



BUILDING RECOMMENDATION ENGINES WITH PYSPARK

Matrix Multiplication

Jamen Long
Data Scientist

Matrix Multiplication

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \bullet \begin{matrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{matrix}$$

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix, on the left, has its first column highlighted in yellow. A red arrow points to the first element of this column, labeled '1'. The second matrix, in the middle, has its first row highlighted in light blue. A red arrow points to the first element of this row, labeled '9'. To the right of the multiplication symbol (\bullet) is an equals sign (=). The result is a 3x3 matrix where the first element is labeled $(1 * 9)$. All other elements in the result matrix are currently empty.

1	2	3
4	5	6
7	8	9

\bullet

9	8	7
6	5	4
3	2	1

=

$(1 * 9)$		

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix, on the left, has columns labeled 1, 2, and 3. A red arrow points from the first column to the number 1. The second matrix, in the center, has rows labeled 9, 6, and 3. A red arrow points from the top row to the number 9. The result of the multiplication is shown on the right, preceded by an equals sign (=). The first cell of the result matrix contains the expression $(1*9) + (2*6)$.

1	2	3
4	5	6
7	8	9

•

9	8	7
6	5	4
3	2	1

=

$(1*9) + (2*6)$		

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix on the left has columns labeled 1, 2, and 3. The second matrix on the right has rows labeled 9, 6, and 3. The result is shown as a 3x3 matrix with the formula for the top-left cell: $(1*9) + (2*6) + (3*3)$. The matrices are separated by a black dot and an equals sign.

1	2	3
4	5	6
7	8	9

•

9	8	7
6	5	4
3	2	1

=

$(1*9) + (2*6) + (3*3)$		

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix on the left is yellow and has values 1, 2, 3 in its first row. A red arrow points from the second column of the first matrix to the third column of the second matrix. The second matrix is light blue and has values 9, 8, 7 in its top row. A red arrow points from the first column of the second matrix to the third column of the result matrix. The result matrix on the right is light green and has values 9 + 12 + 9, 12 + 15 + 12, and 12 + 9 + 9 in its top row. The middle row and bottom row are empty.

1	2	3
4	5	6
7	8	9

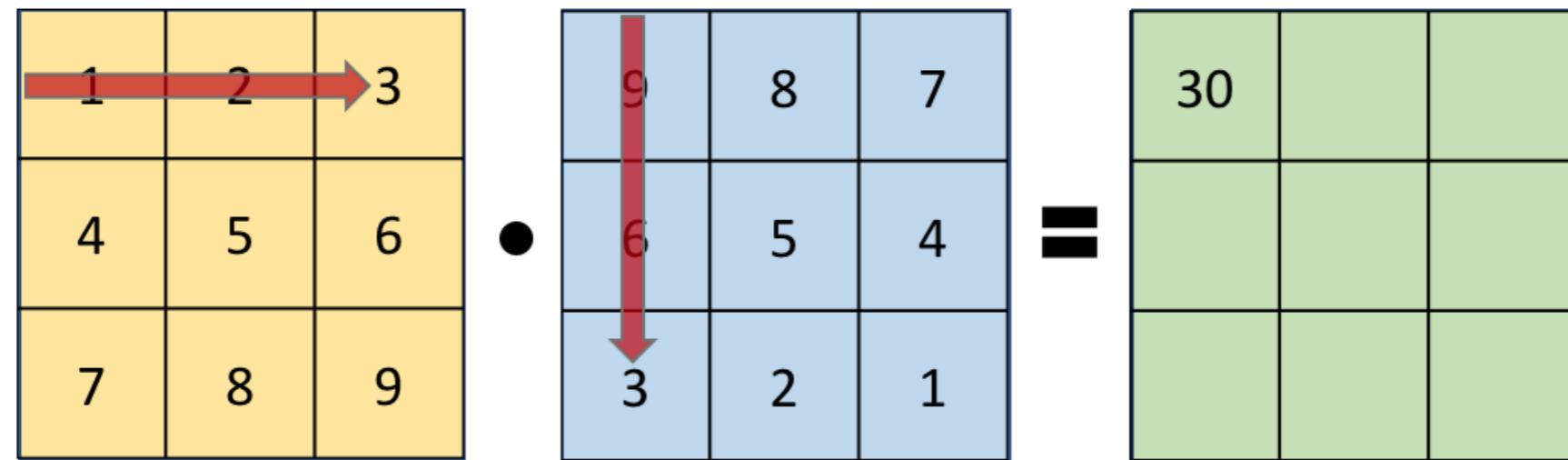
•

9	8	7
6	5	4
3	2	1

=

9 + 12 + 9		

Matrix Multiplication



Matrix Multiplication

$$\begin{matrix} \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} & \bullet & \begin{matrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{matrix} & = & \begin{matrix} 30 & (1*8) & \\ & & \end{matrix} \end{matrix}$$

The diagram illustrates matrix multiplication between two 3x3 matrices. The first matrix has columns labeled 1, 2, and 3. The second matrix has rows labeled 8 and 1. The result is a 3x3 matrix where the first element is 30 and the second element is labeled $(1*8)$. Red arrows point to the first column of the first matrix and the second row of the second matrix.

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix (left) has columns labeled 1, 2, and 3. The second matrix (right) has rows labeled 8 and 5. The result is a 3x3 matrix where the first element is 30 and the formula $(1*8) + (2*5)$ is shown.

1	2	3
4	5	6
7	8	9

•

9	8	7
6	5	4
3	2	1

=

30	$(1*8) + (2*5)$	

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix (left) has columns labeled 1, 2, and 3. The second matrix (middle) has rows labeled 8, 5, and 2. The result (right) is a 3x3 matrix with the value 30 at the top-left position and the formula $(1*8) + (2*5) + (3*2)$ written above it.

1	2	3
4	5	6
7	8	9

•

9	8	7
6	5	4
3	2	1

=

30	$(1*8) + (2*5) + (3*2)$	

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix, on the left, has columns labeled 1, 2, and 3. The second matrix, in the middle, has rows labeled 8, 5, and 2. The result of the multiplication is shown on the right.

Matrix 1:

1	2	3
4	5	6
7	8	9

Matrix 2:

9	8	7
6	5	4
3	2	1

Result:

30	24	

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix (left) has columns labeled 1, 2, and 3. The second matrix (middle) has rows labeled 1, 2, and 3. The result (right) is a 3x3 matrix with the value $(1*7) + (2*4) + (3*1)$ in its top-right cell.

1	2	3
4	5	6
7	8	9

•

9	8	7
6	5	4
3	2	1

=

30	24	$(1*7) + (2*4) + (3*1)$

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix, on the left, has columns labeled 1, 2, and 3. The second matrix, in the middle, has rows labeled 7, 4, and 1. The result of the multiplication is shown on the right.

Matrix 1:

1	2	3
4	5	6
7	8	9

Matrix 2:

9	8	7
6	5	4
3	2	1

Result:

30	24	18

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix (left) has columns labeled 1, 2, 3 and rows labeled 4, 5, 6. The second matrix (middle) has columns labeled 9, 8, 7 and rows labeled 6, 5, 4. The result of the multiplication is a third matrix (right) with columns labeled 30, 24, 18 and rows labeled 84, 0, 0.

Matrix 1:

1	2	3
4	5	6
7	8	9

Matrix 2:

9	8	7
6	5	4
3	2	1

Result:

30	24	18
84	0	0
0	0	0

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix (left) has columns colored yellow, yellow, and yellow. The second matrix (middle) has columns colored light blue, light blue, and light blue. The result matrix (right) has columns colored light green, light green, and light green.

The multiplication is shown as:

1	2	3
4	5	6
7	8	9

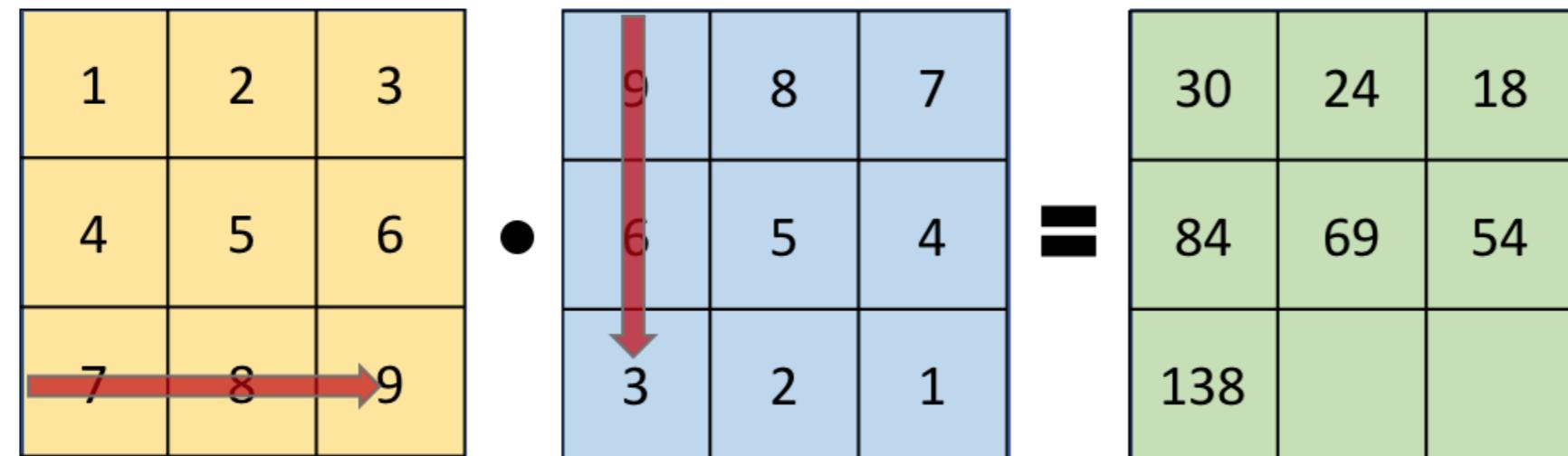
•

9	8	7
6	5	4
3	2	1

=

30	24	18
84	69	54

Matrix Multiplication

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \bullet \begin{matrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{matrix} = \begin{matrix} 30 & 24 & 18 \\ 84 & 69 & 54 \\ 138 & & \end{matrix}$$


Matrix Multiplication

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \bullet \begin{matrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{matrix} = \begin{matrix} 30 & 24 & 18 \\ 84 & 69 & 54 \\ 138 & 114 & \end{matrix}$$

