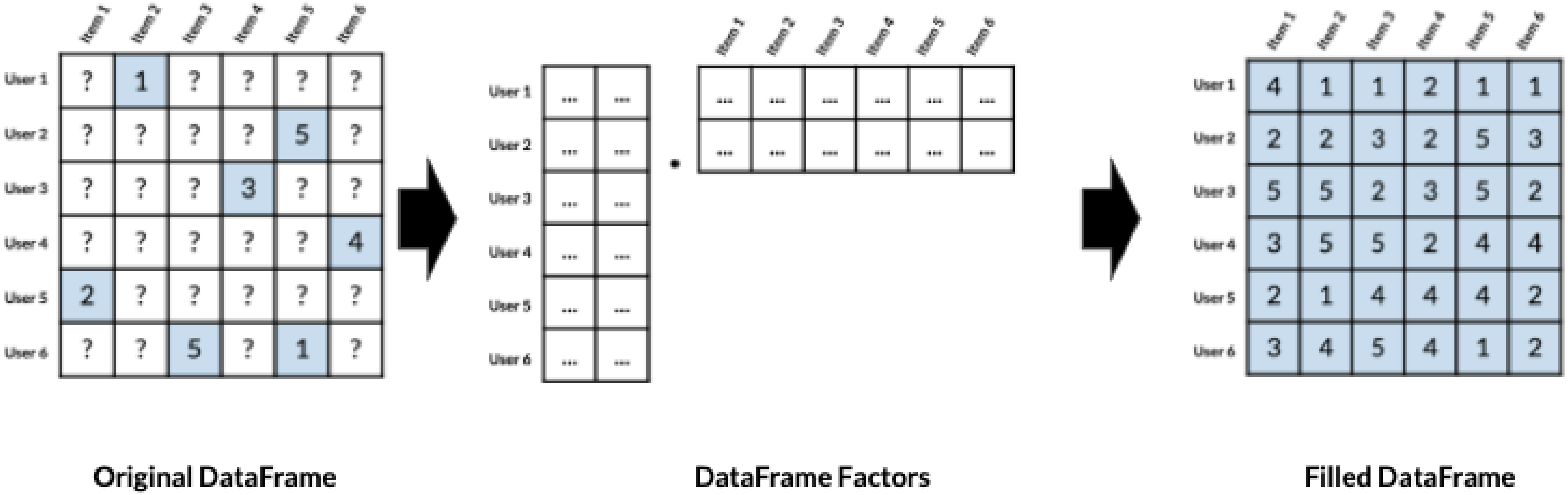
	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	?	1	?	?	?	?
User 2	?	?	?	?	5	?
User 3	?	?	?	3	?	?
User 4	?	?	?	?	?	4
User 5	2	?	?	?	?	?
User 6	?	?	5	?	1	?

