1 a. 25*20 Random Table with 200 cells of 1-7 values and 300 cells unfilled.

```
Random_array = sprandn(25,20,0.5132) //Initialize the table with needed density
Num_range=randi([1,7],25,20)//Specifying the range of integers : 1-7
Random_array(find(Random_array))=Num_range(find(Random_array))//Replace the inital table with integer array
```

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
7	0	4	4	0	0	2	1	0	0	5	2	5	7	7	0	0	0	5	5
0	2	1	0	0	0	3	5	0	2	1	0	1	3	0	0	0	0	0	6
0	0	0	0	2	6	6	4	7	0	0	0	0	0	6	0	0	7	0	2
6	0	0	0	0	0	0	7	0	0	0	1	0	0	4	0	5	1	4	2
0	0	6	0	0	0	0	0	0	0	0	1	0	0	0	0	0	3	5	4
7	0	0	0	5	6	4	1	2	7	4	0	1	3	0	2	4	0	0	0
0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	4	0	3
2	0	5	0	0	3	0	2	7	7	0	0	0	1	5	0	6	0	0	1
0	3	0	0	0	0	0	5	7	1	0	5	0	0	0	4	0	0	0	7
3	2	0	0	5	5	3	1	0	2	0	0	0	0	1	1	7	7	1	0
0	0	0	3	2	0	1	7	0	0	1	6	0	0	0	0	0	0	3	5
0	0	0	0	0	0	0	5	0	0	7	0	0	0	6	2	4	7	5	0
4	1	3	0	0	5	0	0	0	0	0	0	0	2	0	0	0	0	4	0
0	0	0	6	7	0	6	4	0	0	1	4	0	3	6	0	0	0	0	5
0	5	2	0	0	4	2	4	2	0	4	1	0	0	0	6	0	0	0	0
0	0	0	1	0	0	0	5	7	4	0	0	0	0	3	0	0	0	4	3
0	0	0	0	0	0	7	4	0	0	0	0	0	0	1	2	0	0	7	0
5	0	0	0	7	0	0	0	3	3	0	0	1	0	4	7	6	3	2	5
0	1	0	4	0	0	3	0	6	0	0	0	0	0	5	0	0	7	7	2
7	0	0	4	7	0	0	0	0	0	0	0	0	0	5	0	0	4	0	0
1	6	0	0	4	6	1	0	0	0	0	6	6	4	0	2	0	3	0	4
0	7	0	0	6	4	0	0	3	0	7	0	2	0	0	0	1	0	4	0
0	0	0	0	0	0	5	4	3	2	0	0	0	6	0	4	0	6	0	0
0	0	0	0	0	1	0	1	3	0	2	6	4	0	0	0	0	0	0	0
0	0	0	5	7	6	0	6	1	4	0	6	0	4	0	0	0	0	3	0

1 b. Splitting the above 25*20 table with 200 non-zero values into training set with 150 non-zero values and 50 non-zero values as testing set

Training Set : 150

```
p=0.22%; //Percentage of Training in the entire data
A_ones=find(Train_set) //Finding the cells with non zero values
A_ones_change=A_ones(rand(size(A_ones))<=p) //Getting random 150 cells
Train_set(A_ones_change)=0 // Setting the other values to 0
```