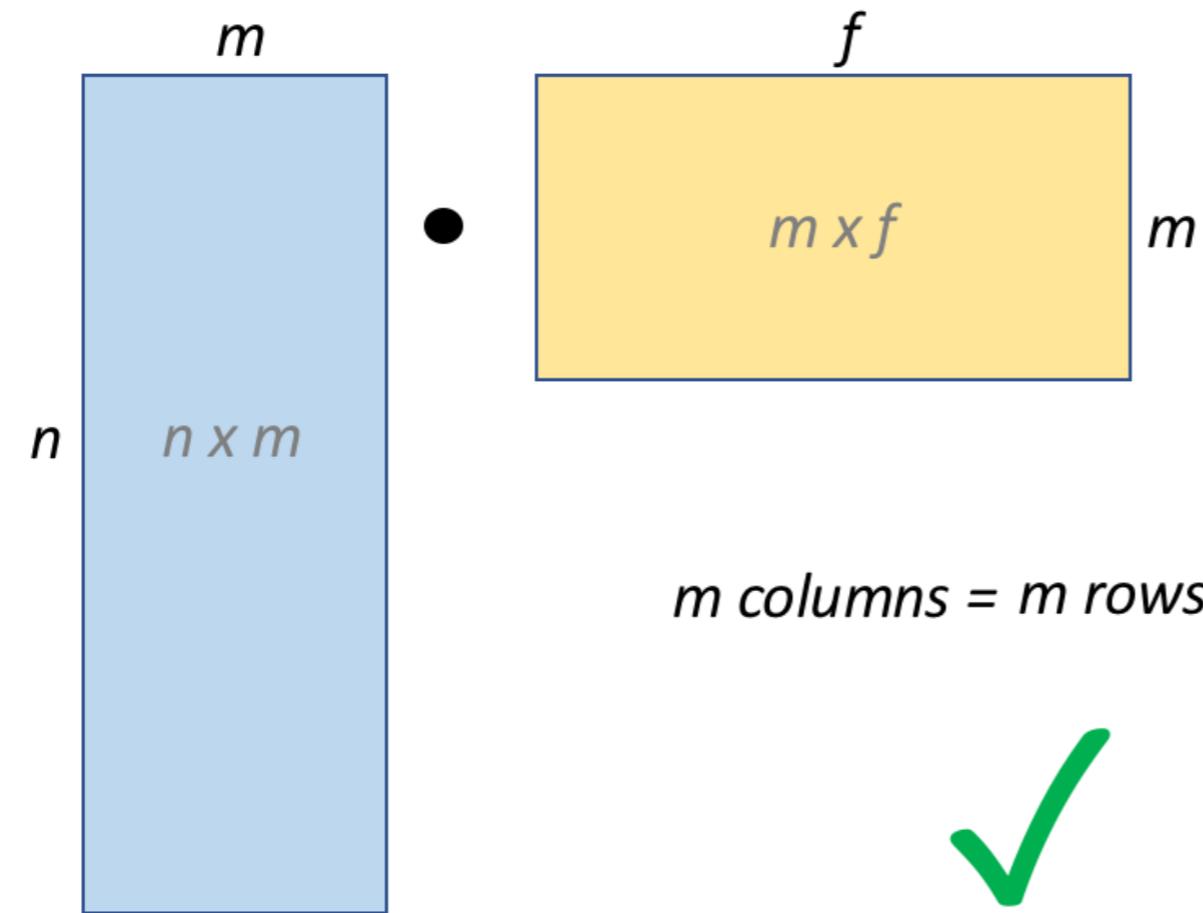
The diagram illustrates the multiplication of two 3x3 matrices. The first matrix has columns labeled 1, 2, 3 and rows labeled 4, 5, 6. The second matrix has columns labeled 9, 8, 7 and rows labeled 6, 5, 4. A red arrow points from the third column of the first matrix to the third row of the second matrix, indicating the calculation of the element at the intersection of the third row and third column of the resulting matrix. The resulting matrix has columns labeled 30, 24, 18 and rows labeled 84, 69, 54.

Matrix Multiplication

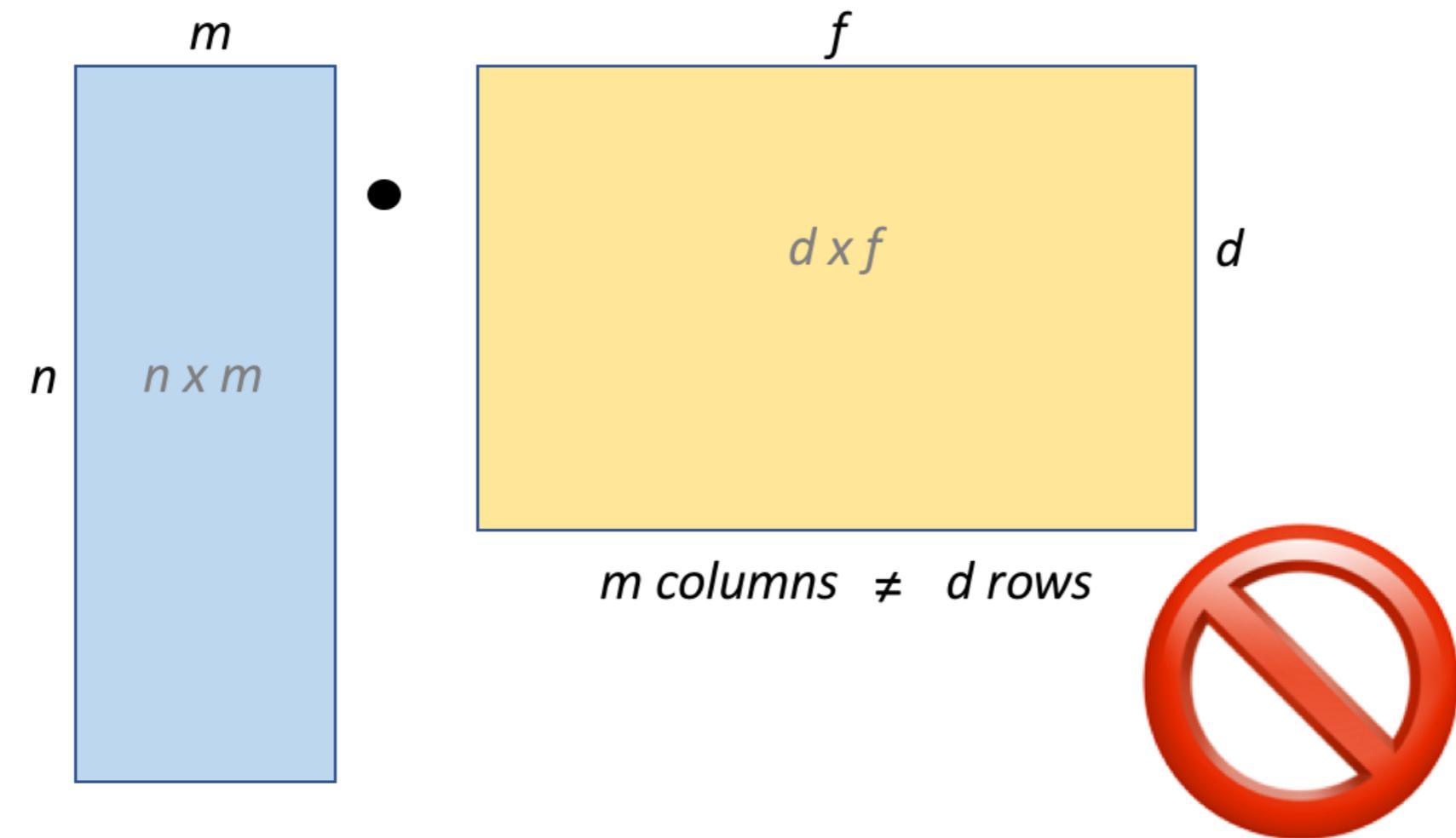
$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \bullet \begin{matrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{matrix} = \begin{matrix} 30 & 24 & 18 \\ 84 & 69 & 54 \\ 138 & 114 & 90 \end{matrix}$$

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix has columns labeled 1, 2, 3 and rows labeled 4, 5, 6. The second matrix has columns labeled 9, 8, 7 and rows labeled 6, 5, 4. A red arrow points from the bottom row of the first matrix to the rightmost column of the second matrix, which is highlighted in red. The result of the multiplication is a third matrix with columns labeled 30, 24, 18 and rows labeled 84, 69, 54. The value 138 is shown in the bottom-left cell of the result matrix.

Matrix Multiplication



Matrix Multiplication





BUILDING RECOMMENDATION ENGINES WITH PYSPARK

Let's practice!