Original DataFrame

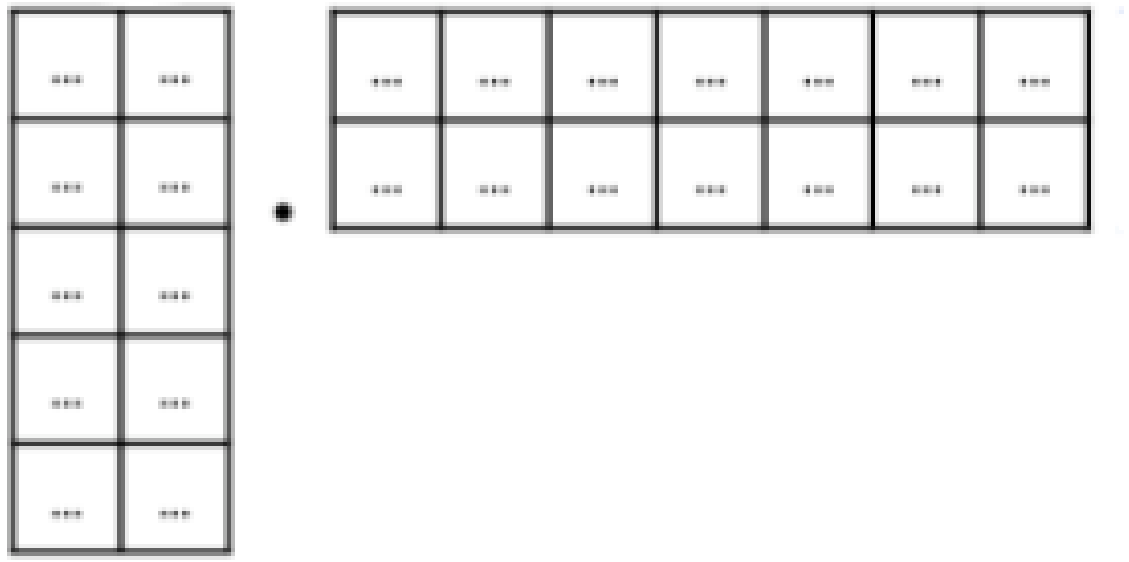
Matrix factorization



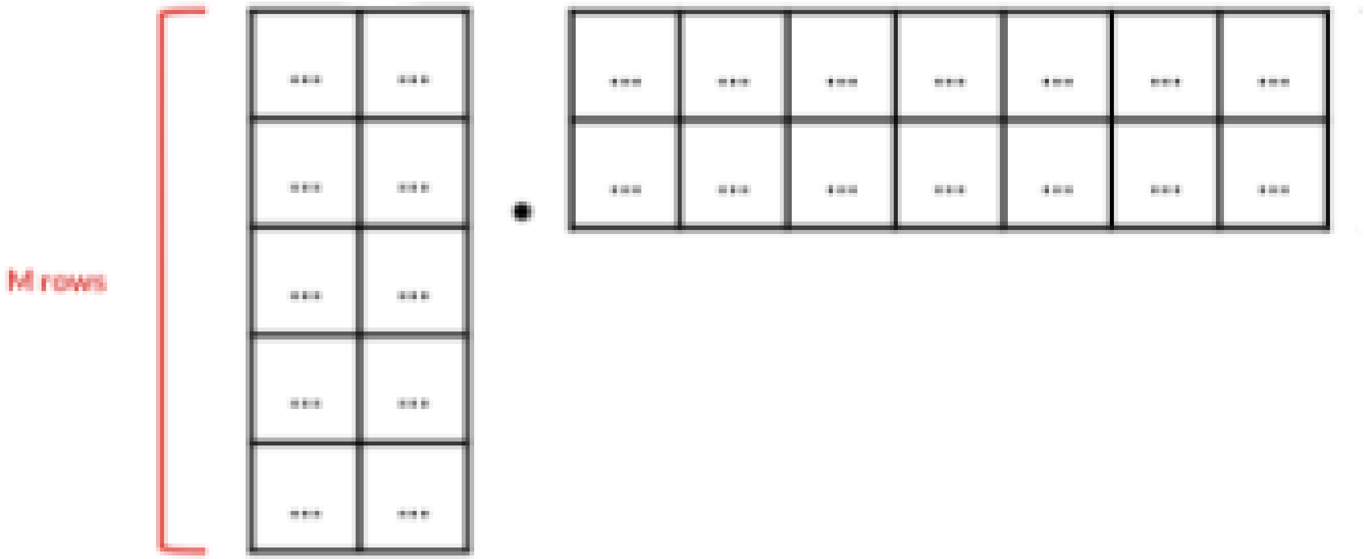
Matrix factorization



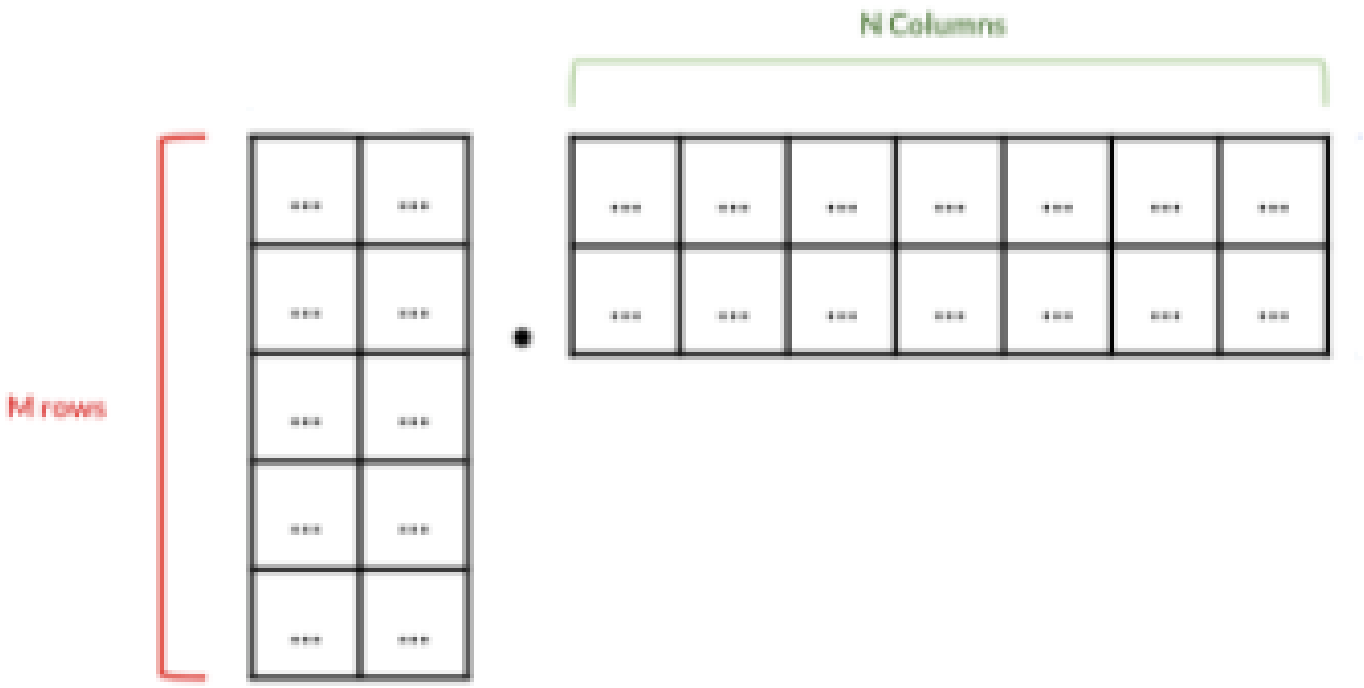
Matrix multiplication



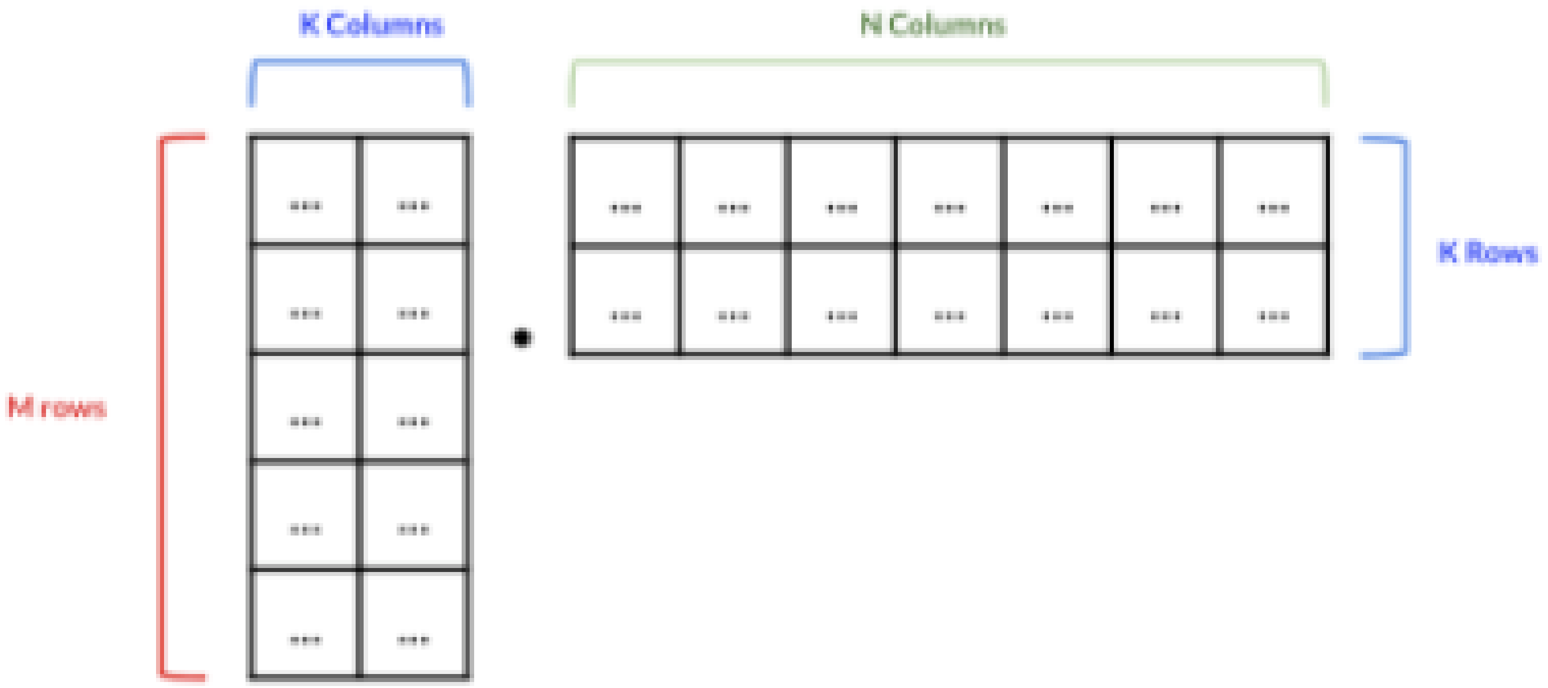
Matrix multiplication



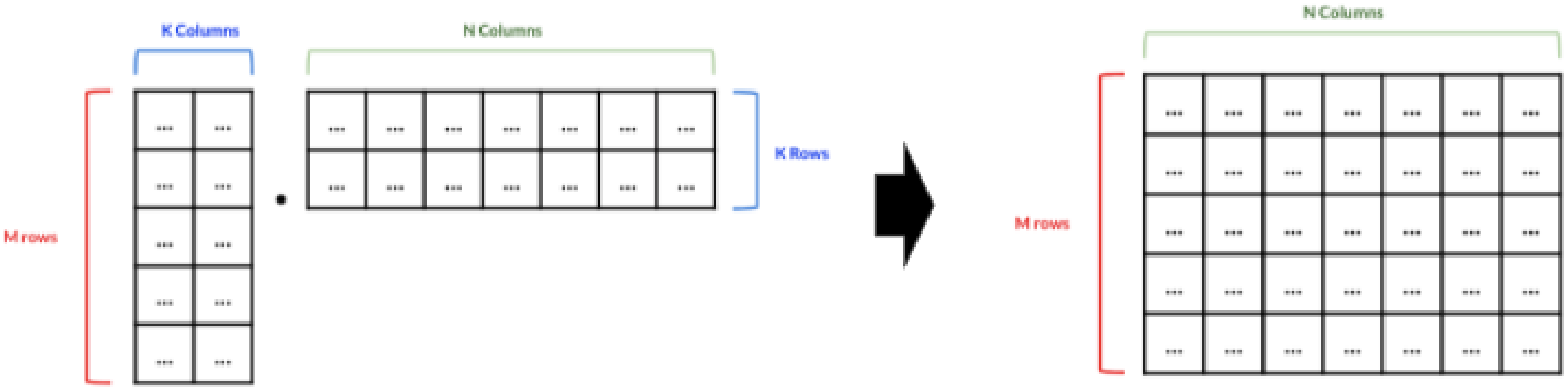
Matrix multiplication



Matrix multiplication



Matrix multiplication



Matrix multiplication

```
print(matrix_x)
```

```
[[4, 1],
 [2, 2],
 [3, 3]]
```

```
print(matrix_b)
```

```
[[1, 0, 4],
 [0, 1, 6]]
```

Matrix multiplication

```python
import numpy as np

dot_product = np.dot(matrix_x, matrix_b)
print(dot_product)
```

```
[[ 4  1 22]
 [ 2  2 20]
 [ 3  3 30]]
```

Let's practice!

BUILDING RECOMMENDATION ENGINES IN PYTHON

Matrix factorization

BUILDING RECOMMENDATION ENGINES IN PYTHON

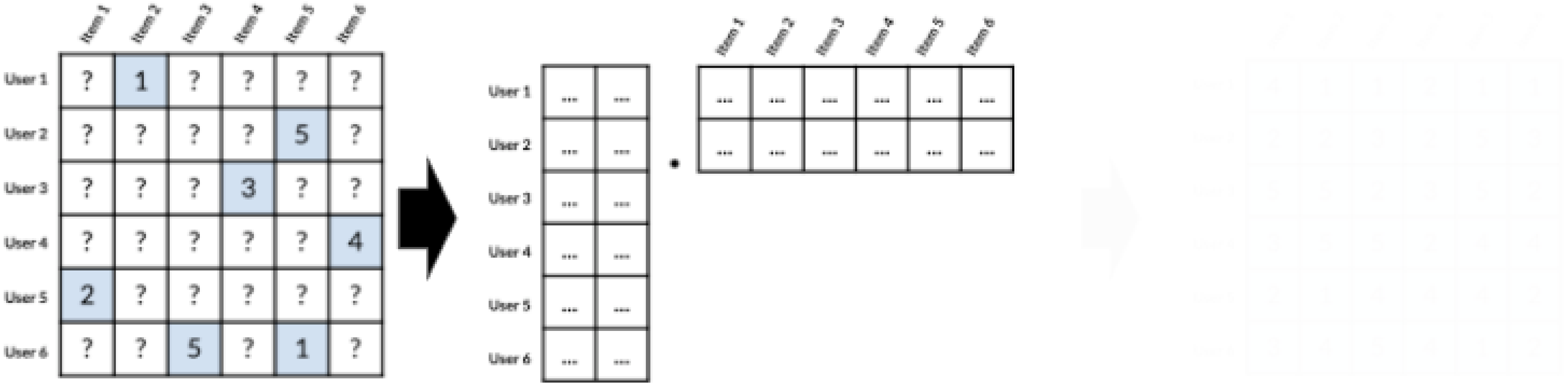
Rob O'Callaghan
Director of Data

Why this helps with sparse matrices

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	?	1	?	?	?	?
User 2	?	?	?	?	5	?
User 3	?	?	?	3	?	?
User 4	?	?	?	?	?	4
User 5	2	?	?	?	?	?
User 6	?	?	5	?	1	?

Original DataFrame

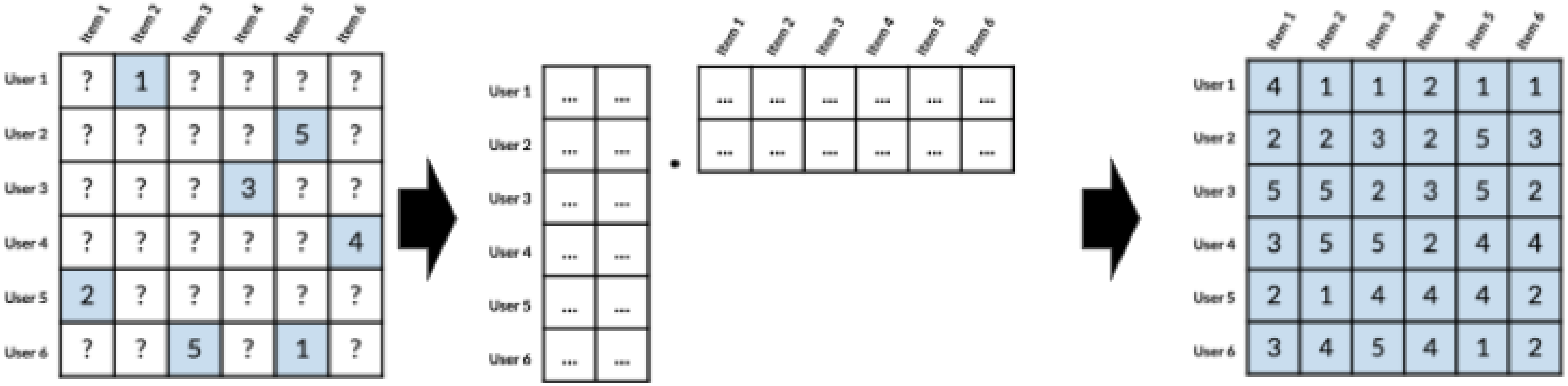
Why this helps with sparse matrices



Original DataFrame

DataFrame Factors

Why this helps with sparse matrices



Original DataFrame

DataFrame Factors

Filled DataFrame

What matrix factorization looks like

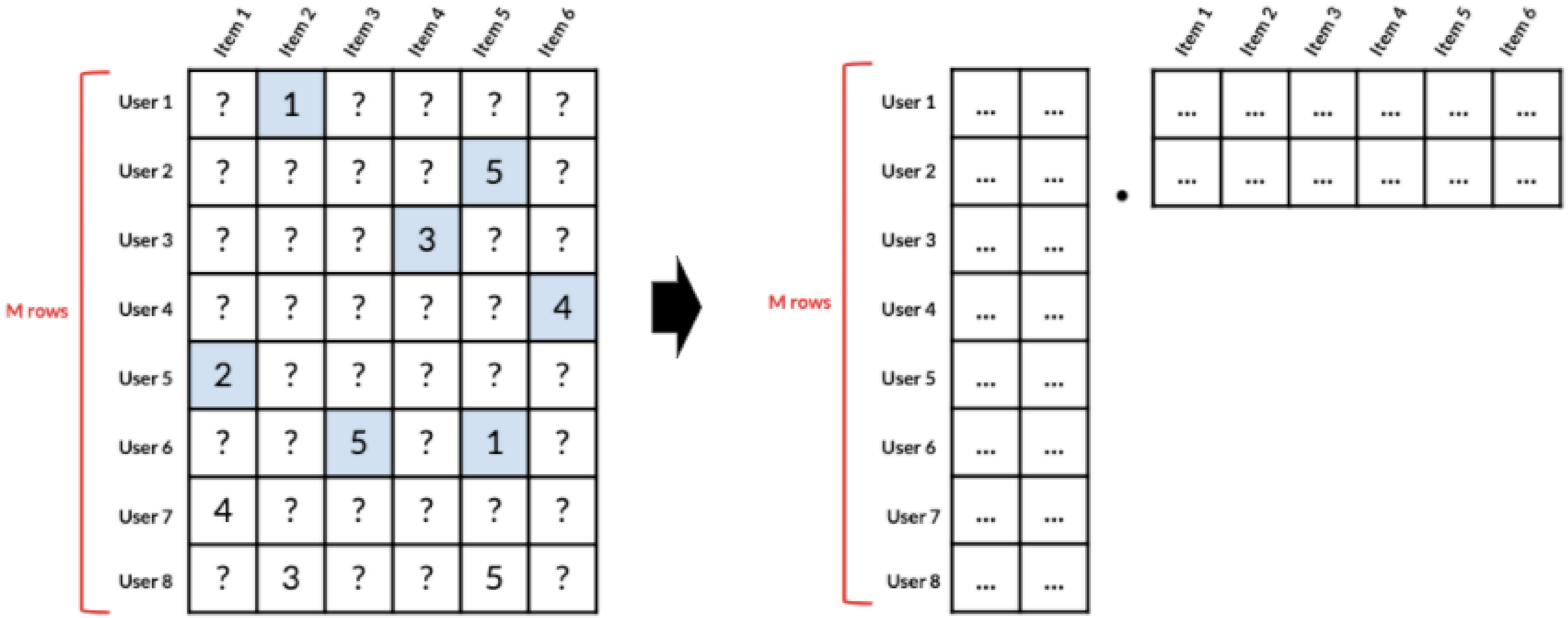
	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	?	1	?	?	?	?
User 2	?	?	?	?	5	?
User 3	?	?	?	3	?	?
User 4	?	?	?	?	?	4
User 5	2	?	?	?	?	?
User 6	?	?	5	?	1	?
User 7	4	?	?	?	?	?
User 8	?	3	?	?	5	?

User 1	...	...
User 2	...	...
User 3	...	...
User 4	...	...
User 5	...	...
User 6	...	...
User 7	...	...
User 8	...	...

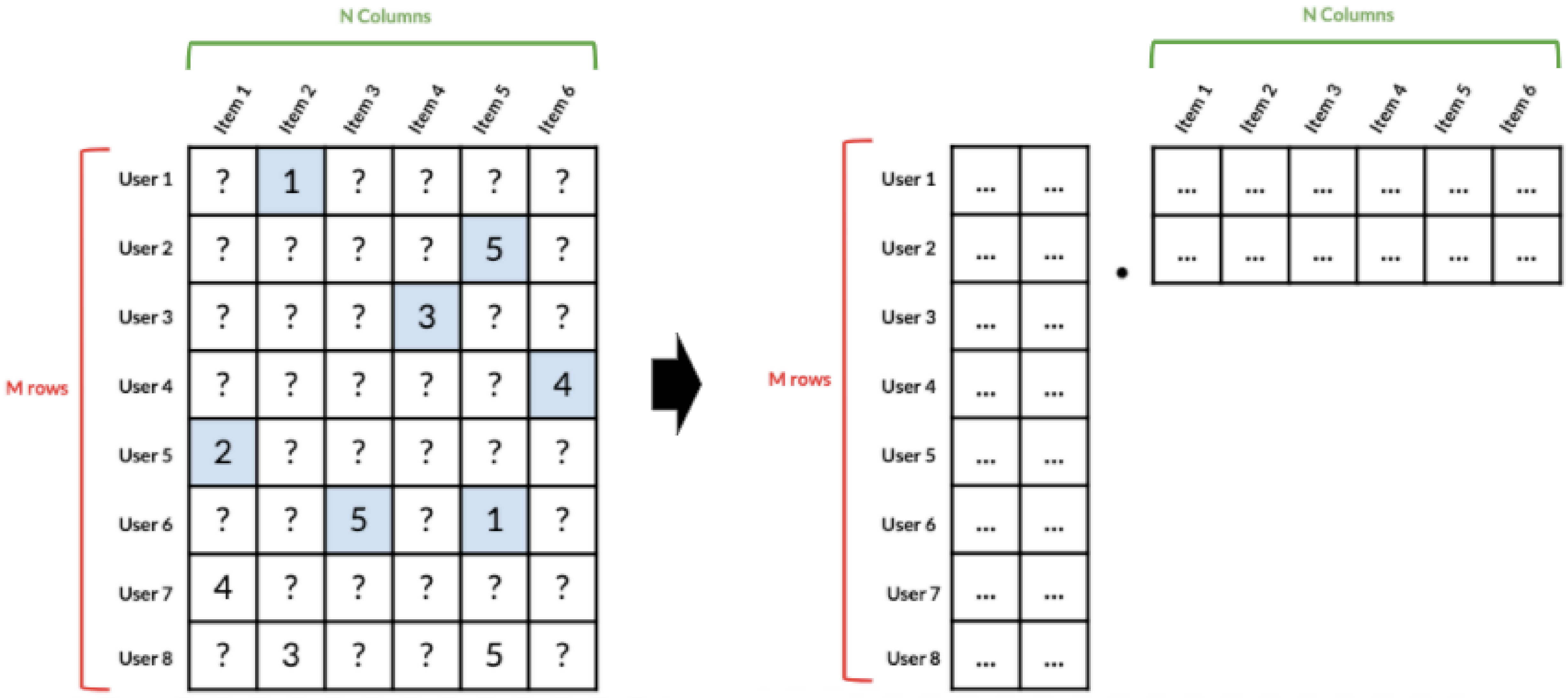
•

Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
...	...	...	...	...	...
...	...	...	...	...	...