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
7	0	4	4	0	0	2	1	0	0	5	0	5	7	7	0	0	0	0	5
0	2	0	0	0	0	3	5	0	2	1	0	1	3	0	0	0	0	0	0
0	0	0	0	2	0	6	4	7	0	0	0	0	0	6	0	0	7	0	0
6	0	0	0	0	0	0	7	0	0	0	1	0	0	4	0	5	1	4	2
0	0	6	0	0	0	0	0	0	0	0	1	0	0	0	0	0	3	5	0
7	0	0	0	5	6	4	1	2	7	4	0	1	3	0	2	4	0	0	0
0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3
2	0	5	0	0	3	0	2	7	7	0	0	0	0	5	0	6	0	0	1
0	3	0	0	0	0	0	5	7	1	0	5	0	0	0	4	0	0	0	0
3	2	0	0	5	0	3	0	0	2	0	0	0	0	1	1	7	7	1	0
0	0	0	3	2	0	1	7	0	0	1	6	0	0	0	0	0	0	3	0
0	0	0	0	0	0	0	5	0	0	0	0	0	0	6	2	4	7	5	0
4	1	0	0	0	5	0	0	0	0	0	0	0	2	0	0	0	0	4	0
0	0	0	0	7	0	6	4	0	0	1	4	0	3	6	0	0	0	0	5
0	0	2	0	0	4	0	4	2	0	4	1	0	0	0	6	0	0	0	0
0	0	0	1	0	0	0	5	0	4	0	0	0	0	3	0	0	0	4	0
0	0	0	0	0	0	7	4	0	0	0	0	0	0	1	2	0	0	7	0
5	0	0	0	0	0	0	0	3	0	0	0	0	0	4	7	0	3	2	5
0	1	0	4	0	0	3	0	6	0	0	0	0	0	0	0	0	7	0	2
7	0	0	4	7	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0
1	6	0	0	4	0	1	0	0	0	0	0	6	4	0	2	0	3	0	4
0	7	0	0	6	4	0	0	3	0	0	0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	5	4	0	2	0	0	0	6	0	4	0	6	0	0
0	0	0	0	0	1	0	1	3	0	2	6	0	0	0	0	0	0	0	0
0	0	0	5	7	6	0	6	1	0	0	6	0	4	0	0	0	0	3	0

Testing Set : 50

```
Test_set = Total_data - Train_set //Subtracting Entire data from training data
```

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	5	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0

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

1 c. Tool Box : MatlabALS , Data : Training_Set(200 values), Latent Features : 2 , Matlab Function : 'nnmf'

```
opt=statset('Maxiter',10,'Display',"final"); //Various parameters for the nnmf method
[W0,H0]=nnmf(Train_set,2,'replicates',5,"options",opt,"algorithm","als")//Applying the ALS to training set
```

W :

LATENT_W11	LATENT_W12
0	12.49508157
3.837458705	1.252364009
10.46449992	0.681195225
6.634914346	3.700862887
3.082213478	0.077343874
0.535661636	12.37274639
0	1.163169552
4.159262173	5.813847836
7.720417975	0
3.931132664	5.72829394
7.948318768	0
9.663757627	0
0.300874192	4.153556971
6.496046109	5.07121886
4.083687617	1.466178469
5.468955383	0
7.682872342	0
3.46421347	5.000414115
5.29683289	0.971553143
0.198069807	7.378109975
0.708790071	6.679350608
2.451527489	3.476083094
6.470636369	1.92440211
3.215062757	0
7.062855046	3.811154107

H:

H1	0	0.06	0.03	0.08	0.14	0.05	0.29	0.6	0.299	0.059	0	0.267	0	0.052	0.281	0.176	0.116	0.356	0.322	0.040863
H2	0.56	0.12	0.13	0.14	0.37	0.24	0.15	0.01	0.081	0.224	0.2076799	0	0.209	0.322	0.225	0.099	0.201	0.096	0.012	0.248802

1 d. Predicting the ratings using the W , H matrices obtained above for the Test_set 50 values

```
Test_set_pred = W0*H0 //Multiplyting Latent Features to get Predicted matrix
B_ones = find(Test_set==0) //Making the cells other than test set values as zeros
Test_set_pred(B_ones)=0
Eone= immse(Test_set,Test_set_pred)//Calculating mean squared error between test set and predicted values for test set.
```

Test Set:

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	5	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
0	0	0	0	0	5	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0
0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	5	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Predicted Set

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
											0.00							0.14	
		0.26																	0.47
					0.72														0.67
																			0.15
																	0.11		
													2.08						
																			0.31
					1.6		2.41												
																			0.32
		0.55																	
			1.24																
	0.41					1.4													
								1.63											0.22
				2.33					1.327			1.04				1.4			
														1.7				1.71	
														1.71					
					1.66						0.18								
										0.72		0.726				0.98			
								2.08											
												0.00							
									1.274										

Generated Test Error : 1.09

2a. 25*20 Table with 200 cells of 1-7 values and 300 cells unfilled in such a way to minimize the error.