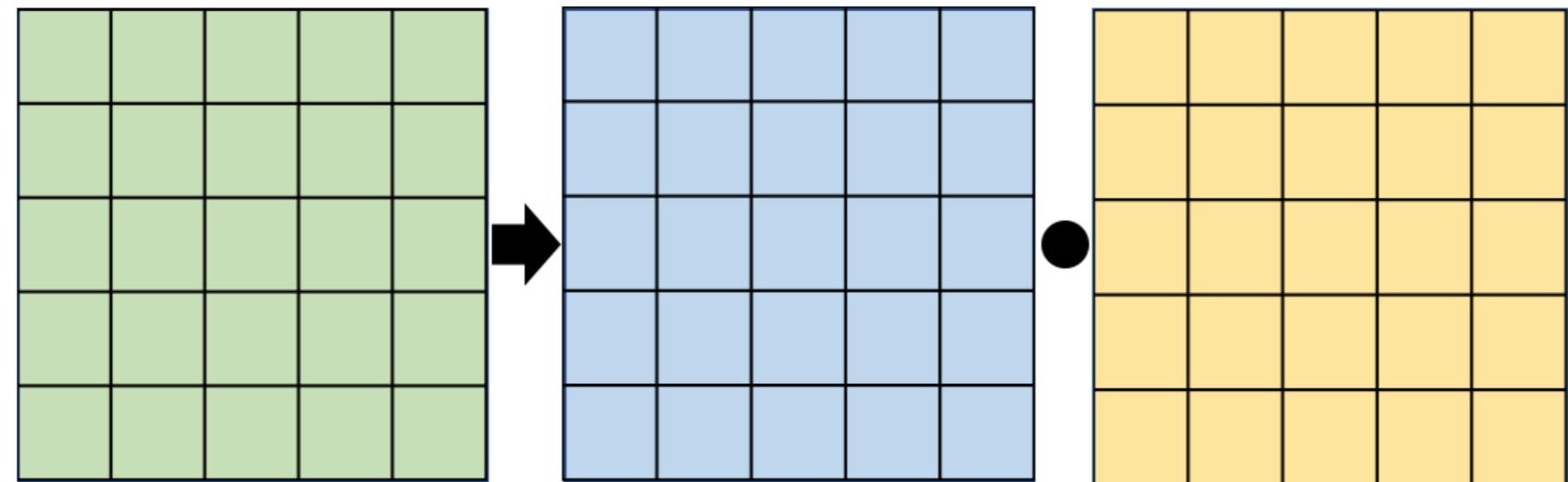


BUILDING RECOMMENDATION ENGINES WITH PYSPARK

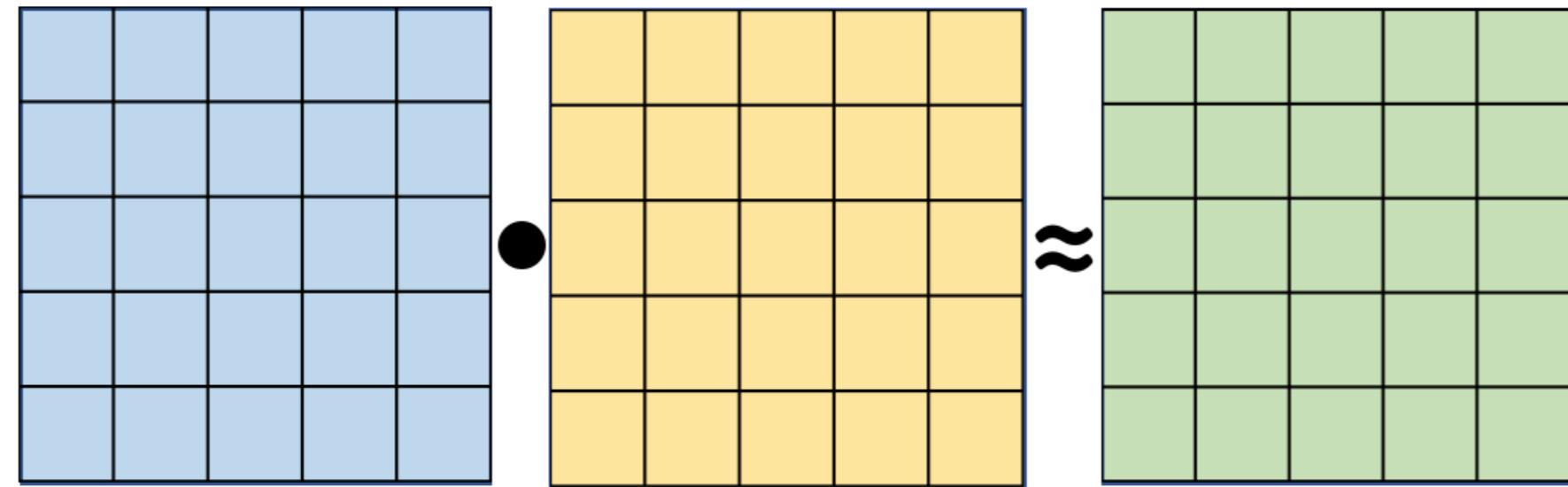
Overview of Matrix Factorization

Jamen Long
Data Scientist

Matrix Factorization



Matrix Factorization



Matrix Factorization

5	1	4	3	3
2	2	4	3	2
1	4	2	4	5
2	2	3	4	2
3	4	4	5	5

Matrix Factorization

The diagram illustrates the decomposition of a 5x5 matrix into two matrices: a blue one on the left and an orange one on the right. The result of this multiplication is a black 5x5 matrix.

Original Matrix:

5	1	4	3	3
2	2	4	3	2
1	4	2	4	5
2	2	3	4	2
3	4	4	5	5

Blue Matrix (Left Factor):

1	0	0	0	0
2/5	1	0	0	0
1/5	19/8	1	0	0
2/5	1	2/9	1	0
3/5	17/8	7/9	2/43	1

Orange Matrix (Right Factor):

5	1	4	3	3
0	8/5	12/5	9/5	4/5
0	0	-9/2	-7/8	5/2
0	0	0	43/36	-5/9
0	0	0	0	-18/43

Resulting Black Matrix:

--	--	--	--	--

Matrix Factorization

The diagram illustrates the process of matrix factorization. It shows a green 5x5 rating matrix being multiplied by two blue 5x3 feature matrices and a yellow 3x5 feature matrix to produce a black 5x5 output matrix.

Rating Matrix:

5	1	4	3	3
2	2	4	3	2
1	4	2	4	5
2	2	3	4	2
3	4	4	5	5

Feature Matrices:

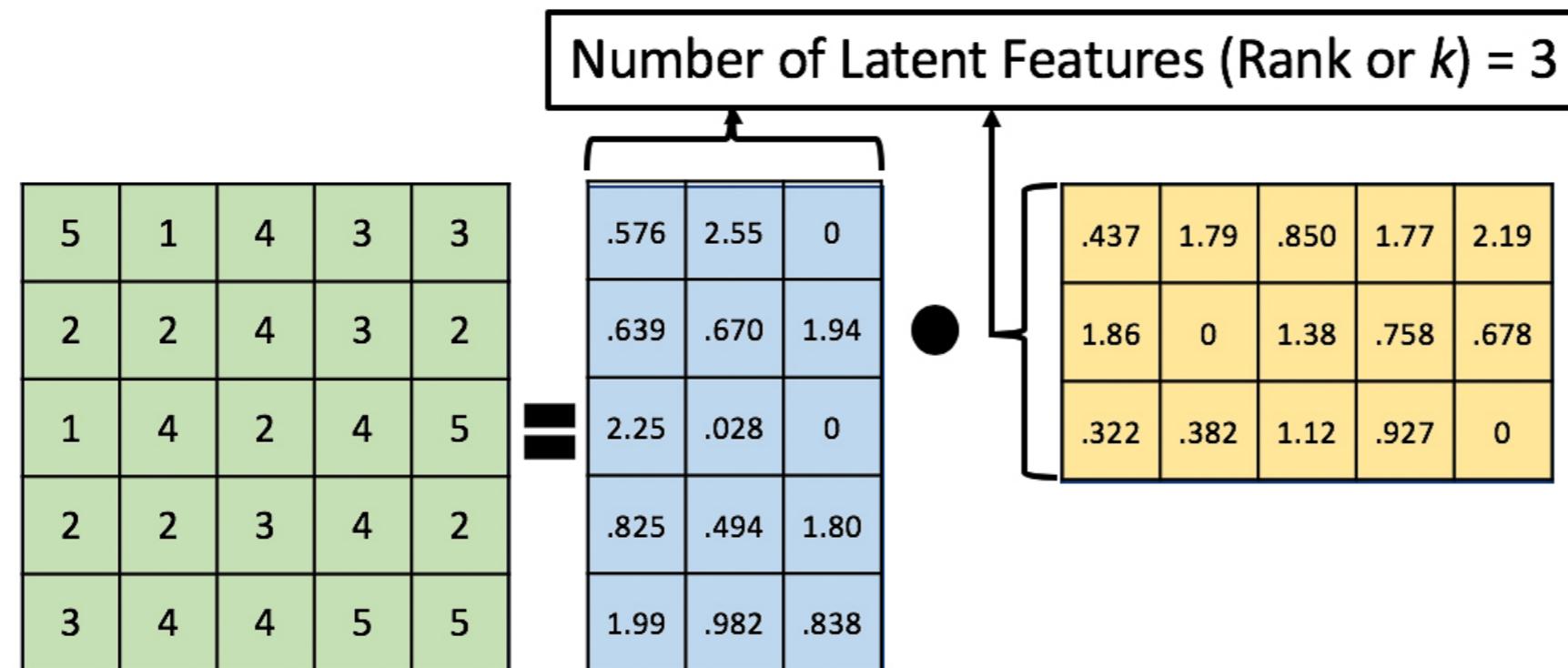
.576	2.55	0
.639	.670	1.94
2.25	.028	0
.825	.494	1.80
1.99	.982	.838

.437	1.79	.850	1.77	2.19
1.86	0	1.38	.758	.678
.322	.382	1.12	.927	0

Output Matrix:

--	--	--	--	--

Rank of Factor Matrices



84	-	48	-
-	50	-	-
-	-	51	-
91	-	-	107

Filling in the Blanks II

$$\begin{matrix} 84 & - & 48 & - \\ - & 50 & - & - \\ - & - & 51 & - \\ 91 & - & - & 107 \end{matrix} = \begin{matrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 \\ 1 & 5 & 2 & 4 & 3 \\ 5 & 1 & 4 & 2 & 3 \end{matrix} \bullet \begin{matrix} 6 & 2 & 3 & 7 \\ 7 & 8 & 4 & 8 \\ 8 & 0 & 5 & 7 \\ 5 & 3 & 3 & 9 \\ 4 & 2 & 2 & 6 \end{matrix}$$

Filling In the Blanks III

The diagram illustrates two matrix operations: matrix multiplication and element-wise product.

Matrix Multiplication: A green circle highlights the value **84** in the first row of the first matrix. This value is the result of multiplying the first row of the first matrix by the first column of the second matrix. The second matrix has a black double equals sign (**=**) below it, indicating it is the result of the multiplication.

Element-wise Product: A green circle highlights the value **6** in the first row of the third matrix. This value is the result of multiplying the first row of the first matrix by the first column of the third matrix. A black dot (**•**) is placed next to the second matrix, indicating it is the result of the element-wise product.

Matrices:

- First Matrix:**

84	-	48	-
-	50	-	-
-	-	51	-
91	-	-	107
- Second Matrix:**

1	2	3	4	5
5	4	3	2	1
1	5	2	4	3
5	1	4	2	3
- Third Matrix:**

6	2	3	7
7	8	4	8
8	0	5	7
5	3	3	9
4	2	2	6

Filling In the Blanks IV

The diagram illustrates the multiplication of three matrices:

- Matrix 1:**

84	-	48	-
-	50	-	-
-	-	51	-
91	-	-	107
- Matrix 2:**

1	2	3	4	5
5	4	3	2	1
1	5	2	4	3
5	1	4	2	3
- Matrix 3:**

6	2	3	7
7	8	4	8
8	0	5	7
5	3	3	9
4	2	2	6

A green circle highlights the value 50 in the first matrix.

Filling In the Blanks V

The diagram illustrates the process of filling in missing values (blanks) using matrix multiplication. It shows three matrices: a sparse input matrix, a weight matrix, and a dense output matrix.