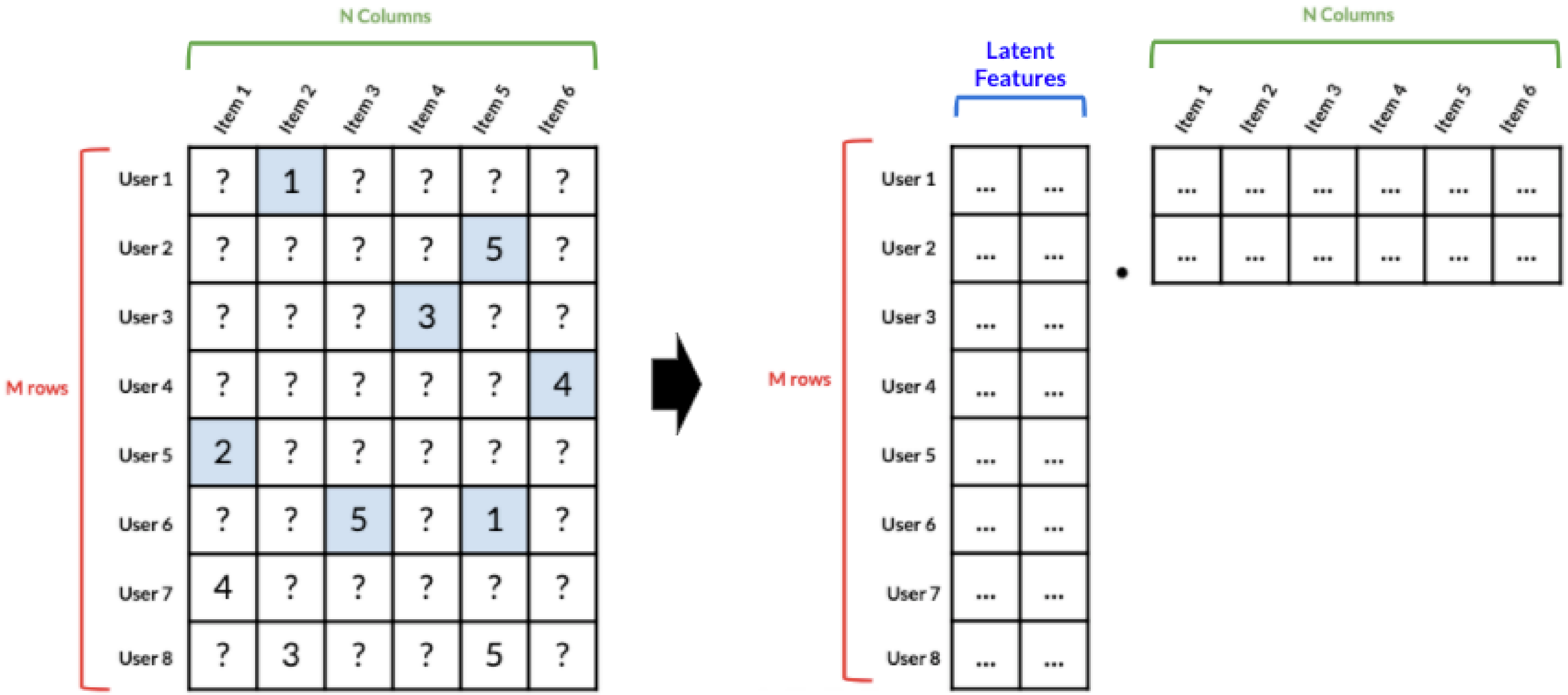
What matrix factorization looks like



What matrix factorization looks like



What matrix factorization looks like



Latent features

	Fantasy Novel 1	Fantasy Novel 2	Horror Story	Romance Book 1	Romance Book 2	Romance Book 3
User 1		5				1
User 2			3			
User 3	4				2	
User 4				5		

User 1	0.8	0.2
User 2	0.5	0.4
User 3	0.7	0.1
User 4	0.1	0.9

Fantasy Novel 1	Fantasy Novel 2	Horror Story	Romance Book 1	Romance Book 2	Romance Book 3
...	...	...	...	...	...
...	...	...	...	...	...

Latent features

	Fantasy Novel 1	Fantasy Novel 2	Horror Story	Romance Book 1	Romance Book 2	Romance Book 3
User 1		5				1
User 2			3			
User 3	4				2	
User 4				5		

	Latent Feature 1	
User 1	0.8	0.2
User 2	0.5	0.4
User 3	0.7	0.1
User 4	0.1	0.9

•

Fantasy Novel 1	Fantasy Novel 2	Horror Story	Romance Book 1	Romance Book 2	Romance Book 3
...	...	...	...	...	...
...	...	...	...	...	...

Latent features

	Fantasy Novel 1	Fantasy Novel 2	Horror Story	Romance Book 1	Romance Book 2	Romance Book 3
User 1		5				1
User 2			3			
User 3	4				2	
User 4				5		

	Latent Feature 1	Latent Feature 2
User 1	0.8	0.2
User 2	0.5	0.4
User 3	0.7	0.1
User 4	0.1	0.9

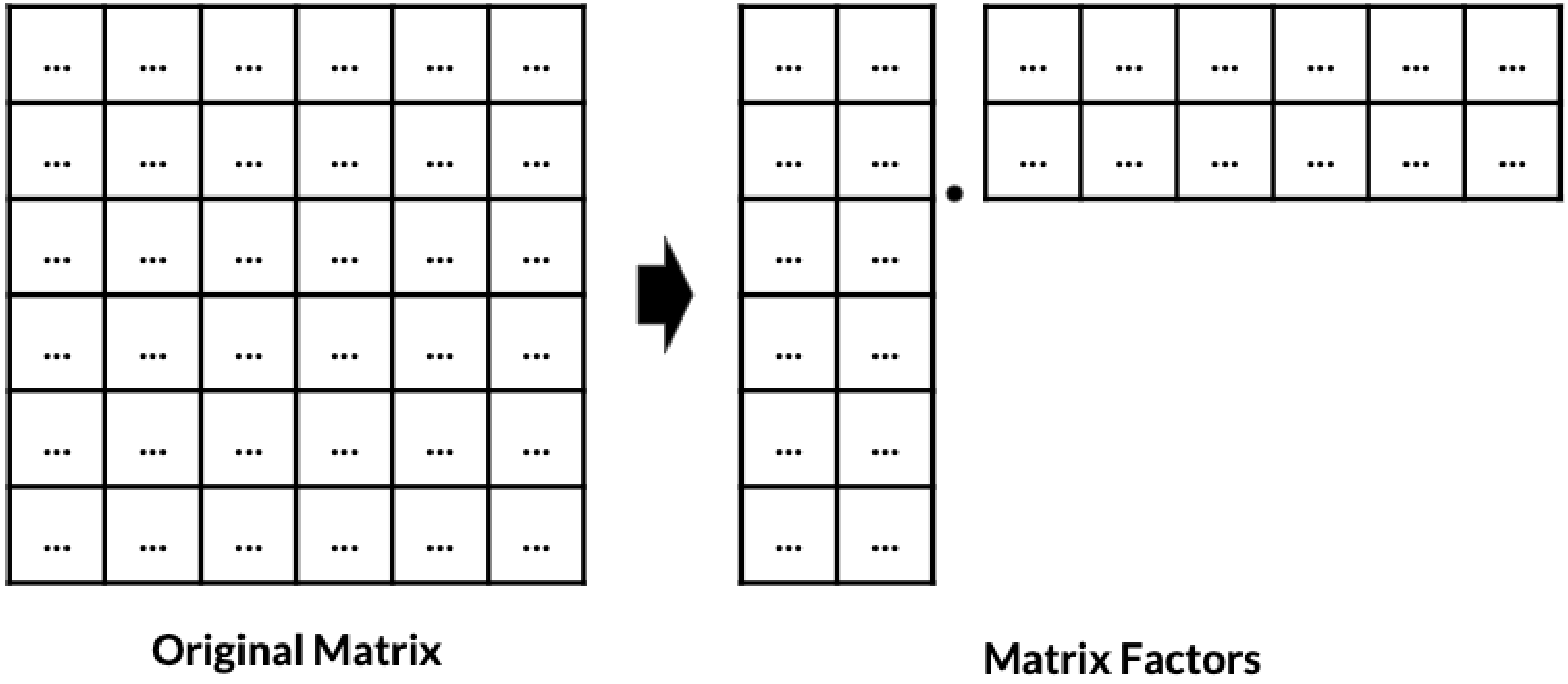
Fantasy Novel 1	Fantasy Novel 2	Horror Story	Romance Book 1	Romance Book 2	Romance Book 3
...	...	...	...	...	...
...	...	...	...	...	...

Information loss

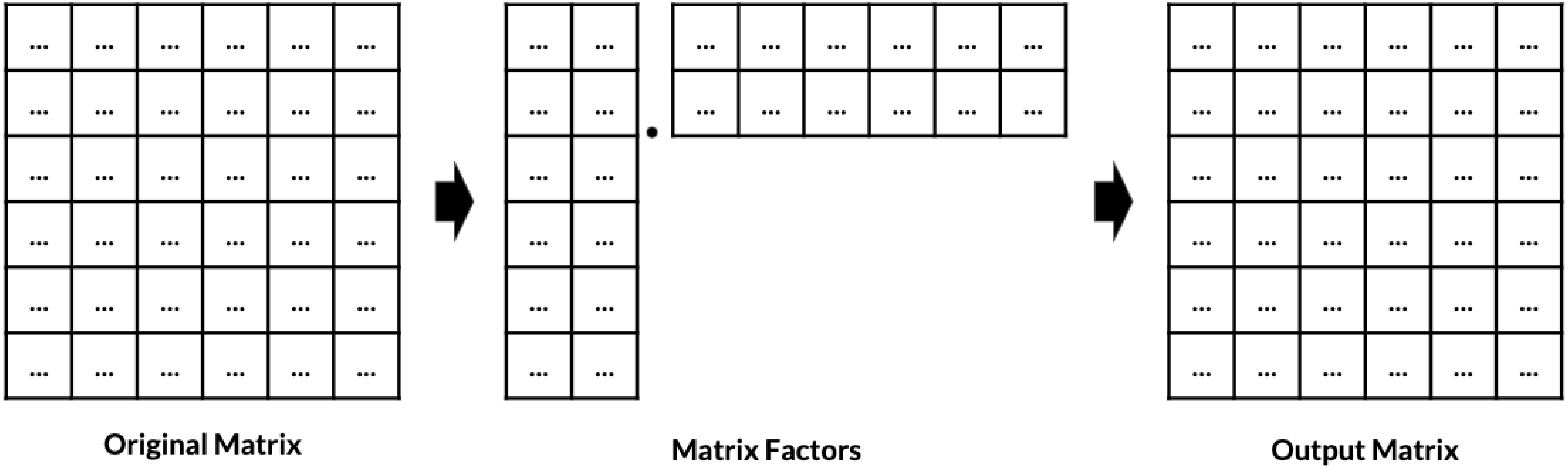
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...

Original Matrix

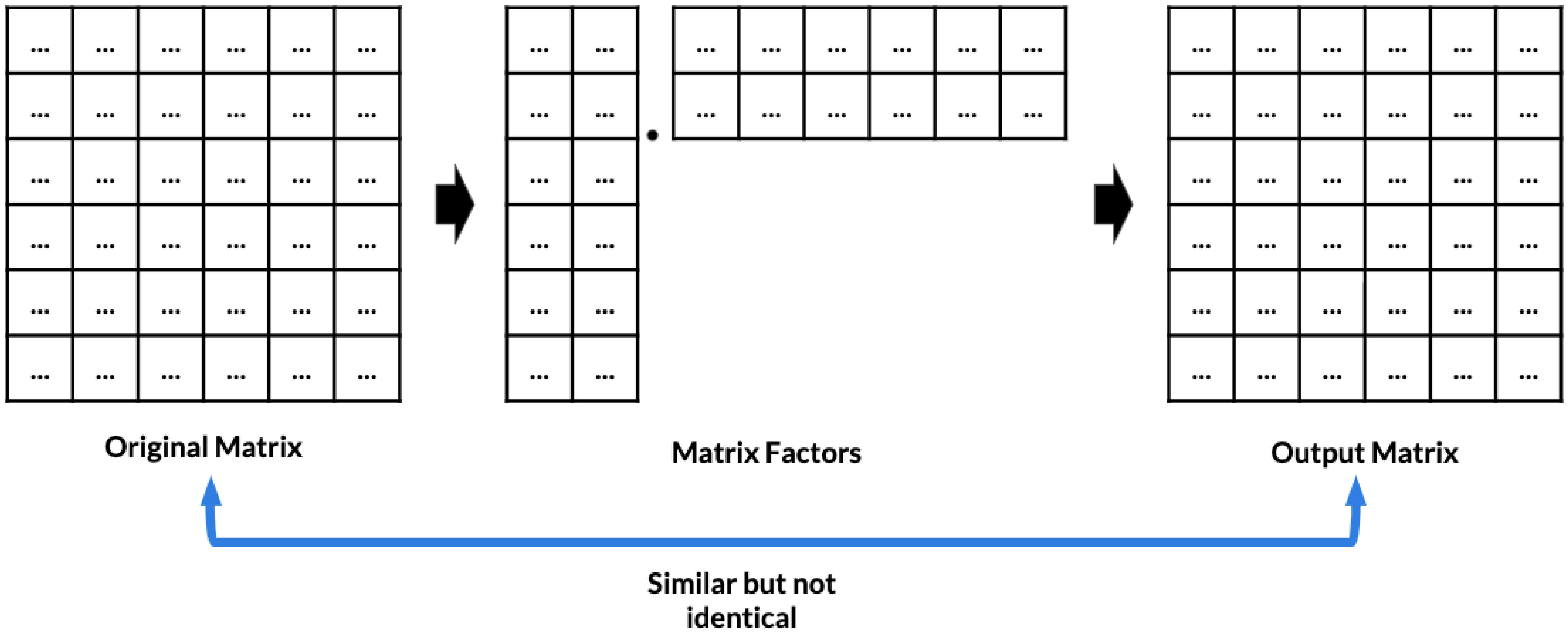
Information loss



Information loss



Information loss



Let's practice!