Logic: The logic is to have almost similar or similar ratings for all the users on all the items . This means that all of the given users and items are very similar and have predictable rates. Hence the error decreases.

```
Random_array = sprandn(25,20,0.5132) //Initialize the table with needed density
Num_range=randi([3,3],25,20)//Specifying the range of integers : only 3
```

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	3	3	0	0	0	3	3	0	0	3	0	0	3	0	0	3	0
3	0	3	3	3	0	3	0	0	3	0	0	0	3	0	0	0	3	0	0
3	3	3	0	3	0	3	0	0	3	0	0	3	0	3	3	0	0	0	0
0	0	3	0	0	0	0	3	3	0	0	0	0	0	3	3	3	0	3	3
3	0	0	0	0	0	3	3	0	0	3	3	0	0	3	0	0	0	0	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	3	0	3	3	0
0	3	0	0	0	0	3	0	0	0	3	0	0	0	3	0	3	0	3	0
3	0	0	3	0	0	0	3	0	0	3	3	3	3	3	3	0	0	0	0
3	0	0	3	0	0	3	0	0	0	0	3	0	3	3	0	0	0	0	3
0	0	0	3	0	3	3	3	0	3	3	0	0	3	3	0	0	3	0	3
3	0	3	0	3	0	3	0	3	3	0	0	3	3	0	0	0	3	0	3
0	3	0	0	0	0	0	3	3	0	3	3	0	0	3	0	3	0	3	0
0	0	0	0	0	3	3	0	3	0	0	3	0	0	0	0	3	3	3	3
3	0	0	0	0	3	0	3	3	0	0	3	0	0	0	0	0	0	0	0
0	0	0	3	0	0	0	3	3	0	0	0	0	0	0	0	3	0	3	0
0	3	0	0	0	0	3	3	0	3	0	0	0	0	3	0	3	0	3	0
3	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	3	3	3	3
0	3	0	0	3	0	3	0	0	0	3	3	3	3	3	3	0	0	3	0
3	0	3	0	3	0	0	3	0	0	3	3	3	0	3	0	3	0	0	0
3	3	3	3	3	0	3	3	0	0	3	3	0	0	3	3	3	3	0	0
0	0	0	0	0	0	3	0	3	0	3	0	0	0	3	3	3	0	3	0
0	0	0	3	0	3	0	0	0	3	3	3	0	3	0	0	3	3	0	3
0	3	3	3	0	3	0	0	0	0	0	0	3	0	0	3	0	0	3	0
0	3	0	3	0	3	3	0	3	3	3	3	3	3	0	0	3	0	3	0
0	0	3	0	0	3	0	3	0	3	0	3	3	3	0	0	0	3	0	3

2b. Splitting the above 25*20 table with 200 non-zero values into training set with 150 non-zero values and 50 non-zero values as testing set

Training Set : 150

```
p=0.22%; //Percentage of Training in the entire data
A_ones=find(Train_set) //Finding the cells with non zero values
A_ones_change=A_ones(rand(size(A_ones))<=p) //Getting random 150 cells
Train_set(A_ones_change)=0 // Setting the other values to 0
```

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	3	3	0	0	0	3	3	0	0	0	0	0	3	0	0	3	0
3	0	3	3	3	0	0	0	0	3	0	0	0	3	0	0	0	3	0	0
3	0	3	0	3	0	3	0	0	3	0	0	3	0	3	3	0	0	0	0
0	0	3	0	0	0	0	3	3	0	0	0	0	0	3	0	3	0	3	3
3	0	0	0	0	0	3	3	0	0	3	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	3	0	3	3	0
0	3	0	0	0	0	3	0	0	0	3	0	0	0	3	0	0	0	3	0
3	0	0	0	0	0	0	3	0	0	3	0	3	3	0	3	0	0	0	0
3	0	0	3	0	0	3	0	0	0	0	0	0	3	0	0	0	0	0	3
0	0	0	0	0	0	3	3	0	3	3	0	0	3	3	0	0	3	0	3
3	0	3	0	3	0	3	0	3	3	0	0	0	3	0	0	0	3	0	0
0	3	0	0	0	0	0	3	0	0	3	3	0	0	0	0	3	0	3	0
0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	3	3	3	3
0	0	0	0	0	3	0	3	3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	3	0	3	0
0	3	0	0	0	0	0	3	0	3	0	0	0	0	3	0	3	0	3	0
3	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	3	3	0	3
0	3	0	0	3	0	3	0	0	0	0	3	3	3	3	3	0	0	3	0
3	0	3	0	3	0	0	0	0	0	3	3	3	0	3	0	3	0	0	0
3	0	3	0	3	0	0	3	0	0	3	0	0	0	3	0	3	3	0	0
0	0	0	0	0	0	3	0	3	0	3	0	0	0	3	3	3	0	3	0
0	0	0	3	0	0	0	0	0	3	3	0	0	3	0	0	0	3	0	3
0	3	3	3	0	3	0	0	0	0	0	0	3	0	0	3	0	0	3	0
0	3	0	3	0	3	3	0	3	3	3	3	3	3	0	0	0	0	3	0
0	0	0	0	0	3	0	3	0	3	0	3	3	3	0	0	0	0	0	3