Sparse Input Matrix:

84	-	48	-
-	50	-	-
-	-	51	-
91	-	-	107

Weight Matrix:

1	2	3	4	5
5	4	3	2	1
1	5	2	4	3
5	1	4	2	3

Dense Output Matrix:

6	2	3	7
7	8	4	8
8	0	5	7
5	3	3	9
4	2	2	6

A green oval highlights the value 51 in the sparse input matrix, which corresponds to the value 3 in the second row of the weight matrix. A green circle highlights the value 3 in the third row of the weight matrix, which corresponds to the value 3 in the third row of the output matrix. A green dot at the bottom indicates the result of the multiplication.

Filling In the Blanks VI

The diagram illustrates the multiplication of two matrices to fill in missing values. The result matrix has a green circle around its bottom-right corner value.

Matrix 1 (Input):

84	-	48	-
-	50	-	-
-	-	51	-
91	-	-	107

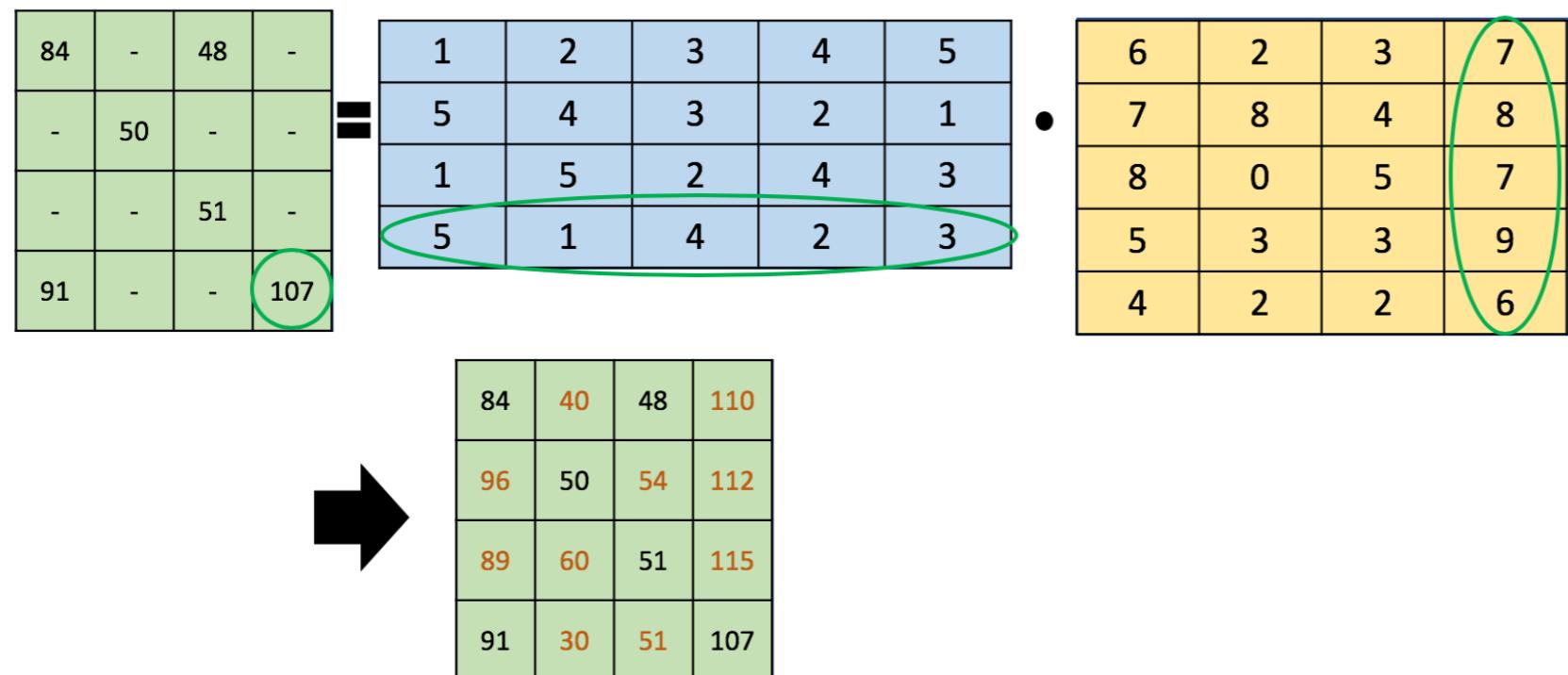
Matrix 2 (Input):

1	2	3	4	5
5	4	3	2	1
1	5	2	4	3
5	1	4	2	3

Result Matrix (Output):

6	2	3	7
7	8	4	8
8	0	5	7
5	3	3	9
4	2	2	6

Filling In the Blanks VII





BUILDING RECOMMENDATION ENGINES WITH PYSPARK

Let's practice!



BUILDING RECOMMENDATION ENGINES WITH PYSPARK

How ALS Alternates to Generate Predictions

Jamen Long
Data Scientist

	movieId	1	2	3	4	5	6
userId							
1		-	-	-	-	-	-
2		-	-	-	-	-	-
3		-	-	-	-	-	-
4		-	-	-	-	-	-
5		-	-	-	4	-	-
6		-	-	-	-	-	-
7		3	-	-	-	-	-
8		-	-	-	-	-	-
9		4	-	-	-	-	-
10		-	-	-	-	-	-
11		-	-	-	-	-	-
12		-	-	-	-	-	-

A large, semi-transparent gray letter 'R' is centered in the middle of the table, serving as a watermark or logo.

	movieId	1	2	3	4	5	6
	userId						
1		-	-	-	-	-	-
2		-	-	-	-	-	-
3		-	-	-	-	-	-
4		-	-	-	-	-	-
5		-	-	4	-	-	-
6		-	-	-	-	-	-
7		3	-	-	-	-	-
8		-	-	-	-	-	-
9		4	-	-	-	-	-
10		-	-	-	-	-	-
11		-	-	-	-	-	-
12		-	-	-	-	-	-

		U_LF_0	U_LF_1	U_LF_2
User	_id	0.000000	0.092497	0.007393
User_0		0.000000	0.000000	1.322806
User_1		0.293443	0.000000	0.362894
User_2		0.140157	1.074063	1.332295
User_3		0.495512	0.075752	0.529940
User_4		0.219477	0.124221	0.000000
User_5		0.022552	0.162423	1.088869
User_6		0.999537	0.109175	0.372134
User_7		0.124147	0.180746	0.276849
User_8		0.144918	0.154747	0.196846
User_9		0.177815	0.025850	0.015767
User_10		0.066618	0.117502	0.064694
User_11		0.367768	0.000000	0.322991
User_12				

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6	
M_LF	_0	1.296350	0.290096	0.000000	0.000000	0.044772	0.372374	0.000000
M_LF	_1	0.297598	0.000000	0.008054	0.005646	0.036239	0.296742	0.172025
M_LF	_2	1.265775	0.980059	0.471187	0.096935	0.465978	0.687785	0.419522

	movieId	1	2	3	4	5	6
userId	1	-	-	-	-	-	-
1	2	-	-	-	-	-	-
2	3	-	-	-	-	-	-
3	4	-	-	-	-	-	-
4	5	-	-	-	-	-	-
5	6	-	-	-	-	-	-
6	7	3	-	-	-	-	-
7	8	-	-	-	-	-	-
8	9	4	-	-	-	-	-
9	10	-	-	-	-	-	-
10	11	-	-	-	-	-	-
11	12	-	-	-	-	-	-

Constant

	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	1.074023	1.332295
User_4	0.495512	0.07752	0.529940
User_5	0.219477	0.121221	0.000000
User_6	0.022552	0.162423	1.088869
User_7	0.999537	0.109175	0.372134
User_8	0.114117	0.180746	0.276849
User_9	0.144918	0.154747	0.196846
User_10	0.177815	0.025850	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

Constant

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	0.000000	0.000000	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	0.00834	0.005646	0.036239	0.296742	0.172025
M_LF_2	1.265775	0.080059	0.47181	0.096935	0.465978	0.687785	0.419522

Adjusted

movieId	1	2	3	4	5	6
userId	-	-	-	-	-	-
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-
6	-	-	-	-	-	-
7	3	-	-	-	-	-
8	-	-	-	-	-	-
9	4	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-
12	-	-	-	-	-	-

Constant

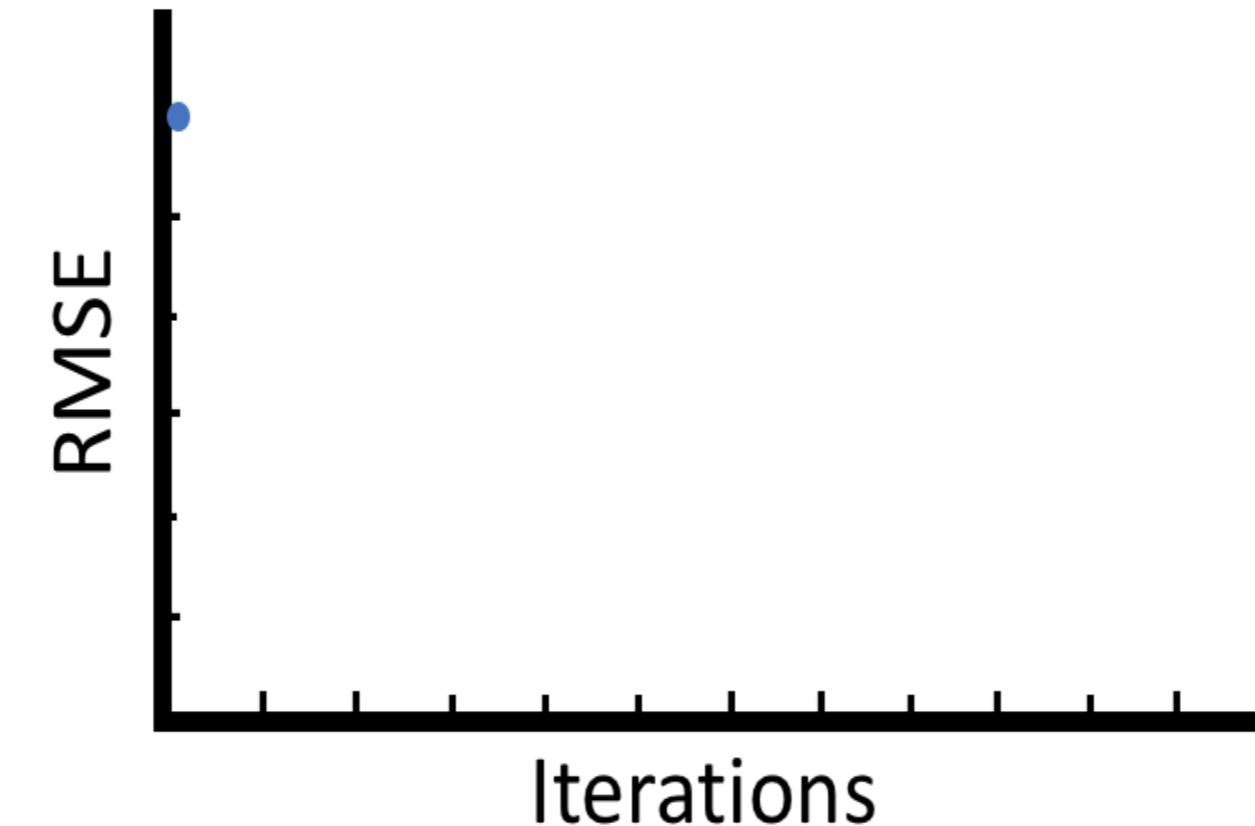
	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	1.074023	1.332295
User_4	0.495512	0.07752	0.529940
User_5	0.219477	0.121221	0.000000
User_6	0.022552	0.162423	1.088869
User_7	0.999537	0.109175	0.372134
User_8	0.114117	0.180746	0.276849
User_9	0.144918	0.154747	0.196846
User_10	0.177815	0.025850	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