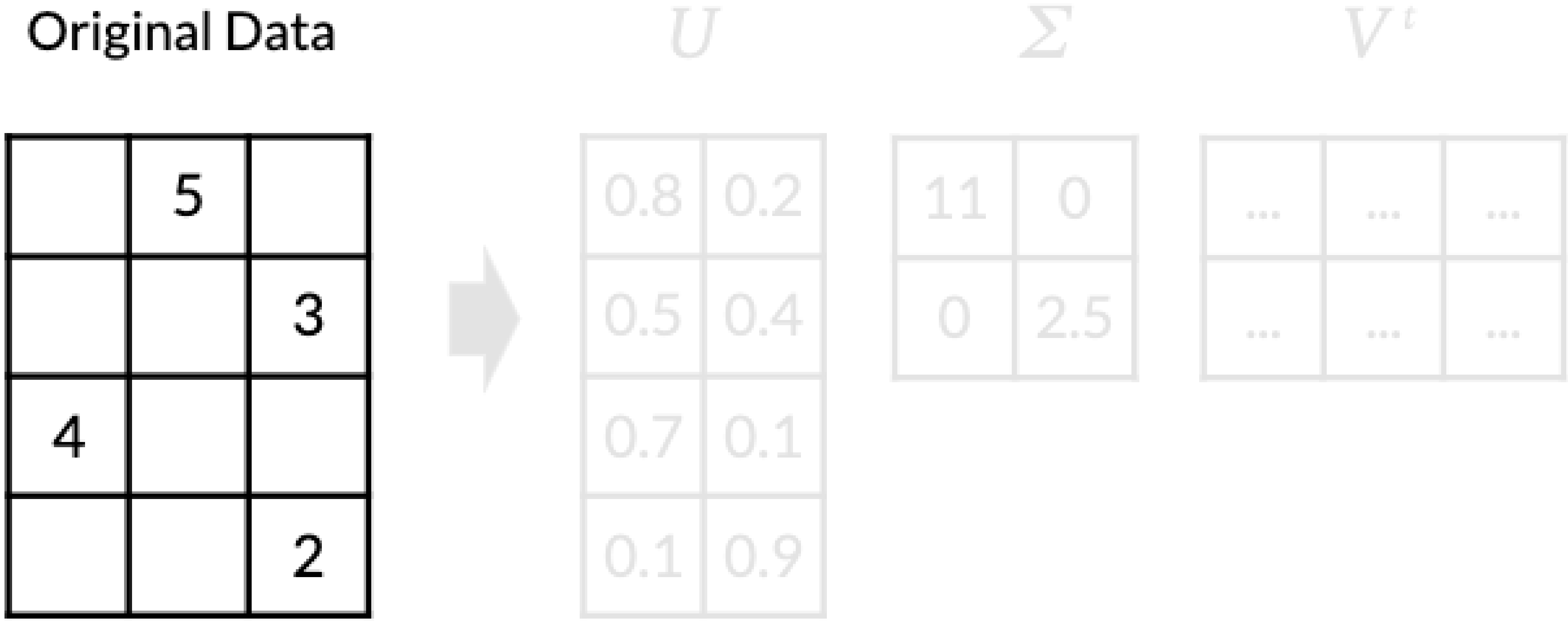
BUILDING RECOMMENDATION ENGINES IN PYTHON

Singular value decomposition (SVD)

BUILDING RECOMMENDATION ENGINES IN PYTHON

Rob O'Callaghan
Director of Data

What SVD does



What SVD does

Original Data

	5	
		3
4		
		2

U

0.8	0.2
0.5	0.4
0.7	0.1
0.1	0.9

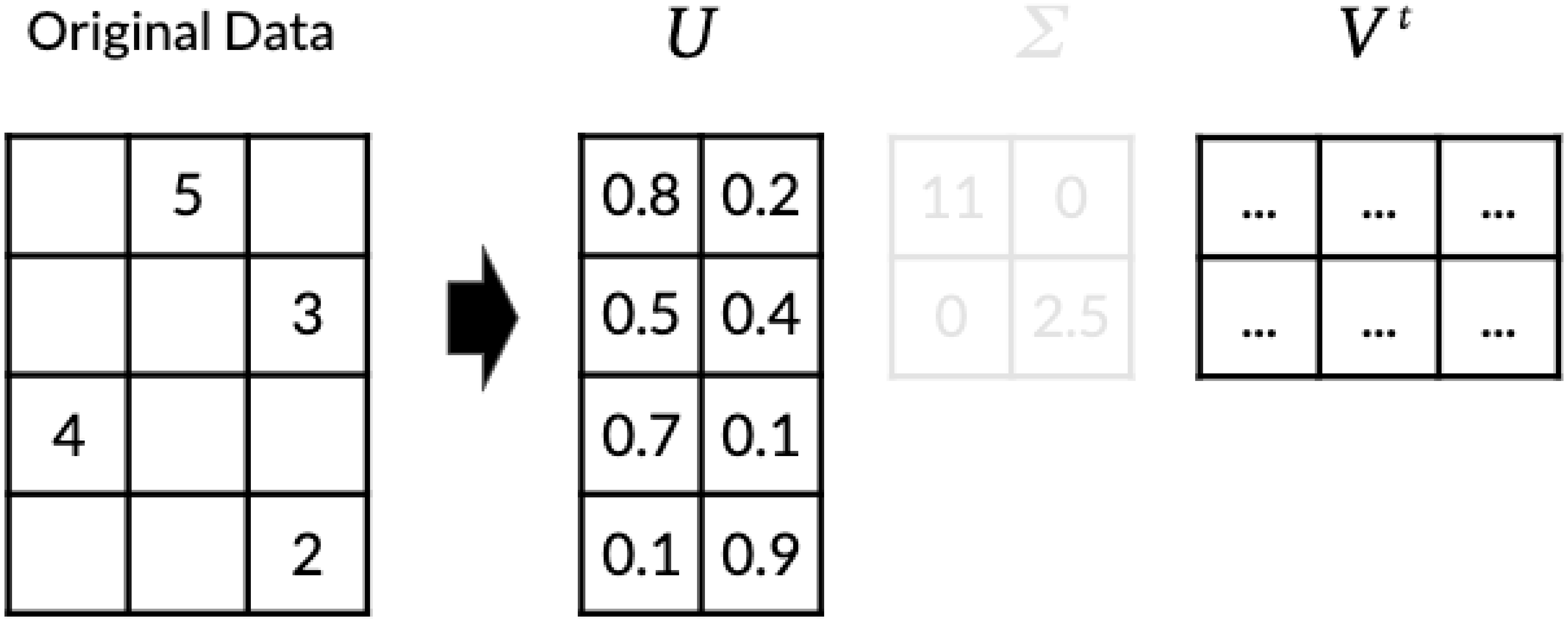
Σ

11	0
0	2.5

V^t

...	...	...
...	...	...

What SVD does



What SVD does

Original Data

	5	
		3
4		
		2

U

0.8	0.2
0.5	0.4
0.7	0.1
0.1	0.9

Σ

11	0
0	2.5

V^t

...	...	...
...	...	...

Prepping our data

```
print(book_ratings_df.shape)
```

```
(220, 500)
```

```
avg_ratings = book_ratings_df.mean(axis=1)
print(avg_ratings)
```

```
array([[4.5 ],
       [3.5],
       [2.5],
       [3.5],
        ...
       [2.2]])
```

Prepping our data

```
user_ratings_pivot_centered = user_ratings_df.sub(avg_ratings, axis=0)
user_ratings_df.fillna(0, inplace=True)
print(user_ratings_df)
```

```
          The Great Gatsby    The Catcher in the Rye    Fifty Shades of Grey
User_233               0.0                       0.0                     0.0
User_651               0.0                       0.5                    -0.5
User_965               0.5                      -0.5                     0.0
     ...               ...                       ...                     ...
```

Applying SVD

```python
from scipy.sparse.linalg import svds
U, sigma, Vt = svds(user_ratings_pivot_centered)
```

```python
print(U.shape)
```

```
(610, 6)
```

```python
print(Vt.shape)
```

```
(6, 1000)
```

Applying SVD

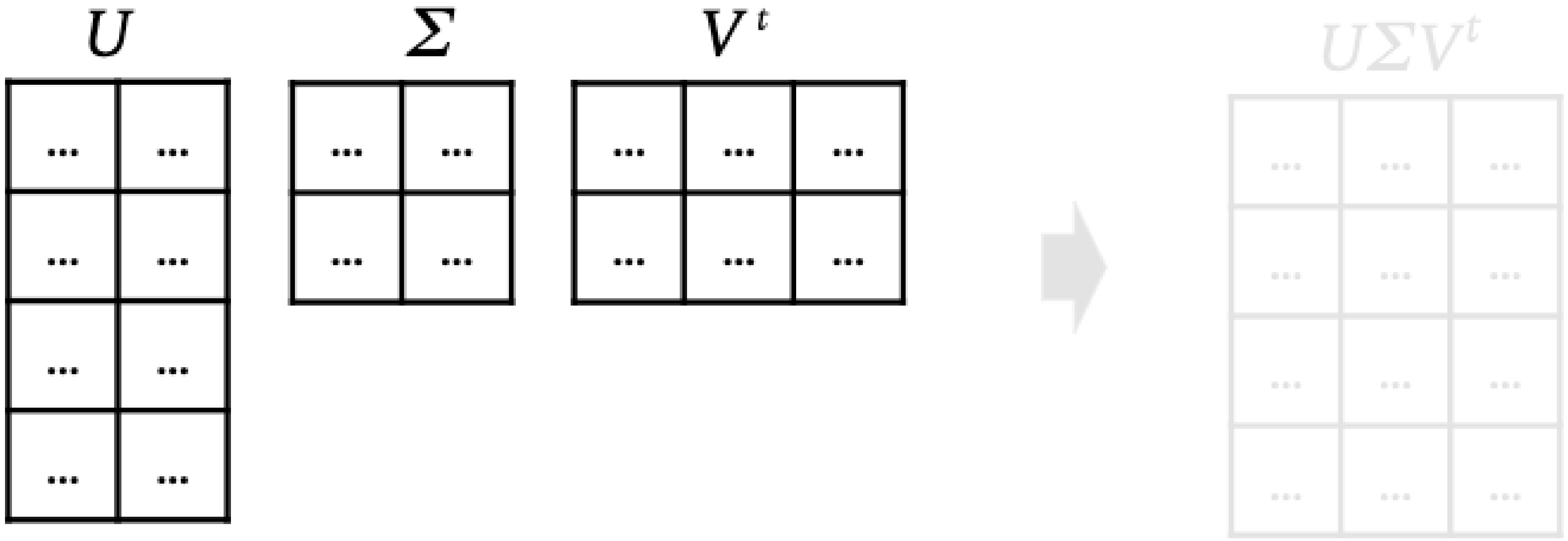
```
print(sigma)
```

```
[3.0, 4.8, -12.6, -3.8, 8.2, 7.3]
```