Testing Set : 50

```
Test_set = Total_data - Train_set
```

```
Test_set = Total_data - Train_set //Subtracting Entire data from training data
```

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
0	0	0	3	0	0	0	0	0	0	0	3	0	0	3	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	0	0	0	0	0
0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	3	0	0	0	0	0	3	0	0	0	0	0
0	0	0	0	0	3	3	0	0	0	0	3	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
0	0	0	3	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0
0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0
0	3	0	3	0	0	3	0	0	0	0	3	0	0	0	3	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	3	0	0	0	0	0	3	0	0	0	0	3	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0

2.c. **Tool Box** : MatlabALS , **Data** : Training_Set(200 values), **Latent Features** : 2 , **Matlab Function** : 'nnmf'

```
opt=statset('Maxiter',10,'Display',"final"); //Various parameters for the nnmf method
[W0,H0]=nnmf(Train_set,2,'replicates',5,"options",opt,"algorithm","als")//Applying the ALS to training set
```

W :

2.67	2.31
7.32	0
5.06	1.79
0.57	5.47
2.2	2.29
1.32	2.45
0.03	5
3.44	1.83
4.72	0
5.13	2.18
7.23	0
0	6.12
1.08	3.16
0.38	1.56
0	3.85
0.08	5.64
2.93	0.69
2.71	5.19
3.17	3.6
3.66	3.35
0.18	6.36
5.47	0
1.62	3.58
4.57	4.61
4.16	1.16

H:

0.38	0	0.24	0.25	0.28	0.1	0.238	0.073	0.0850638	0.391	0.154	0.085	0.177	0.438	0.055	0.083	0	0.332885	0	0.218397
0	0	0.08	0	0.05	0.07	0.196	0.292	0.1906184	0.02	0.281	0.188	0.143	0	0.351	0.196	0	0	1	0.036315

2.d. **Predicting the ratings using the W , H matrices obtained above for the Test_set 50 values**

```
Test_set_pred = W0*H0 //Multiplyting Latent Features to get Predicted matrix
B_ones = find(Test_set==0) //Making the cells other than test set values as zeros
Test_set_pred(B_ones)=0
Eone= immse(Test_set,Test_set_pred)//Calculating mean squared error between test set and predicted values for test set.
```

Test Set:

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
0	0	0	3	0	0	0	0	0	0	0	3	0	0	3	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	0	0	0	0	0
0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	3	0	0	0	0	0	3	0	0	0	0	0
0	0	0	0	0	3	3	0	0	0	0	3	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
0	0	0	3	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0
0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0
0	3	0	3	0	0	3	0	0	0	0	3	0	0	0	3	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	3	0	0	0	0	0	3	0	0	0	0	3	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0

Predicted Set

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	0	0	0.00	0.803	0	0	0	0	0	0	0.00
0	0	0	0	0	0	1.74	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0.54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.117	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.926	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.905	0	0	0.00
0	0	0	0.87	0	0	0	0	0	0	0	0.6382516	0	0	0.831	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0.4035695	0	0	0.259	0	0	0	0	0.00
0	0	0	1.3	0	0.69	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	1.282	0	0	0	0	0	0	1.58
0	0	0	0	0	0	0	0	1.17	0	0	0	0	0	2.15	0	0	0	0	0.00
0	0	0	0	0	0.33	0.88	0	0	0	0	0.6867434	0	0	0	0	0	0	0	0.00
0.14	0	0	0	0	0	0	0	0	0	0	0.3253381	0	0	0	0	0	0	0	0.00
0	0	0	0.02	0	0	0	0	0.73	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	1.12	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.375	0.00
0	0	0	0	0	0	0	0	0	0	1.875	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	1.28	0	0	0	0	0	0	0	0	0	0	0	0.00
0	1.01	0	0.94	0	0	1.53	0	0	0	0	0.9435646	0	0	0	0.959	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0.57	0	0	0	0	0	0.467114	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0.00	0	0	0	1.754	0	0	0.00
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0.00