Constant

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	0.000000	0.000000	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	0.008514	0.005646	0.036239	0.296742	0.172025
M_LF_2	1.265775	0.080059	0.47181	0.096935	0.465978	0.687785	0.419522

Adjusted

Iteration: 1
RMSE = 12,000



	movieId	1	2	3	4	5	6
	userId	1	-	-	-	-	-
1	1	-	-	-	-	-	-
2	2	-	-	-	-	-	-
3	3	-	-	-	-	-	-
4	4	-	-	-	-	-	-
5	5	-	-	-	-	-	-
6	6	-	-	-	-	-	-
7	7	3	-	-	-	-	-
8	8	-	-	-	-	-	-
9	9	4	-	-	-	-	-
10	10	-	-	-	-	-	-
11	11	-	-	-	-	-	-
12	12	-	-	-	-	-	-

Constant

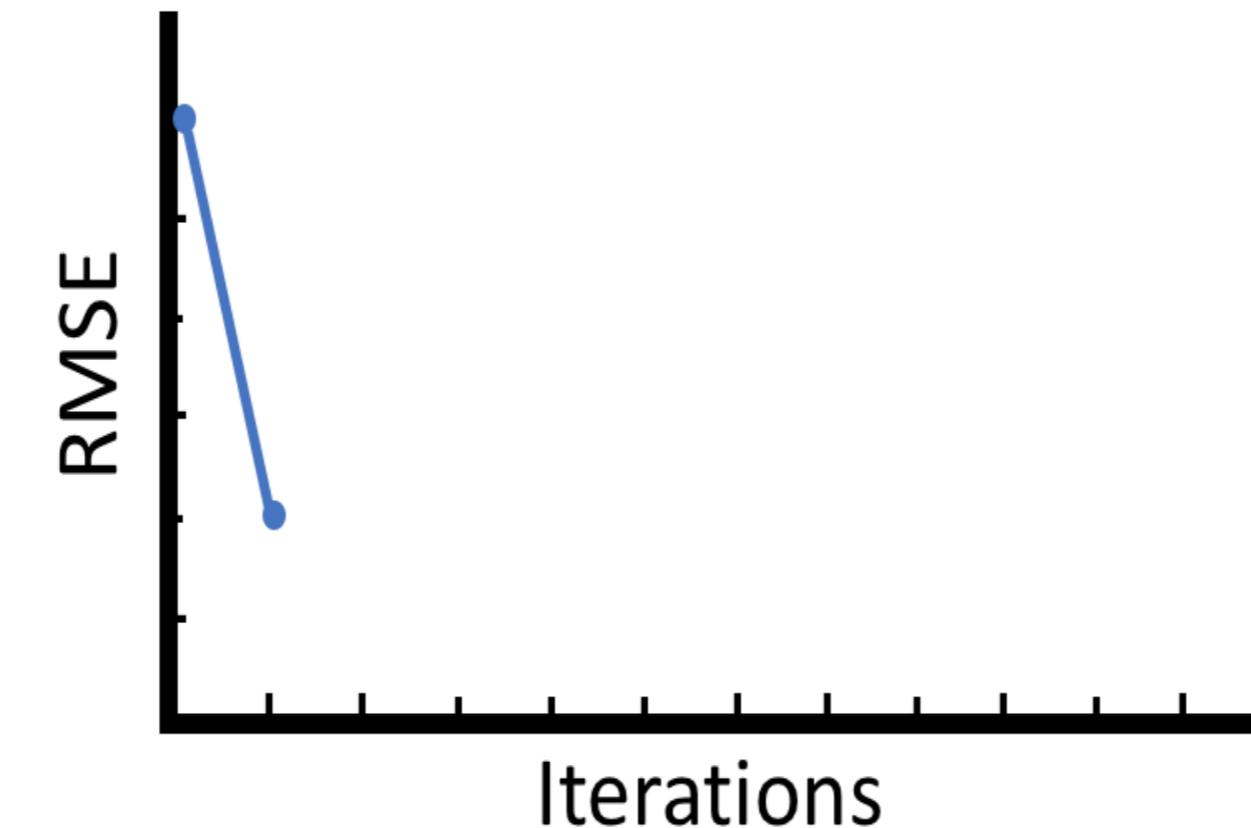
	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	1.074063	0.332295
User_4	0.495512	0.075752	0.529940
User_5	0.219477	0.121221	0.000000
User_6	0.022552	0.162423	1.088869
User_7	0.995337	0.109175	0.372134
User_8	0.124147	0.180746	0.276849
User_9	0.144918	0.154747	0.196846
User_10	0.177815	0.025850	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

Adjusted

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	0.000000	0.000000	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	0.008333	0.005646	0.036239	0.296742	0.172025
M_LF_2	1.265775	0.080059	0.471881	0.096935	0.465978	0.687785	0.419522

Constant

Iteration: 2
RMSE = 4,000



movieId	1	2	3	4	5	6
userId	-	-	-	-	-	-
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-
6	-	-	-	-	-	-
7	3	-	-	-	-	-
8	-	-	-	-	-	-
9	4	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-
12	-	-	-	-	-	-

Constant

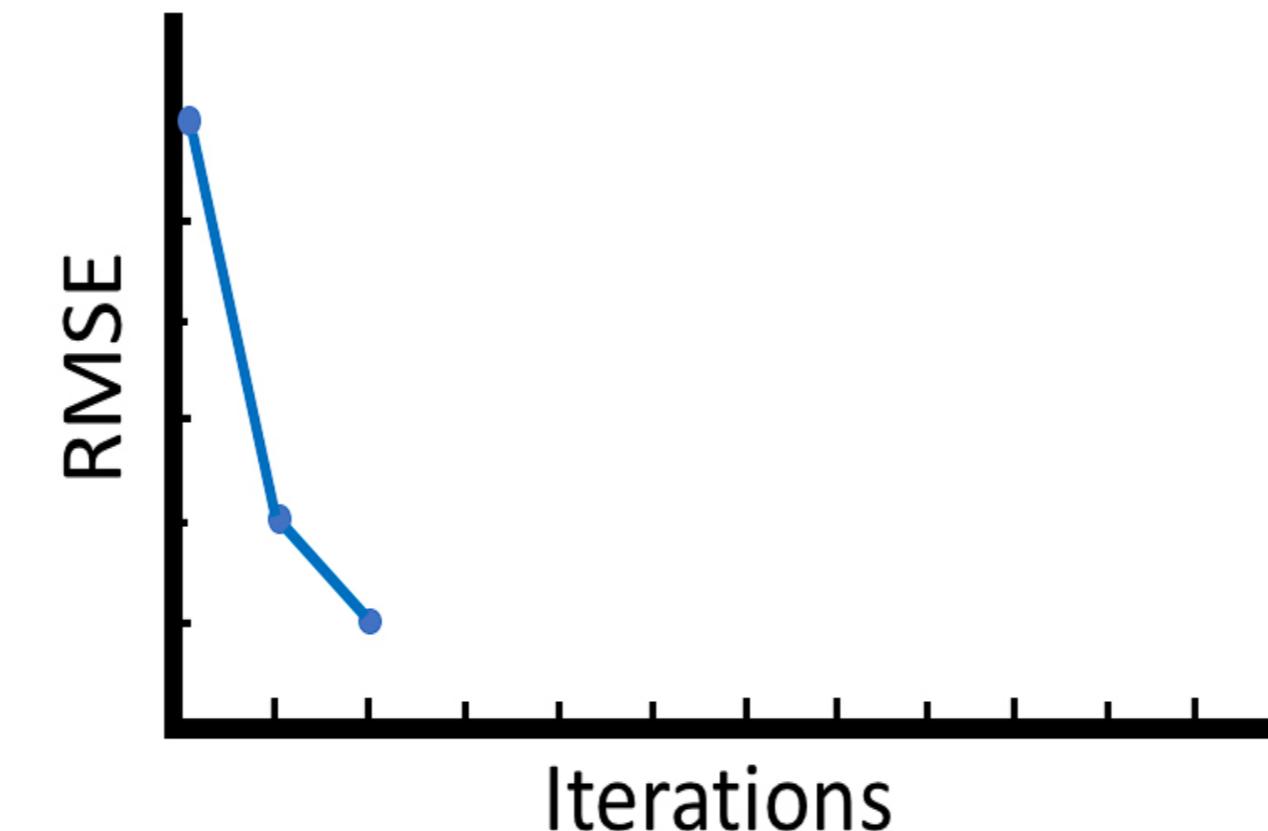
	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	1.074031	1.332295
User_4	0.495512	0.077752	0.529940
User_5	0.219477	0.121221	0.000000
User_6	0.022552	0.162423	1.088869
User_7	0.999517	0.109175	0.372134
User_8	0.114117	0.180746	0.276849
User_9	0.144918	0.154747	0.196846
User_10	0.177815	0.025850	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

Constant

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	0.000000	0.000000	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	0.008044	0.005646	0.036239	0.296742	0.172025
M_LF_2	1.265775	0.080059	0.471181	0.096935	0.465978	0.687785	0.419522

Adjusted

Iteration: 3
RMSE = 2,000



movieId	1	2	3	4	5	6
userId	-	-	-	-	-	-
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-
6	-	-	-	-	-	-
7	3	-	-	-	-	-
8	4	-	-	-	-	-
9	-	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-
12	-	-	-	-	-	-