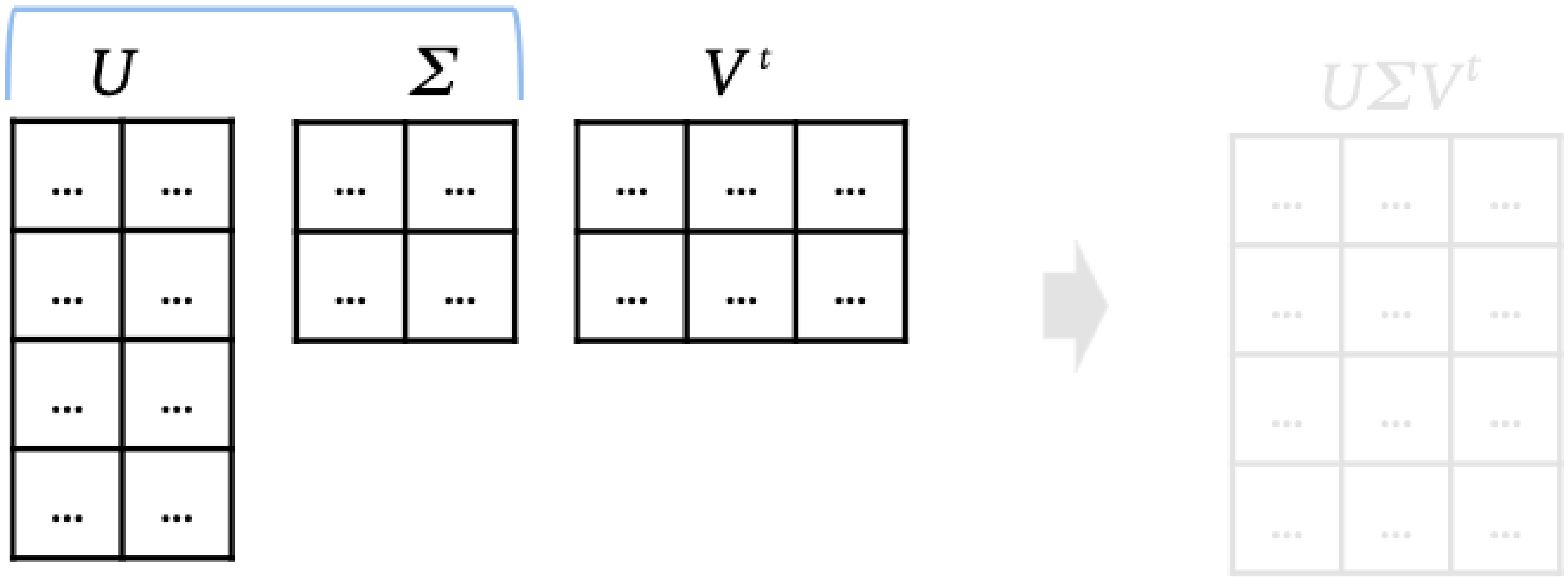
```
sigma = np.diag(sigma)
print(sigma)
```

```
array([   3.0    ,   0.     ,   0.     ,   0.     ,   0.     ,   0.     ],
       [  0.     ,   4.8    ,   0.     ,   0.     ,   0.     ,   0.     ],
       [  0.     ,   0.     , -12.6    ,   0.     ,   0.     ,   0.     ],
       [  0.     ,   0.     ,   0.     ,  -3.8    ,   0.     ,   0.     ],
       [  0.     ,   0.     ,   0.     ,   0.     ,   8.2    ,   0.     ],
       [  0.     ,   0.     ,   0.     ,   0.     ,   0.     ,   7.3    ]),
```

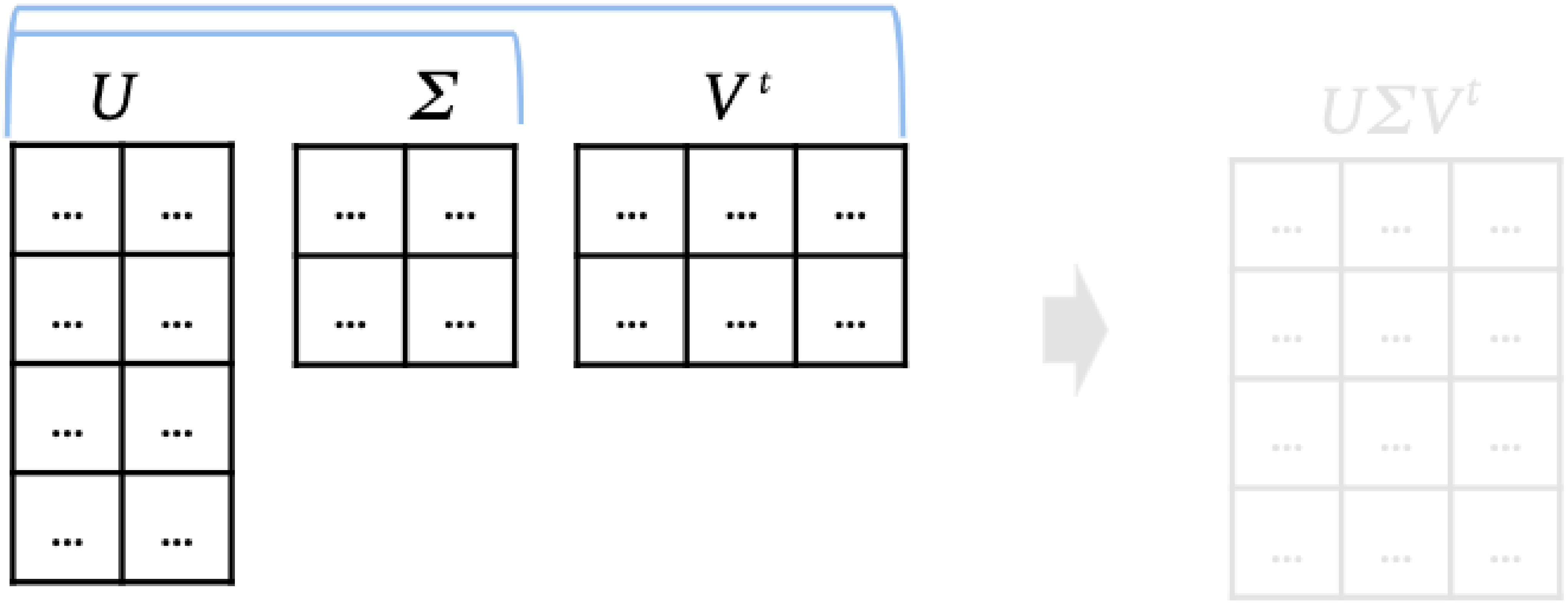
Getting the final matrix



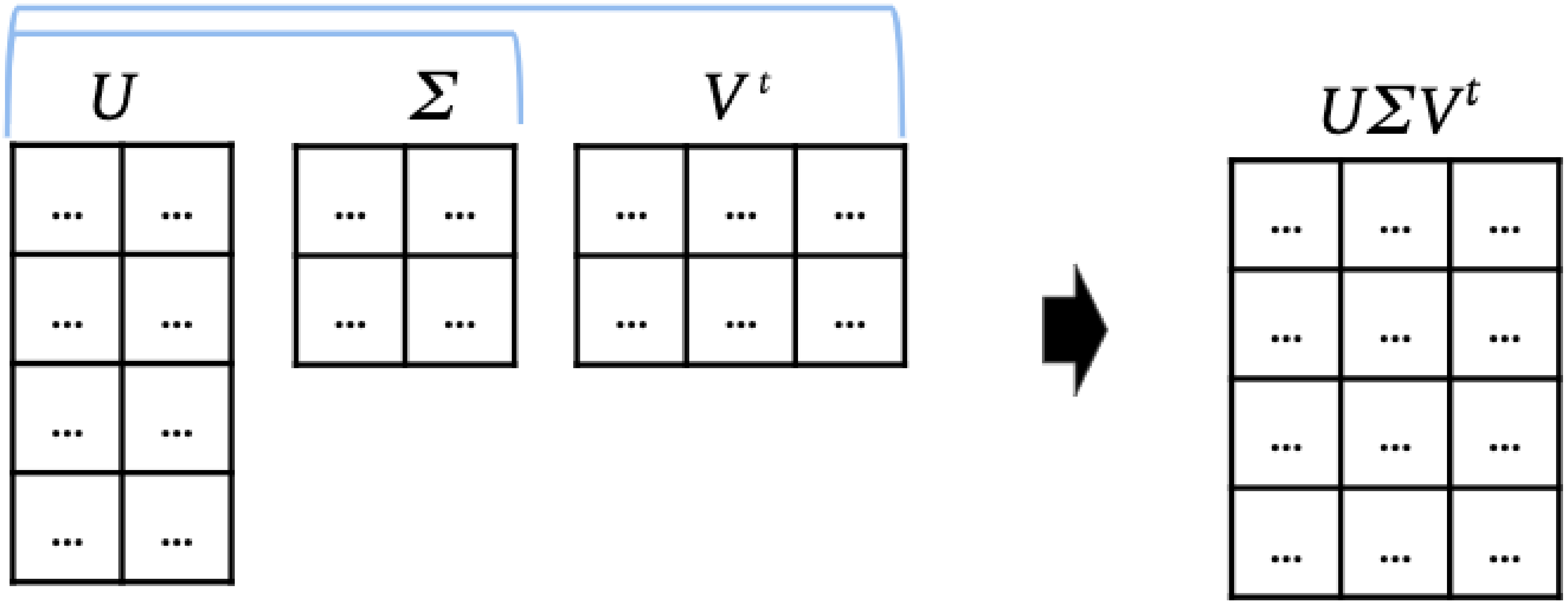
Getting the final matrix



Getting the final matrix



Getting the final matrix



Calculating the product in Python

```
recalculated_ratings =        np.dot(U, sigma)
```

Calculating the product in Python

```
recalculated_ratings = np.dot(np.dot(U, sigma), Vt)
print(recalculated_ratings)
```

```
[[  0.1      -0.9       -3.6.     ...   ]
 [ -2.3       0.5       -0.5      ...   ]
 [  0.5      -0.5        2.0      ...   ]
 [ ...        ...        ...      ...   ]]
```

Add averages back

```python
recalculated_ratings = recalculated_ratings + avg_ratings.values.reshape(-1, 1)
print(recalculated_ratings)
```

```
[[  4.6         3.6          0.9        ...    ]
 [  1.8         4.0          3.0        ...    ]
 [  3.0         2.0          4.5        ...    ]
 [ ...          ...          ...        ...    ]]
```

```python
print(book_ratings_df)
```

```
[[  5.0         4.0           NA        ...    ]
 [   NA         4.0          3.0        ...    ]
 [  3.0         2.0           NA        ...    ]
 [ ...          ...          ...        ...    ]]
```

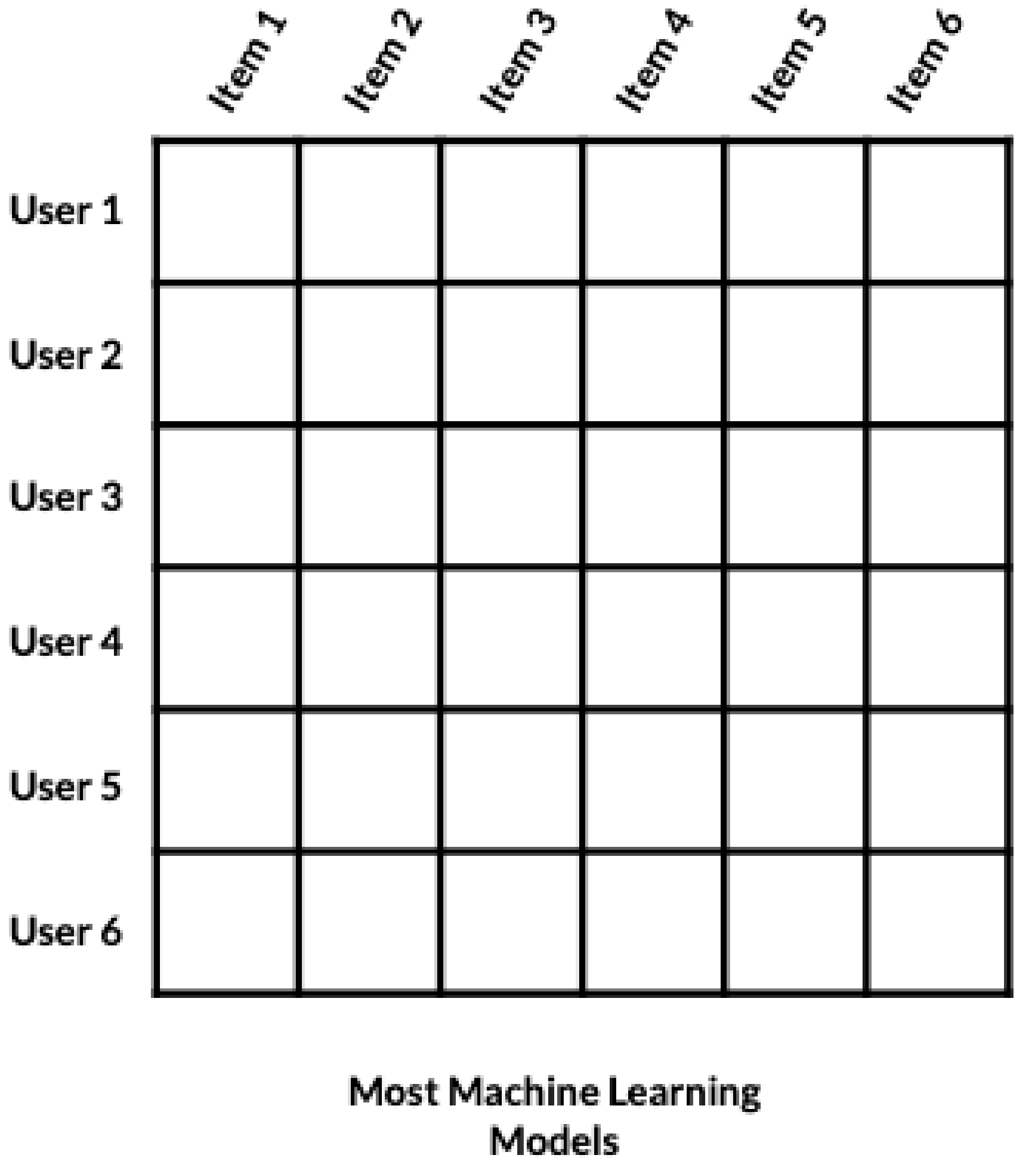
Let's practice!