Generated Test Error : 0.31

3a. 25*20 Table with 200 cells of 1-7 values and 300 cells unfilled in such a way to maximize the error.

```
Random_array = sprandn(25,20,0.5132) //Initialize the table with needed density
Data is generated plugged in manually for the density of 200 non-zero cells in 500.

The logic behind how data is generted is discussed in 3b at the bottom.
```

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	7	0	6	1	2	3	0	0	0	0	0
1	7	0	1	2	3	4	0	0	0	0	0	5	6	0	5	1	7	0	2
0	6	0	2	0	7	0	0	0	1	0	3	4	5	0	0	0	0	7	0
0	5	0	0	0	1	0	2	6	3	0	0	4	7	6	0	2	0	0	0
0	0	0	0	0	3	0	7	0	5	1	2	0	0	0	0	4	6	1	5
2	0	7	3	0	1	5	7	0	0	0	4	5	0	0	0	0	0	0	0
3	4	0	0	1	7	0	0	0	2	0	3	6	0	4	5	7	2	3	0
0	0	6	0	1	5	0	0	0	2	3	4	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0	6	2	1	7	4	0	1	5	0	6	7
0	0	0	4	7	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
0	2	0	5	0	7	1	0	3	4	0	0	0	0	1	0	6	0	4	0
4	0	0	0	6	1	0	6	7	0	4	0	0	0	0	2	0	0	2	0
0	1	0	0	0	0	7	1	0	5	6	2	0	3	0	0	0	4	5	1
0	0	5	6	1	2	0	7	0	0	0	0	0	0	0	3	0	3	4	0
0	0	0	0	2	4	0	6	0	0	0	0	0	0	0	0	0	0	1	0
5	0	0	4	0	0	0	1	2	4	0	7	2	0	0	0	0	0	3	6
6	0	0	0	0	0	0	1	0	2	0	0	0	0	7	0	0	0	0	3
7	0	0	2	1	0	3	3	0	0	0	0	0	0	0	6	0	0	0	4
1	7	0	1	3	5	0	0	0	0	0	5	0	0	2	1	0	5	0	0
0	0	0	7	0	1	3	0	2	0	4	5	0	0	0	7	0	0	6	0
2	0	0	0	0	0	4	0	6	6	1	0	3	0	5	0	0	1	2	0
0	6	0	0	0	0	1	0	4	0	0	0	2	3	0	6	0	0	5	1
0	0	4	0	0	0	2	1	0	0	0	1	0	0	0	0	3	0	2	2
3	5	0	0	0	6	0	0	1	7	0	7	0	2	4	0	0	0	1	0
0	0	0	4	0	0	7	0	0	0	7	1	1	0	0	3	2	0	7	0

3.a.1. Splitting the above 25*20 table with 200 non-zero values into training set with 150 non-zero values and 50 non-zero values as testing set

Training Set : 150

```
p=0.22%; //Percentage of Training in the entire data
A_ones=find(Train_set) //Finding the cells with non zero values
A_ones_change=A_ones(rand(size(A_ones))<=p) //Getting random 150 cells
Train_set(A_ones_change)=0 // Setting the other values to 0
```