Constant

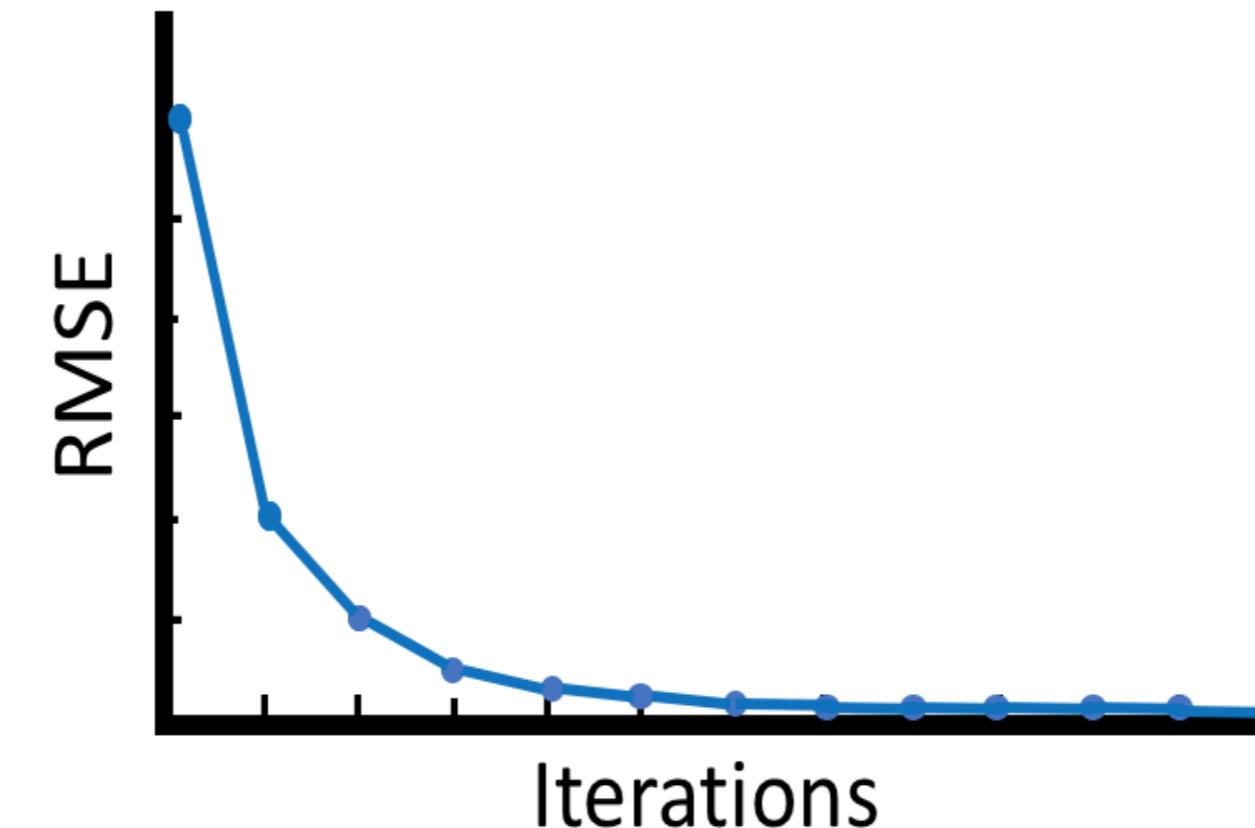
	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	1.074063	0.332295
User_4	0.495512	0.078752	0.529940
User_5	0.219477	0.121221	0.000000
User_6	0.022552	0.162423	1.088869
User_7	0.995337	0.109175	0.372134
User_8	0.124147	0.180746	0.276849
User_9	0.144918	0.154747	0.196846
User_10	0.177815	0.025850	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

Adjusted

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	0.000000	0.000000	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	0.000000	0.005646	0.036239	0.296742	0.172025
M_LF_2	1.265775	0.080059	0.47182	0.096935	0.465978	0.687785	0.419522

Constant

Iteration: n
RMSE = 0.6



	movieId	1	2	3	4	5	6
	userId						
1		-	-	-	-	-	-
2		-	-	-	-	-	-
3		-	-	-	-	-	-
4		-	-	-	-	-	-
5		-	4	-	-	-	-
6		-	-	-	-	-	-
7		3	-	-	-	-	-
8		-	-	-	-	-	-
9		4	-	-	-	-	-
10		-	-	-	-	-	-
11		-	-	-	-	-	-
12		-	-	-	-	-	-

Original Sparse Matrix

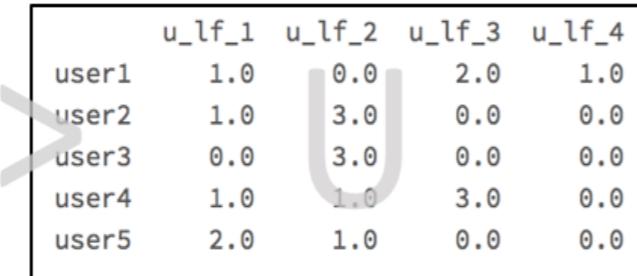


	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
User_1	1.296428	3.937481	0.128226	0.616399	0.909806	0.554946
User_2	0.440784	0.170991	0.035177	0.182239	0.358864	2.782786
User_3	1.346387	0.636410	0.135210	3.421250	1.287243	0.743692
User_4	0.663118	0.250311	0.051797	0.271871	2.867512	0.235352
User_5	0.063669	0.001000	4.000000	0.014328	0.118589	0.021369
User_6	1.073698	1.343627	0.106466	0.514285	0.805503	0.484745
User_7	3.000000	0.176224	0.036689	0.222114	0.660547	0.174899
User_8	0.307343	0.131904	0.027857	0.141114	3.989767	0.147237
User_9	4.000000	0.093997	0.019955	0.103822	0.235271	0.109201
User_10	0.067036	0.007638	1.452647	0.016245	0.084729	0.011062
User_11	0.082730	0.031429	0.006935	0.037387	0.104170	2.039812
User_12	0.423238	0.152189	0.031309	0.166972	4.001211	0.135502

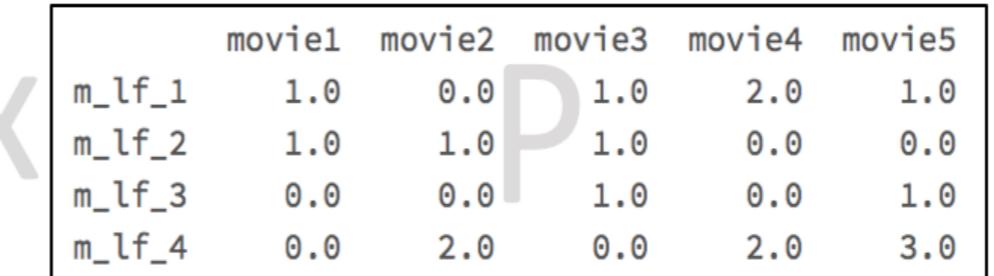
Completed Prediction Matrix

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-



	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

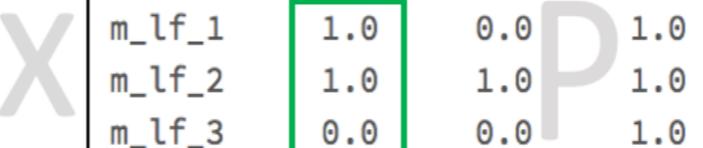


	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-



	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0



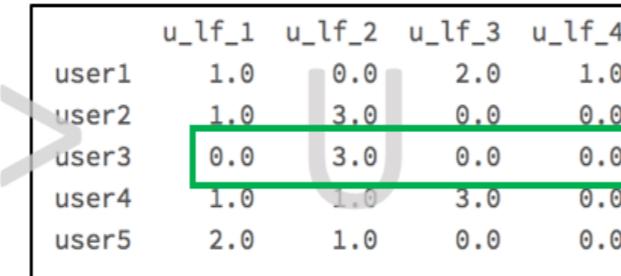
	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-

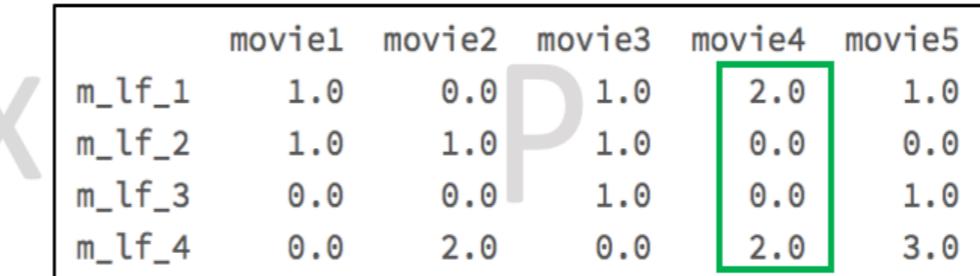
	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-

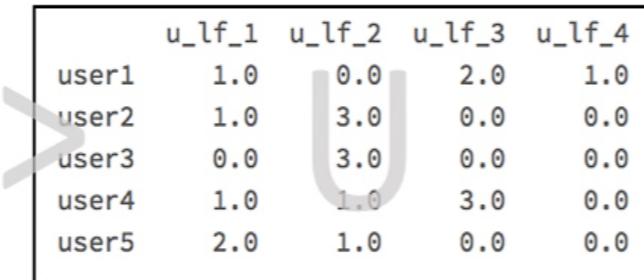


	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

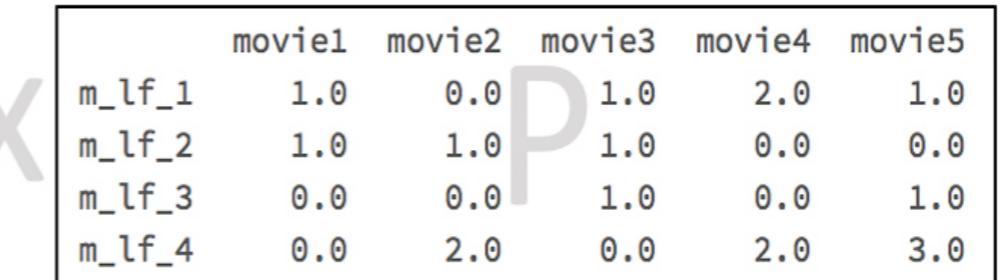


	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

	movie1	movie2	movie3	movie4	movie5
user1			3	-	5
user2	3	-	-	-	1
user3			3	2	
user4		1	-	-	
user5	2	-	-	-	