Validating your predictions

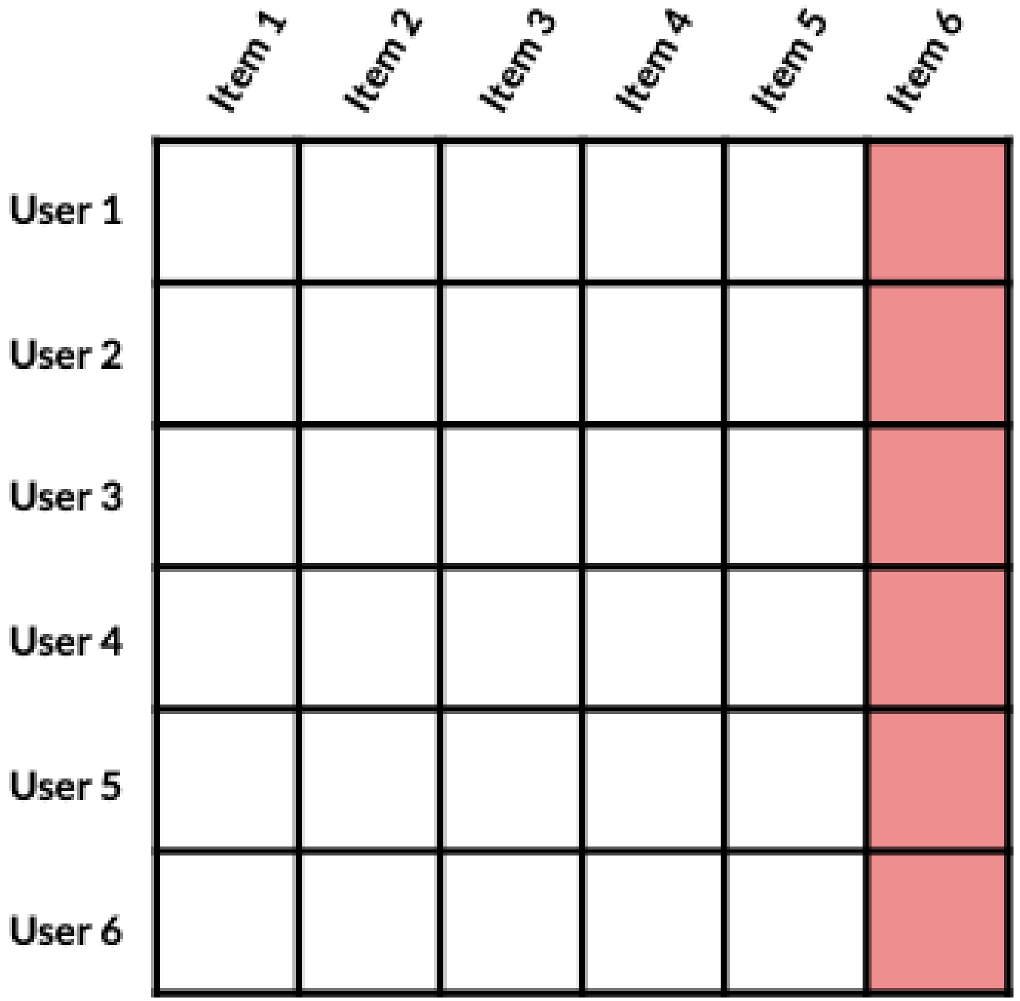
BUILDING RECOMMENDATION ENGINES IN PYTHON

Rob O'Callaghan
Director of Data

Hold-out sets



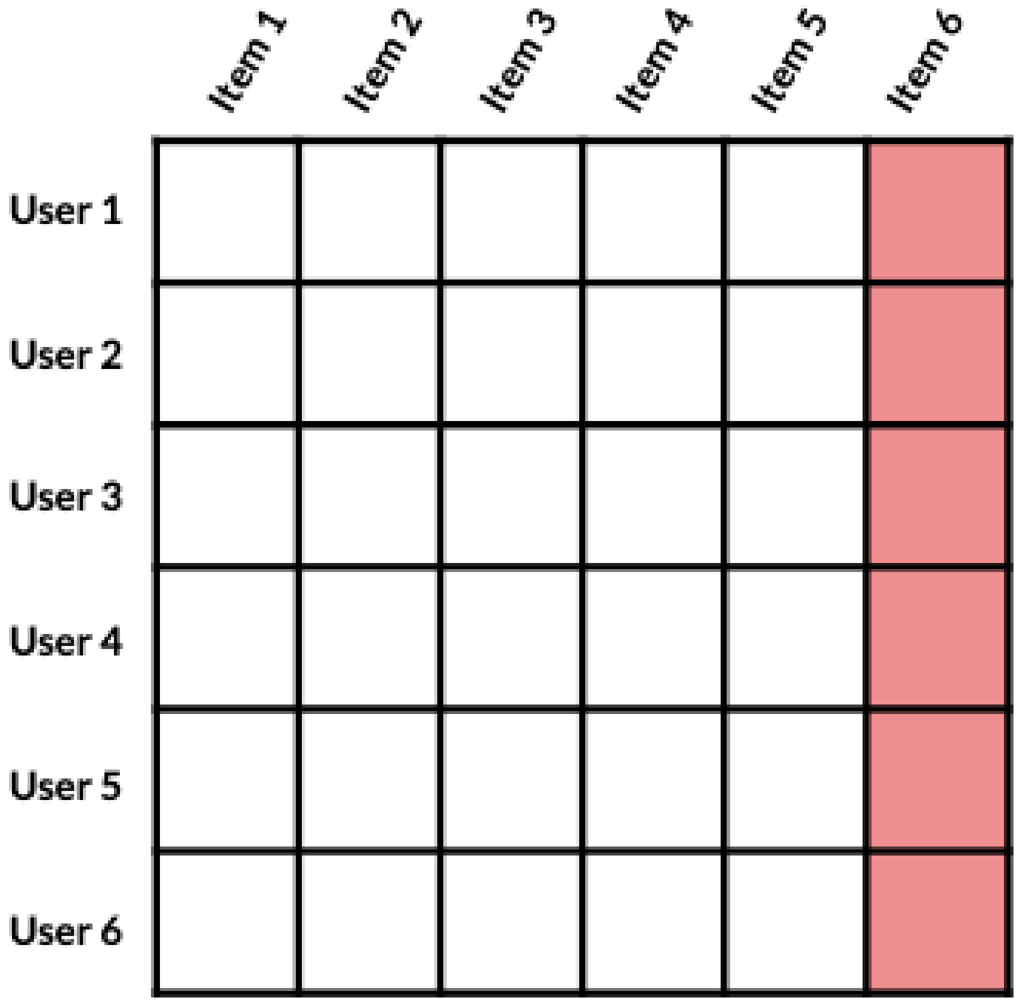
Hold-out sets



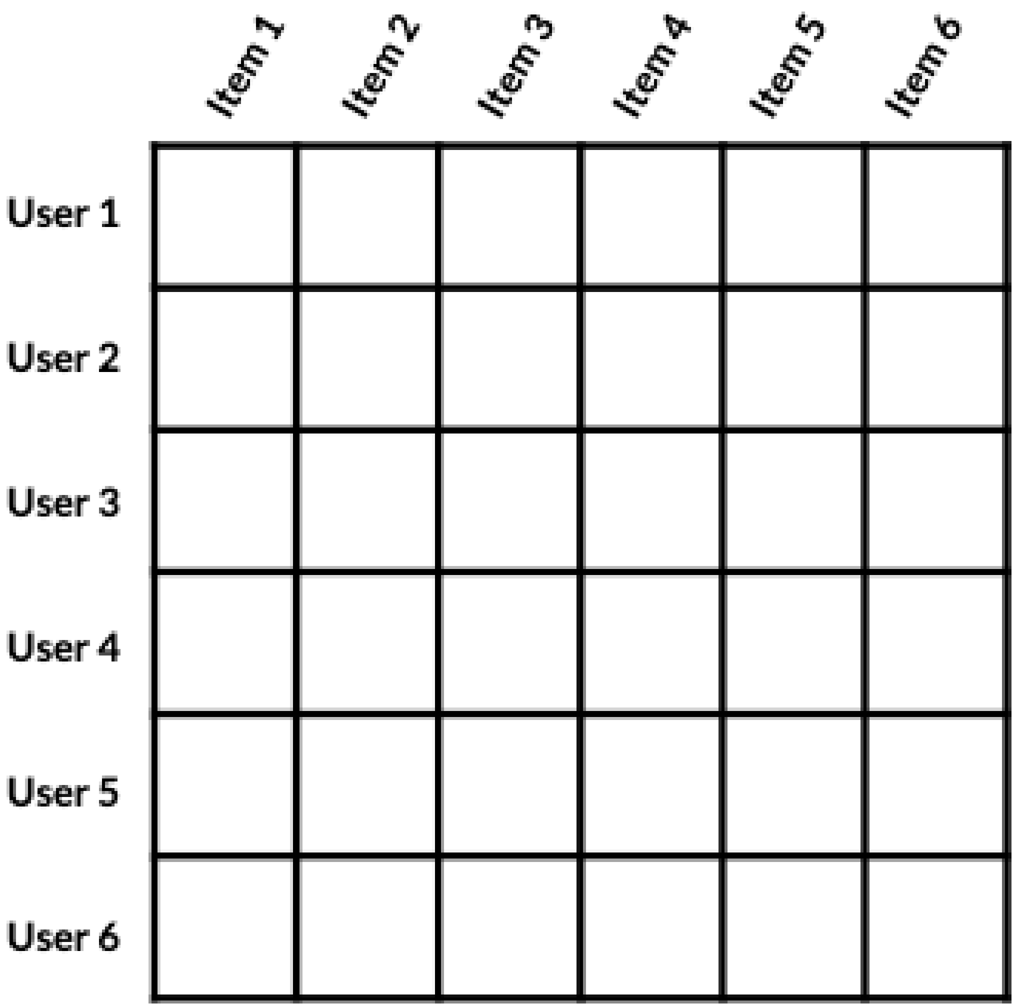
Most Machine Learning Models

Recommendation Engines

Hold-out sets

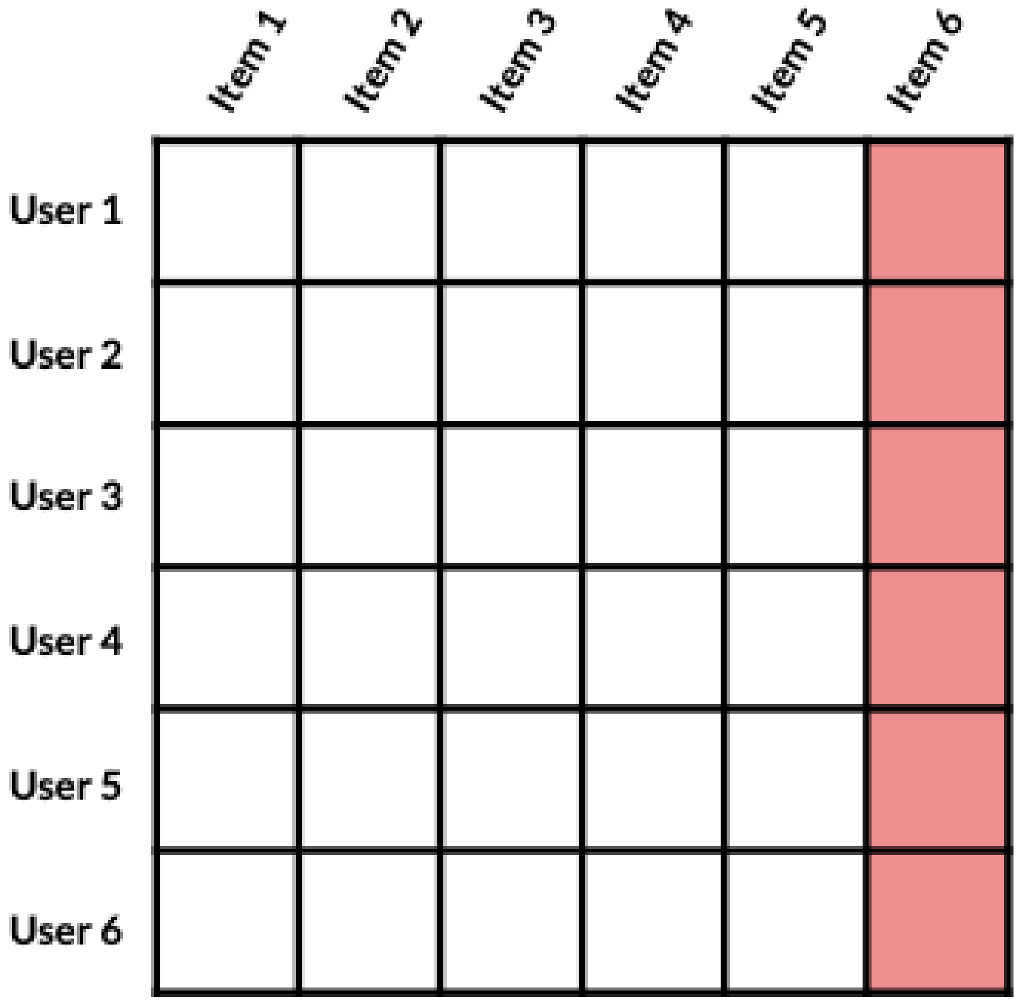


Most Machine Learning Models

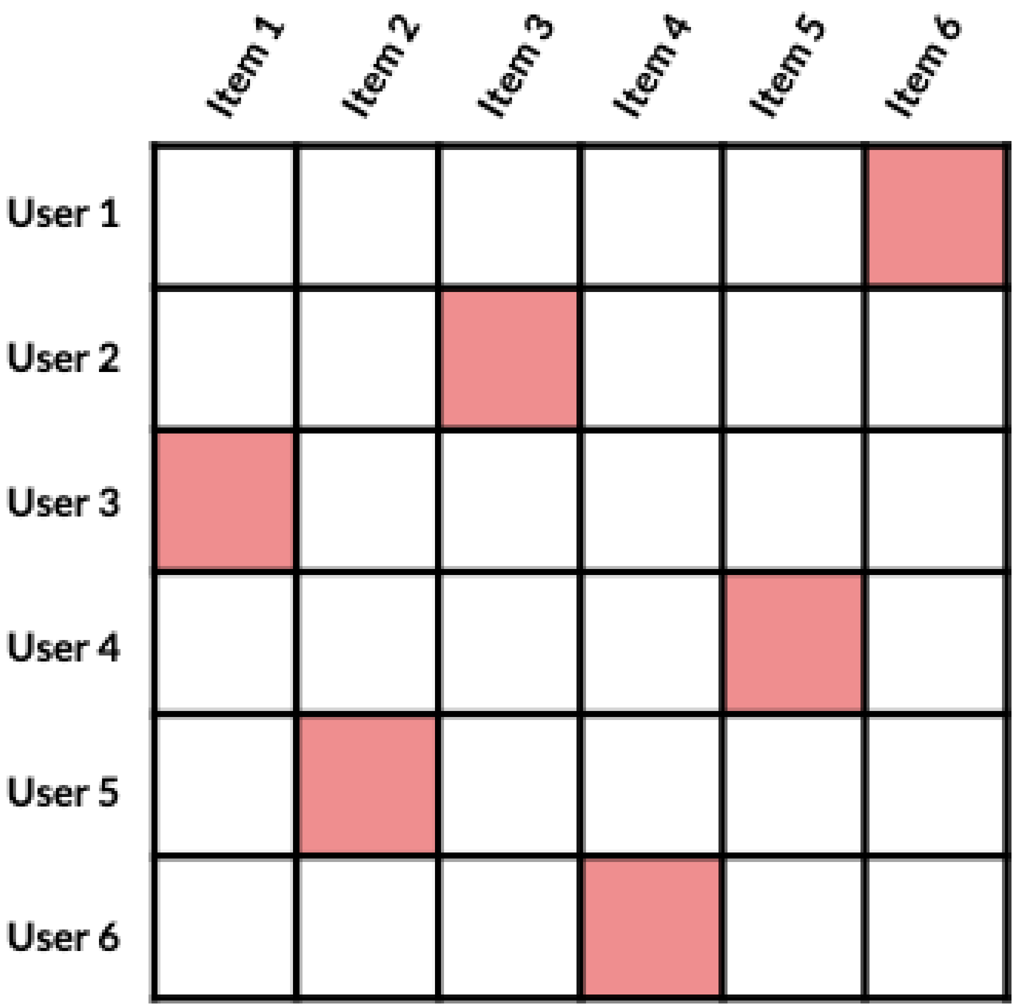


Recommendation Engines

Hold-out sets

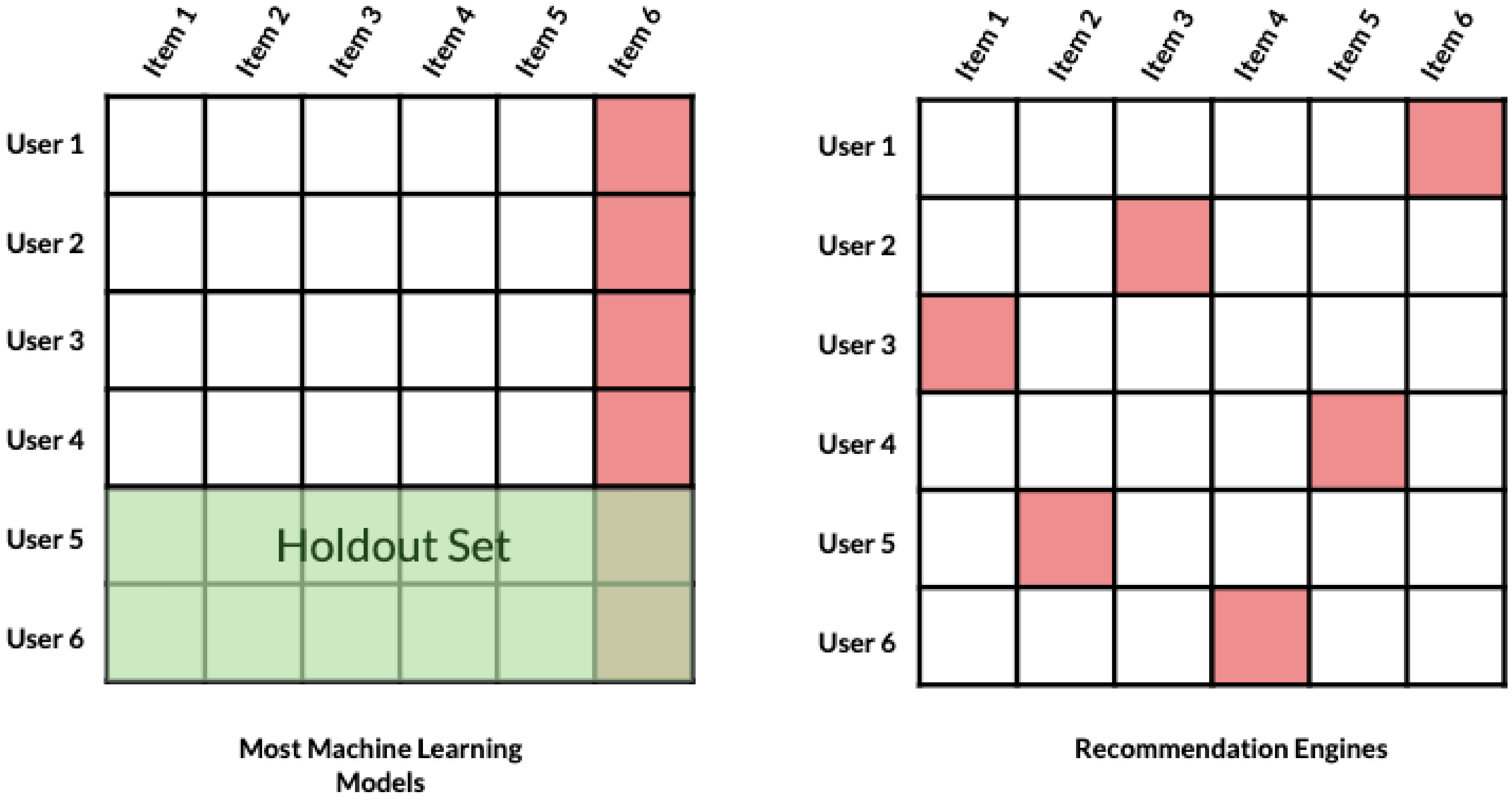


Most Machine Learning Models

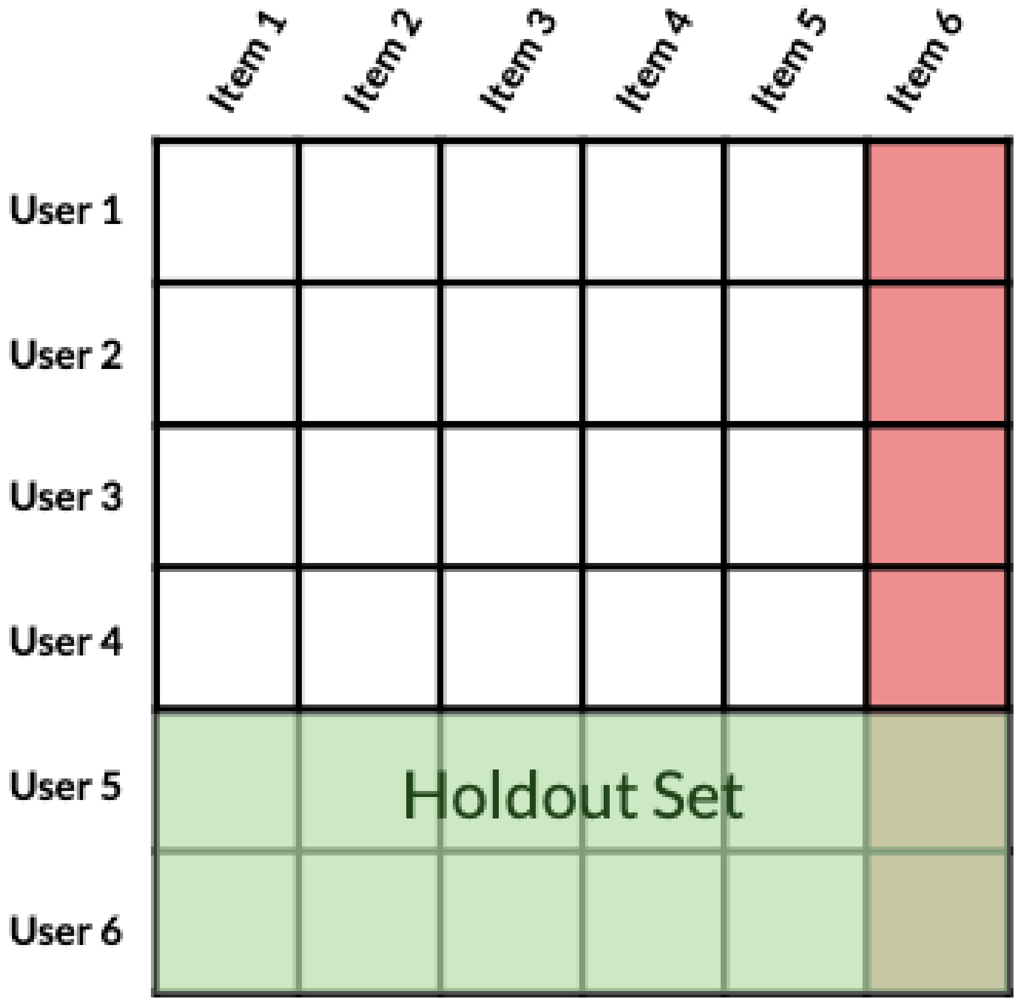


Recommendation Engines

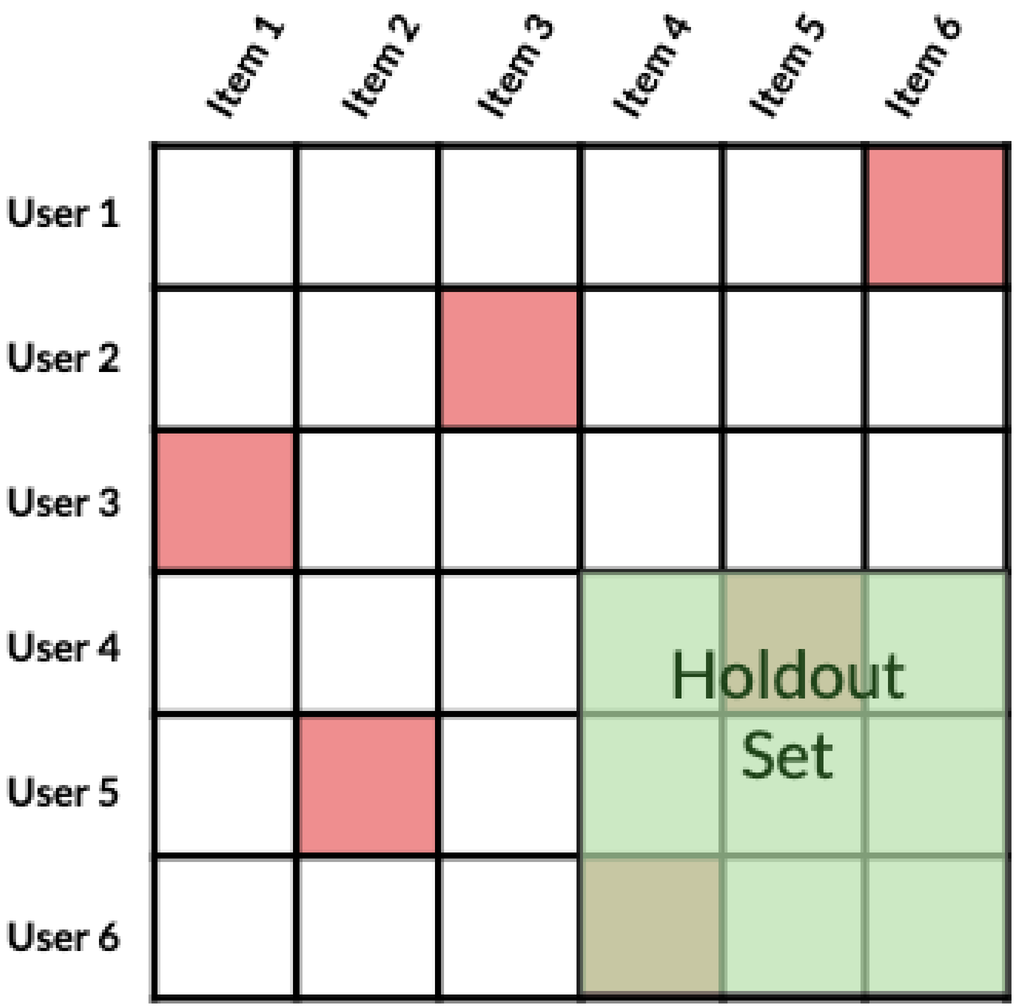
Hold-out sets



Hold-out sets



Most Machine Learning Models



Recommendation Engines

Separating the hold-out set

```
actual_values = act_ratings_df.iloc[:20, :100].values
act_ratings_df.iloc[:20, :100] = np.nan
```

Generate predictions as before.

```
predicted_values = calc_pred_ratings_df.iloc[:20, :100].values
```

Masking the hold-out set

```
mask = ~np.isnan(actual_values)
```

```
print(actual_values[mask])
```