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	7	0	6	1	2	3	0	0	0	0	0
0	7	0	0	2	3	4	0	0	0	0	0	5	6	0	5	1	0	0	2
0	6	0	2	0	7	0	0	0	1	0	0	4	0	0	0	0	0	0	0
0	5	0	0	0	1	0	0	6	0	0	0	0	7	0	0	2	0	0	0
0	0	0	0	0	0	0	7	0	5	1	0	0	0	0	0	0	6	0	5
0	0	7	3	0	1	5	7	0	0	0	4	5	0	0	0	0	0	0	0
3	4	0	0	1	7	0	0	0	0	0	3	6	0	0	5	7	0	3	0
0	0	6	0	0	5	0	0	0	2	3	4	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0	6	0	1	0	0	0	0	0	0	6	7
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
0	2	0	5	0	7	0	0	3	4	0	0	0	0	1	0	6	0	4	0
4	0	0	0	6	1	0	6	7	0	4	0	0	0	0	2	0	0	2	0
0	1	0	0	0	0	7	1	0	5	6	2	0	3	0	0	0	0	5	1
0	0	5	0	1	2	0	0	0	0	0	0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
5	0	0	4	0	0	0	1	0	4	0	0	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	2	0	0	0	0	7	0	0	0	0	0
7	0	0	2	1	0	3	3	0	0	0	0	0	0	0	0	0	0	0	0
1	7	0	1	3	0	0	0	0	0	0	5	0	0	2	1	0	0	0	0
0	0	0	0	0	1	3	0	2	0	4	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	4	0	0	0	1	0	3	0	5	0	0	1	2	0
0	0	0	0	0	0	1	0	4	0	0	0	0	3	0	0	0	0	5	0
0	0	4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	1	0	0	7	0	2	4	0	0	0	1	0
0	0	0	4	0	0	7	0	0	0	7	1	1	0	0	3	0	0	7	0

Testing Set : 50

```
Test_set = Total_data - Train_set //Subtracting Entire data from training data
```

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	5	0	0	0	0	7	0
0	0	0	0	0	0	0	2	0	3	0	0	4	0	6	0	0	0	0	0
0	0	0	0	0	3	0	0	0	0	0	2	0	0	0	0	4	0	1	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	2	0	0	0	0	4	0	0	2	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	2	0	7	4	0	1	5	0	0	0
0	0	0	4	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0
0	0	0	6	0	0	0	7	0	0	0	0	0	0	0	3	0	0	4	0
0	0	0	0	2	4	0	6	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	2	0	0	7	0	0	0	0	0	0	3	6
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	4
0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	5	0	0
0	0	0	7	0	0	0	0	0	0	0	5	0	0	0	7	0	0	6	0
0	0	0	0	0	0	0	0	6	6	0	0	0	0	0	0	0	0	0	0
0	6	0	0	0	0	0	0	0	0	0	0	2	0	0	6	0	0	0	1
0	0	0	0	0	0	2	0	0	0	0	1	0	0	0	0	3	0	0	2
3	5	0	0	0	6	0	0	0	7	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0

3c. Tool Box : MatlabALS , Data : Training_Set(200 values), Latent Features : 2 , Matlab Function : 'nnmf'

```
opt=statset('Maxiter',10,'Display',"final"); //Various parameters for the nnmf method
[W0,H0]=nnmf(Train_set,2,'replicates',5,"options",opt,"algorithm","als")//Applying the ALS to training set
```

W :

1.55	3.88
9.92	1.02
8.93	0.00
6.26	0.00
0.00	6.09
1.80	7.16
12.60	0.00
3.00	3.00
1.77	5.48
0.00	0.84
7.94	2.40
1.62	5.99
0.00	10.79
1.16	0.86
0.00	0.42
1.16	2.98
0.05	1.76
0.04	3.78
5.37	0.47
0.00	3.30
0.65	4.13
0.95	3.04
0.00	1.83
1.51	2.24
0.00	10.29

H:

0.09	0.54	0.04	0.12	0.11	0.53	0.018	0	0.1359751	0.085	0	0.175	0.339	0.213	0.048	0.235	0.327	0	0.107567	0.03574
0.14	0	0.16	0.17	0.07	0.01	0.471	0	0.1198142	0.347	0	0.214	0.088	0.083	0.107	0.056	0	0	0.418121	0.162416

3d. Predicting the ratings using the W , H matrices obtained above for the Test_set 50 values

```
Test_set_pred = W0*H0 //Multiplyting Latent Features to get Predicted matrix
B_ones = find(Test_set==0) //Making the cells other than test set values as zeros
Test_set_pred(B_ones)=0
Eone= immse(Test_set,Test_set_pred)//Calculating mean squared error between test set and predicted values for test set.
```

Test Set:

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	5	0	0	0	0	7	0
0	0	0	0	0	0	0	2	0	3	0	0	4	0	6	0	0	0	0	0
0	0	0	0	0	3	0	0	0	0	0	2	0	0	0	0	4	0	1	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	2	0	0	0	0	4	0	0	2	0	0