	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0



	movie1	movie2	movie3	movie4	movie5
m_lf_1		1.0	0.0	1.0	2.0
m_lf_2		1.0	1.0	1.0	0.0
m_lf_3		0.0	0.0	1.0	0.0
m_lf_4		0.0	2.0	0.0	2.0
					3.0

	movie1	movie2	movie3	movie4	movie5
user1		1	3	1	5
user2	3	-	-	1	
user3	-	-	3	2	
user4	-	-	1	-	-
user5	-	2	-	-	-

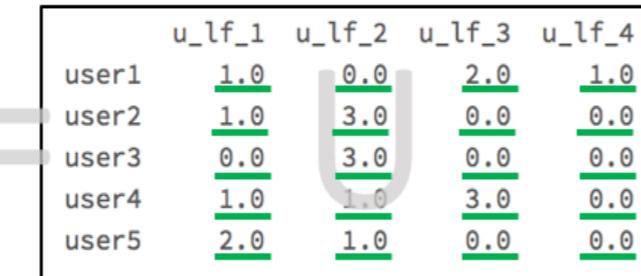
R

	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

X P

	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-



	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-

	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

	movie1	movie2	movie3	movie4	movie5
user1	-	3	-	5	
user2	3	-	-	-	1
user3	-	-	3	2	
user4	-	-	1	-	-
user5	-	2	-	-	-

	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

	movie1	movie2	movie3	movie4	movie5
m_lf_1		1.0	0.0	1.0	2.0
m_lf_2		1.0	1.0	1.0	0.0
m_lf_3		0.0	0.0	1.0	0.0
m_lf_4		0.0	2.0	0.0	2.0
					3.0

	movie1	movie2	movie3	movie4	movie5
user1	1.0	2.0	3.0	4.0	6.0
user2	4.0	3.0	4.0	2.0	1.0
user3	3.0	3.0	3.0	1.0	1.0
user4	2.0	1.0	5.0	1.0	4.0
user5	3.0	1.0	3.0	4.0	2.0



BUILDING RECOMMENDATION ENGINES WITH PYSPARK

Let's practice!



BUILDING RECOMMENDATION ENGINES WITH PYSPARK

Data Preparation for Spark ALS

Jamen Long
Data Scientist

Conventional Dataframe

userId	Good Will H...	Batman For...	Incredibles	Shawshank Redemption	Coco
z097s3	2	3	null	4	4
z176c4	1	null	4	3	4
m821i6	3	4	null	3	5
t872c7	1	2	4	5	null
b728q0	2	null	5	2	null
f540n1	2	1	null	3	1
w066f1	5	null	5	2	5
v081u6	1	null	5	1	1
j197o6	3	2	2	4	null
n202j1	2	null	2	null	2
p755a0	2	3	4	5	5
t791a0	5	5	null	1	4
c460j6	4	1	null	4	4
z595b3	1	2	4	null	1
h296x8	4	3	5	2	4
a610z0	2	1	null	4	4
g025o2	5	4	2	2	null
u902e2	null	3	4	1	5
t893x2	1	4	null	null	5
x668y8	2	3	5	2	null

Row-Based Data Format

```
+-----+-----+-----+
|userId|          variable|rating|
+-----+-----+-----+
|z097s3|  Good Will Hunting|  2|
|z097s3|    Batman Forever|  3|
|z097s3|The Shawshank Red...|  4|
|z097s3|          Coco|  4|
|z176c4|  Good Will Hunting|  1|
|z176c4|    The Incredibles|  4|
|z176c4|The Shawshank Red...|  3|
|z176c4|          Coco|  4|
|m821i6|  Good Will Hunting|  3|
|m821i6|    Batman Forever|  4|
|m821i6|The Shawshank Red...|  3|
|m821i6|          Coco|  5|
|t872c7|  Good Will Hunting|  1|
|t872c7|    Batman Forever|  2|
|t872c7|    The Incredibles|  4|
|t872c7|The Shawshank Red...|  5|
|b728q0|  Good Will Hunting|  2|
|b728q0|    The Incredibles|  5|
|b728q0|The Shawshank Red...|  2|
|f540n1|  Good Will Hunting|  2|
+-----+-----+-----+
```

Row-Based Data Format (cont.)

```
+-----+-----+-----+
|userId|          variable|rating|
+-----+-----+-----+
z097s3 |z097s3| Good Will Hunting| 2 |
|----->|z097s3| Batman Forever| 3 |
|----->|z097s3|The Shawshank Red...| 4 |
|----->|z097s3|           Coco| 4 |
z176c4 |z176c4| Good Will Hunting| 1 |
|----->|z176c4| The Incredibles| 4 |
|----->|z176c4|The Shawshank Red...| 3 |
|----->|z176c4|           Coco| 4 |
m821i6 |m821i6| Good Will Hunting| 3 |
|----->|m821i6| Batman Forever| 4 |
|----->|m821i6|The Shawshank Red...| 3 |
|----->|m821i6|           Coco| 5 |
t872c7 |t872c7| Good Will Hunting| 1 |
|----->|t872c7| Batman Forever| 2 |
|----->|t872c7| The Incredibles| 4 |
|----->|t872c7|The Shawshank Red...| 5 |
b728q0 |b728q0| Good Will Hunting| 2 |
|----->|b728q0| The Incredibles| 5 |
|----->|b728q0|The Shawshank Red...| 2 |
+-----+-----+-----+
```

```
df.printSchema()
```

```
root
| -- userId: string (nullable = true)
| -- variable: string (nullable = false)
| -- rating: long (nullable = true)
```

Must Be Integers

```
df.printSchema()
```

```
root
|--- userId: string (nullable = true)
|--- variable: string (nullable = false)
|--- rating: long (nullable = true)
```

Must be integers!

Row-Based Data Format

userId	variable	rating
z097s3	Good Will Hunting	2
z097s3	Batman Forever	3
z097s3	The Shawshank Red...	4
z097s3	Coco	4
z176c4	Good Will Hunting	1
z176c4	The Incredibles	4
z176c4	The Shawshank Red...	3
z176c4	Coco	4
m821i6	Good Will Hunting	3
m821i6	Batman Forever	4
m821i6	The Shawshank Red...	3
m821i6	Coco	5
t872c7	Good Will Hunting	1
t872c7	Batman Forever	2
t872c7	The Incredibles	4
t872c7	The Shawshank Red...	5
b728q0	Good Will Hunting	2
b728q0	The Incredibles	5
b728q0	The Shawshank Red...	2
f540n11	Good Will Hunting	2

Conventional Dataframe

```
ratings.show()
```

userId	Good Will H...	Batman For...	Incredibles	Shawshank Redemption	Coco
z097s3	2	3	null	4	4
z176c4	1	null	4	3	4
m821i6	3	4	null	3	5
t872c7	1	2	4	5	null
b728q0	2	null	5	2	null
f540n1	2	1	null	3	1
w066f1	5	null	5	2	5
v081u6	1	null	5	1	1
j197o6	3	2	2	4	null
n202j1	2	null	2	null	2
p755a0	2	3	4	5	5
t791a0	5	5	null	1	4
c460j6	4	1	null	4	4
z595b3	1	2	4	null	1
h296x8	4	3	5	2	4
a610z0	2	1	null	4	4
g025o2	5	4	2	2	null
u902e2	null	3	4	1	5
t893x2	1	4	null	null	5
x668y8	2	3	5	2	null

Wide to Long Function

```
# Function to convert conventional datafame into row-based ("long") dataframe  
wide_to_long
```

```
<function __main__.to_long>
```

```
# Function to convert conventional datafame into row-based ("long") dataframe
long_ratings = wide_to_long(ratings)
long_ratings.show()
```

```
+-----+-----+
|userId|      variable|rating|
+-----+-----+
|z097s3|  Good Will Hunting|    2|
|z097s3|    Batman Forever|    3|
|z097s3|The Shawshank Red...|    4|
|z097s3|          Coco|    4|
|z176c4|  Good Will Hunting|    1|
|z176c4|    The Incredibles|    4|
|z176c4|The Shawshank Red...|    3|
|z176c4|          Coco|    4|
|m821i6|  Good Will Hunting|    3|
|m821i6|    Batman Forever|    4|
|m821i6|The Shawshank Red...|    3|
|m821i6|          Coco|    5|
|t872c7|  Good Will Hunting|    1|
|t872c7|    Batman Forever|    2|
|t872c7|    The Incredibles|    4|
|t872c7|The Shawshank Red...|    5|
|b728q0|  Good Will Hunting|    2|
|b728q0|    The Incredibles|    5|
|b728q0|The Shawshank Red...|    2|
|f540n1|  Good Will Hunting|    2|
+-----+-----+
```

Steps to Get Integer Id's

1. Extract unique userIds and movieIds
2. Assign unique integers to each id
3. Rejoin unique integer id's back to the ratings data

Extracting Distinct User Ids

```
users = long_ratings.select('userId').distinct()  
user.show()
```

```
+-----+  
|userId|  
+-----+  
|j197o6|  
|m821i6|  
|g025o2|  
|z176c4|  
|a610z0|  
|c460j6|  
|w066f1|  
|v081u6|  
|t791a0|  
|f540n1|  
|n202j1|  
|t872c7|  
|h296x8|  
|p755a0|  
|t893x2|  
|u902e2|  
|z097s3|  
|z595b3|  
+-----+
```

Monotonically Increasing ID

```
from pyspark.sql.functions import monotonically_increasing_id
```

Coalesce Method

```
from pyspark.sql.functions import monotonically_increasing_id  
users = users.coalesce(1)
```

Persist Method

```
from pyspark.sql.functions import monotonically_increasing_id
users = users.coalesce(1)
users = users.withColumn(
    "userIntId", monotonically_increasing_id()).persist()
users.show()
```

```
+----+-----+
|userId|userIntId|
+----+-----+
|j197o6|      0|
|m821i6|      1|
|g025o2|      2|
|z176c4|      3|
|a610z0|      4|
|c460j6|      5|
|w066f1|      6|
|v081u6|      7|
|t791a0|      8|
|f540n1|      9|
|n202j1|     10|
|t872c7|     11|
|h296x8|     12|
|p755a0|     13|
|t893x2|     14|
+----+-----+
```

Movie Integer Ids

```
movies = long_ratings.select("variable").distinct()
movies = movies.coalesce(1)
movies = movies.withColumn(
    "movieId", monotonically_increasing_id()).persist()
movies.show()
```

```
+-----+-----+
|      variable|movieId|
+-----+-----+
| The Incredibles|      0 |
|          Coco|      1 |
| The Shawshank Red...|      2 |
|   Good Will Hunting|      3 |
|     Batman Forever|      4 |
+-----+-----+
```