```
[4.  4.  5.  3.  3.  ...]
```

```
print(predicted_values[mask])
```

```
[3.76, 4.35,  4.95,  3.5869079 3.686337   ...]
```

Introducing RMSE (root mean squared error)

Predicted	Actual
4	5
3	3
2	4

Introducing RMSE (root mean squared error)

Predicted	Actual	Difference
4	5	1
3	3	0
2	4	2

Introducing RMSE (root mean squared error)

Predicted	Actual	Difference	Difference^2
4	5	1	1
3	3	0	0
2	4	2	4

Introducing RMSE (root mean squared error)

Predicted	Actual	Difference	Difference^2
4	5	1	1
3	3	0	0
2	4	2	4

$$\sqrt{\frac{\text{Sum of Diff}^2}{\text{Count}}}$$

Introducing RMSE (root mean squared error)

Predicted	Actual	Difference	Difference^2
4	5	1	1
3	3	0	0
2	4	2	4

$$\sqrt{\frac{5}{3}}$$

Introducing RMSE (root mean squared error)

Predicted	Actual	Difference	Difference^2
4	5	1	1
3	3	0	0
2	4	2	4

$$\sqrt{\frac{5}{3}} \Rightarrow 1.29$$

RMSE in Python

```
from sklearn.metrics import mean_squared_error

print(mean_squared_error(actual_values[mask],
                         predicted_values[mask],
                         squared=False))
```

```
3.6223997
```

Let's practice!