Joining UserIds and Movields

```
ratings_w_int_ids = long_ratings.join(  
    users, "userId", "left").join(movies, "variable", "left")  
  
ratings_w_int_ids.show()
```

```
+-----+-----+-----+-----+  
|       variable|userId|rating|userIntId|movieId|  
+-----+-----+-----+-----+  
| Good Will Hunting|z097s3|    2|     16|      3|  
| Batman Forever|z097s3|    3|     16|      4|  
| The Shawshank Red...|z097s3|    4|     16|      2|  
|           Coco|z097s3|    4|     16|      1|  
| Good Will Hunting|z176c4|    1|      3|      3|  
| The Incredibles|z176c4|    4|      3|      0|  
| The Shawshank Red...|z176c4|    3|      3|      2|  
|           Coco|z176c4|    4|      3|      1|  
| Good Will Hunting|m821i6|    3|      1|      3|  
| Batman Forever|m821i6|    4|      1|      4|  
| The Shawshank Red...|m821i6|    3|      1|      2|  
|           Coco|m821i6|    5|      1|      1|  
| Good Will Hunting|t872c7|    1|     11|      3|  
| Batman Forever|t872c7|    2|     11|      4|  
| The Incredibles|t872c7|    4|     11|      0|  
| The Shawshank Red...|t872c7|    5|     11|      2|  
+-----+-----+-----+-----+
```

```
from pyspark.ml.functions import col  
  
ratings_data = ratings_w_int_ids.select(  
    col("userIntId").alias("userid"),  
    col("variable").alias("movieId"),  
    col("rating"))  
  
ratings_data.show()
```

```
+----+----+----+  
|userId|movieId|rating|  
+----+----+----+  
|   16|      3|     2|  
|   16|      4|     3|  
|   16|      2|     4|  
|   16|      1|     4|  
|    3|      3|     1|  
|    3|      0|     4|  
|    3|      2|     3|  
|    3|      1|     4|  
|    1|      3|     3|  
|    1|      4|     4|  
|    1|      2|     3|  
|    1|      1|     5|  
|   11|      3|     1|  
|   11|      4|     2|  
|   11|      0|     4|  
|   11|      2|     5|  
+----+----+----+
```



BUILDING RECOMMENDATION ENGINES WITH PYSPARK

Let's practice!



BUILDING RECOMMENDATION ENGINES WITH PYSPARK

ALS Parameters and Hyperparameters

Jamen Long
Data Scientist

Example ALS Model Code

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Column Names

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Arguments

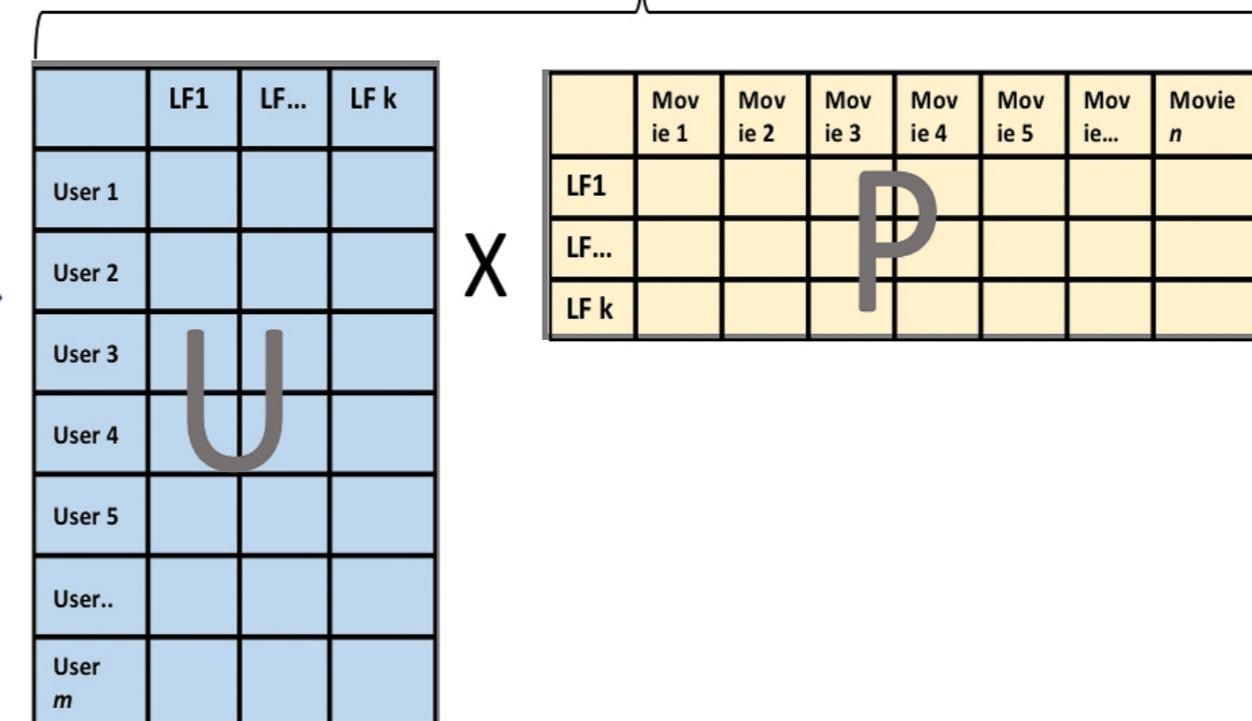
- userCol: Name of column that contains user id's
- itemCol: Name of column that contains item id's
- ratingCol: Name of column that contains ratings

Original Ratings Matrix

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6	Movie 7	Movie ...	Movie n
User 1									
User 2									
User 3									
User 4									
User 5									
User...									
User m									

ALS

Factor Matrices

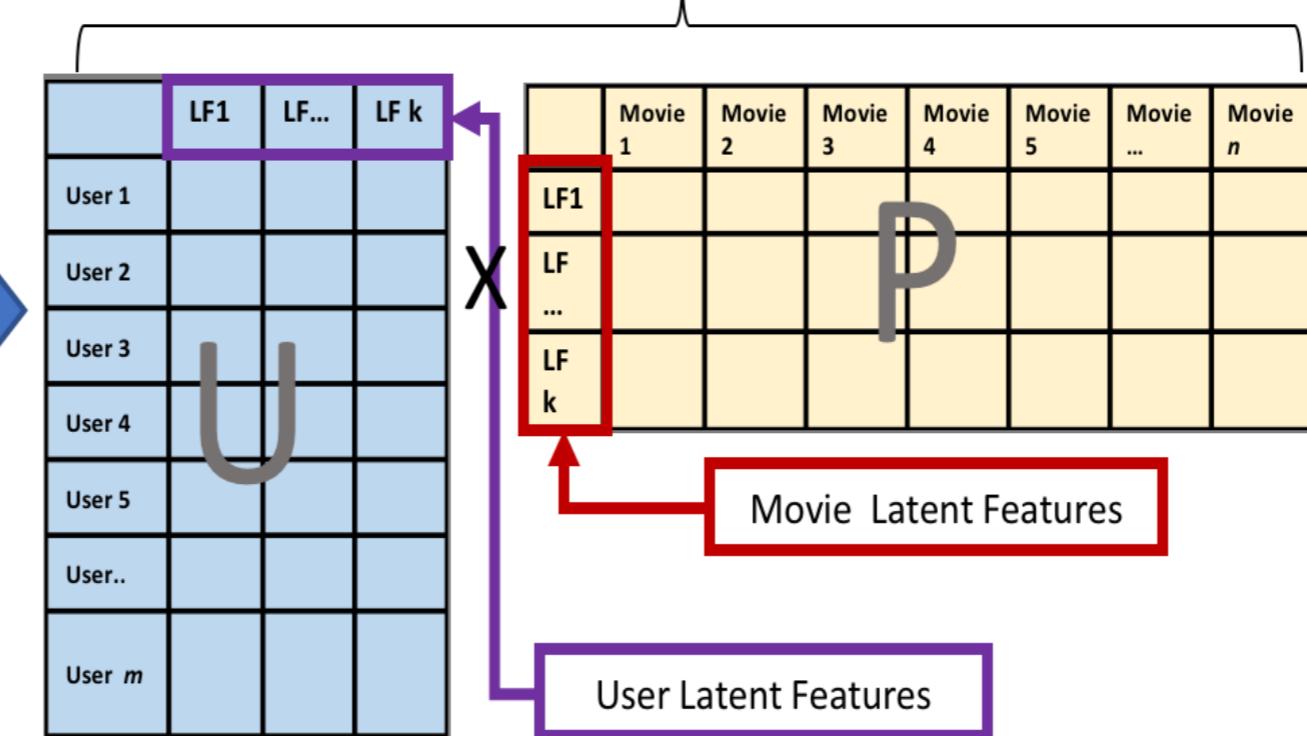


Original Ratings Matrix

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6	Movie 7	Movie ...	Movie n
User 1									
User 2									
User 3									
User 4									
User 5									
User...									
User m									

ALS

Factor Matrices



Rank

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Hyperparameters

- rank, k : number of latent features

MaxIter

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Hyperparameters

- rank, k : number of latent features
- maxIter: number of iterations

RegParam

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Hyperparameters

- rank, k : number of latent features
- maxIter: number of iterations
- regParam: Lambda

Alpha

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Hyperparameters

- rank, k : number of latent features
- maxIter: number of iterations
- regParam: Lambda
- alpha: Discussed later. Only used with implicit ratings.

Non-Negative

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Additional Arguments

- `nonnegative = True`: Ensures positive numbers

Cold Start Strategy

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Additional Arguments

- `nonnegative = True`: Ensures positive numbers
- `coldStartStrategy = "drop"`: Addresses issues with test/train split

Implicit Preferences

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Additional Arguments

- `nonnegative = True`: Ensures positive numbers
- `coldStartStrategy = "drop"`: Addresses issues with test/train split
- `implicitPrefs = True`: True/False depending on ratings type

Sample ALS Model Build

```
als = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
          rank=25, maxIter=100, regParam=.05,
          nonnegative=True,
          coldStartStrategy="drop",
          implicitPrefs=False)
```

Fit and Transform Methods

```
# Fit ALS to training dataset  
model = als.fit(training_data)  
  
# Generate predictions on test dataset  
predictions = model.transform(test_data)
```



BUILDING RECOMMENDATION ENGINES WITH PYSPARK

Let's practice!