BUILDING RECOMMENDATION ENGINES IN PYTHON

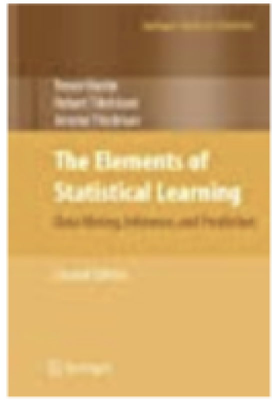
Wrap up

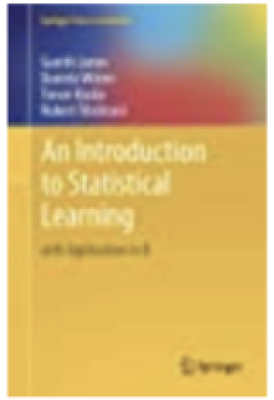
BUILDING RECOMMENDATION ENGINES IN PYTHON

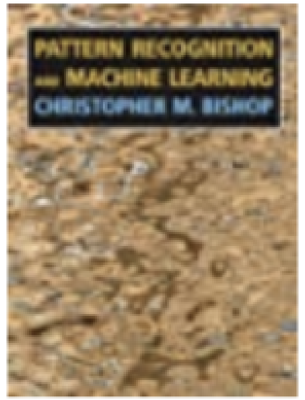
Rob O'Callaghan
Director of Data

Non-personalized models

Frequently bought together



+


+


Total price: $209.02

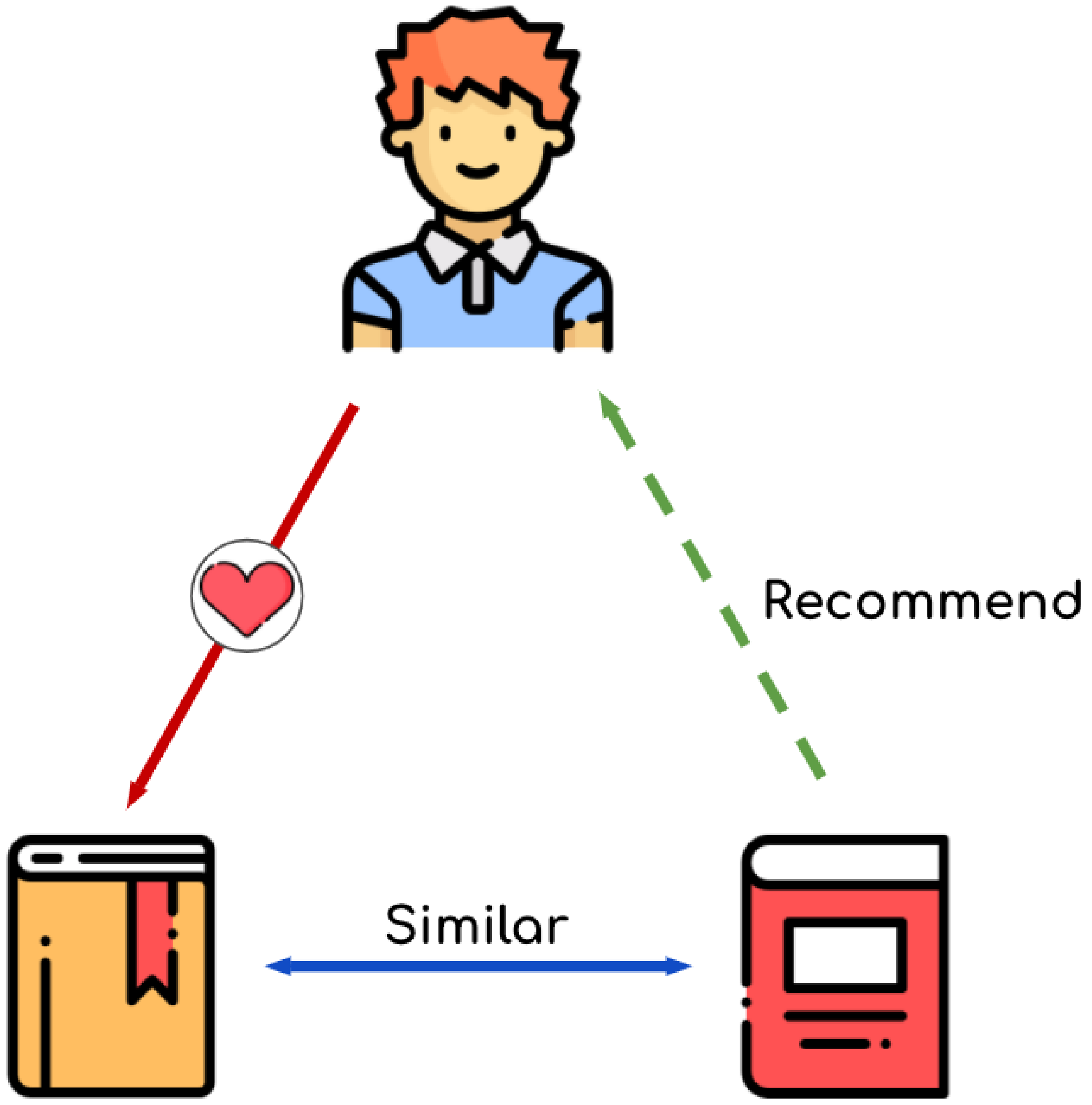
Add all three to Cart

Add all three to List

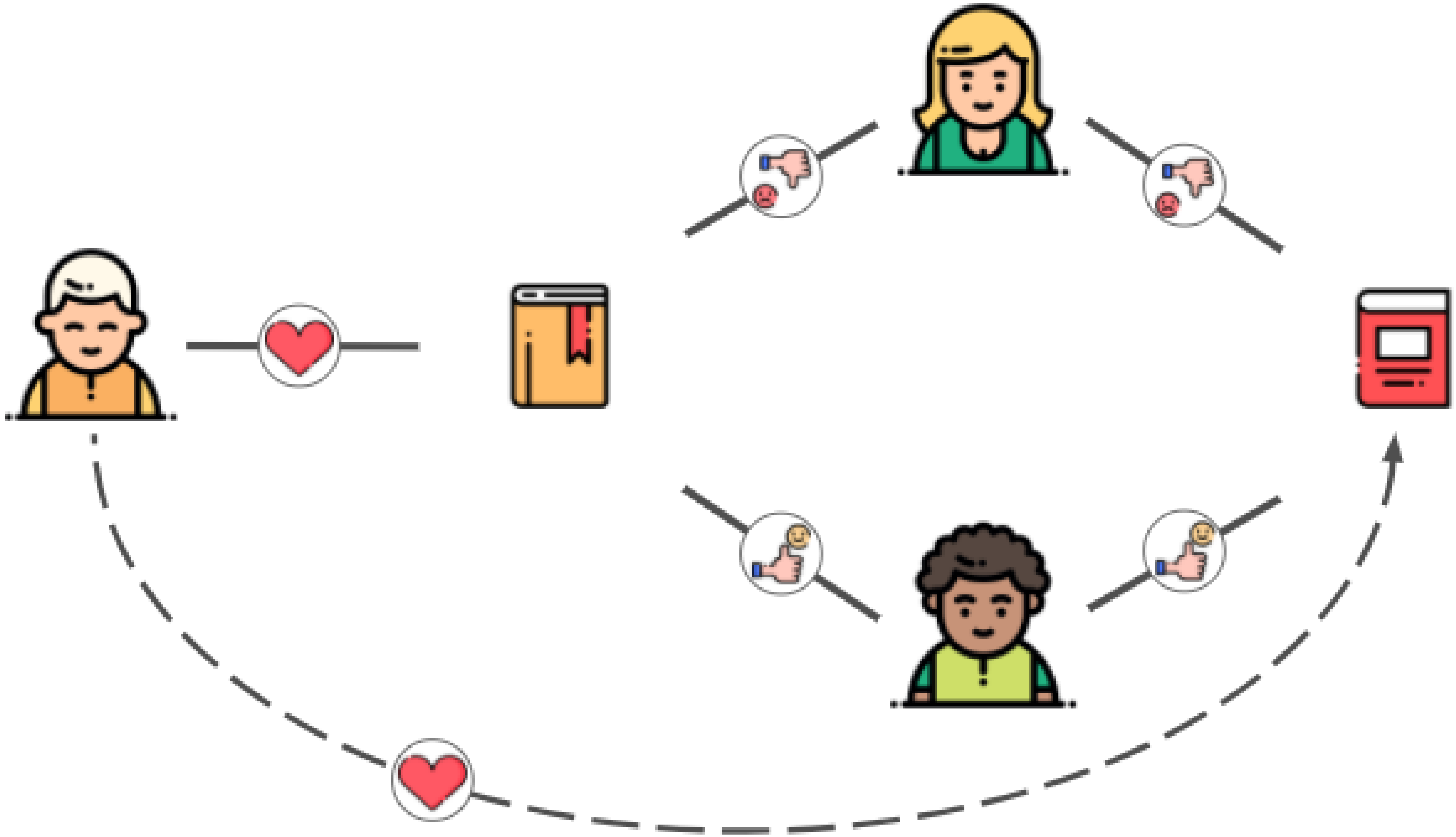
i These items are shipped from and sold by different sellers. Show details

- [x] **This item:** The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition… by Trevor Hastie Hardcover $71.84
- [x] An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics) by Gareth James Hardcover $65.39
- [x] Pattern Recognition and Machine Learning (Information Science and Statistics) by Christopher M. Bishop Hardcover $71.79

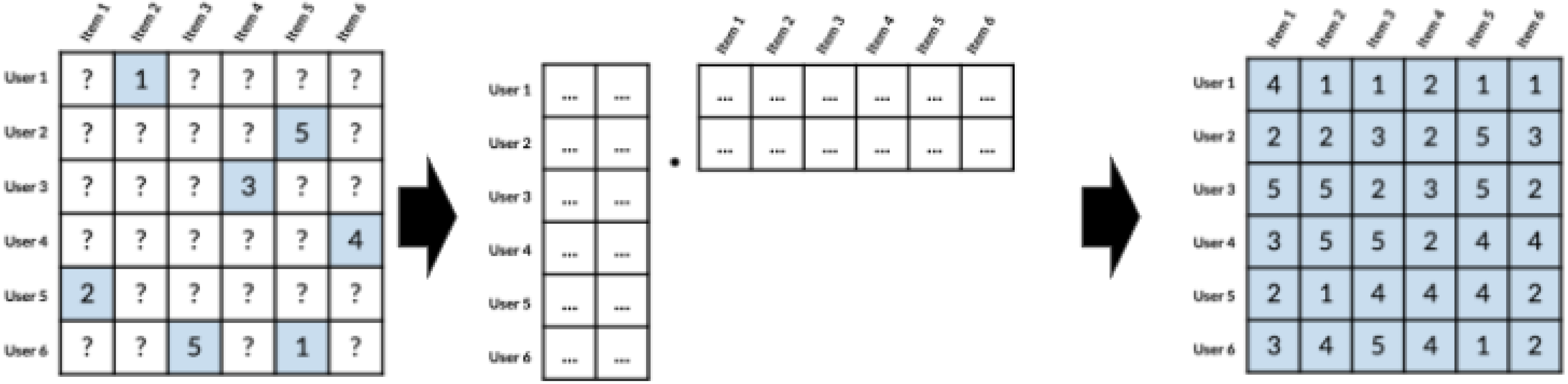
Content-based models



Collaborative filtering



Matrix factorization



Original DataFrame DataFrame Factors Filled DataFrame

Congratulations!

BUILDING RECOMMENDATION ENGINES IN PYTHON