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

Predicted Set

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	0	0	0.00	0	0	0	0	0	0	0	0.00
1.03	0	0	1.32	0	0	0	0	0	0	0	0	0	0	0	0	0	0.094	0	0.00
0	0	0	0	0	0	0	0	0	0	0	1.5608881	0	1.898	0	0	0	0	0.961	0.00
0	0	0	0	0	0	0	0	0	0.532	0	0	2.123	0	0.303	0	0	0	0	0.00
0	0	0	0	0	0.05	0	0	0	0	0	1.3010449	0	0	0	0	0	0	2.547	0.00
1.15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	1.07	0	0	0	0	0.61	0	0	0	0	0.00
0	0	0	0	0.54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	2.247	0	1.084	0.83	0	0.721	0.579	0	0	0.00
0	0	0	0.15	0.06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	1.27	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.997	0	0.00
0	0	0	0.28	0	0	0	0.29	0	0	0	0	0	0	0	0.321	0	0	0.486	0.00
0	0	0	0	0.03	0	0	0.14	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0.51	0	0	0.8386572	0	0	0	0	0	0	1.37	0.53
0.25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.29
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.22	0	0	0	0.62
0	0	0	0	0	2.86	0	0	0	0	0	0	0	0	0	0	0	0.044	0	0.00
0	0	0	0.58	0	0	0	0	0	0	0	0.7052514	0	0	0	0.183	0	0	1.381	0.00
0	0	0	0	0	0	0	0	0.58	1.487	0	0	0	0	0	0	0	0	0	0.00
0	0.51	0	0	0	0	0	0	0	0	0	0	0.59	0	0	0.392	0	0	0	0.53
0	0	0	0	0	0	0.86	0	0	0	0	0.3910278	0	0	0	0	0	0	0	0.30
0.44	0.81	0	0	0	0.82	0	0	0	0.903	0	0	0.00	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00

Generated Test Error : 1.93

3b. Comments on the Data distribution of questions 2 and 3

The logic behind is that the way the data is distributed effects the final error .

In the question 2, all the ratings are similar or amost similar, hence making all the users and the items similar or ones that exhibit similar behaviour. Hence the error is very small as the predicted values are very much near to the original values. Imagine a popular item generally liked by everyone, as it always has the ratings towards higher end , its prediction among various users usually can be same.

In the question 3, the scenario is just the opposite, here the ratings are distributed such a way that it becomes really difficult to fit the ratings onto a line, as there won't be any pattern. Imagine a movie being liked and disliked in equal proportions or an item having mixed ratings through out or a user giving mixed ratings to smilar items. This makes the predictions difficult and hence the increase in error.

4a. 25*20 Table with 200 cells of 1-7 values and 300 cells unfilled in such a way to maximize the error - Used the same table as generated in 3

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	7	0	6	1	2	3	0	0	0	0	0
1	7	0	1	2	3	4	0	0	0	0	0	5	6	0	5	1	7	0	2
0	6	0	2	0	7	0	0	0	1	0	3	4	5	0	0	0	0	7	0
0	5	0	0	0	1	0	2	6	3	0	0	4	7	6	0	2	0	0	0
0	0	0	0	0	3	0	7	0	5	1	2	0	0	0	0	4	6	1	5
2	0	7	3	0	1	5	7	0	0	0	4	5	0	0	0	0	0	0	0
3	4	0	0	1	7	0	0	0	2	0	3	6	0	4	5	7	2	3	0
0	0	6	0	1	5	0	0	0	2	3	4	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0	6	2	1	7	4	0	1	5	0	6	7
0	0	0	4	7	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
0	2	0	5	0	7	1	0	3	4	0	0	0	0	1	0	6	0	4	0
4	0	0	0	6	1	0	6	7	0	4	0	0	0	0	2	0	0	2	0
0	1	0	0	0	0	7	1	0	5	6	2	0	3	0	0	0	4	5	1
0	0	5	6	1	2	0	7	0	0	0	0	0	0	0	3	0	3	4	0
0	0	0	0	2	4	0	6	0	0	0	0	0	0	0	0	0	0	1	0
5	0	0	4	0	0	0	1	2	4	0	7	2	0	0	0	0	0	3	6
6	0	0	0	0	0	0	1	0	2	0	0	0	0	7	0	0	0	0	3
7	0	0	2	1	0	3	3	0	0	0	0	0	0	0	6	0	0	0	4
1	7	0	1	3	5	0	0	0	0	0	5	0	0	2	1	0	5	0	0
0	0	0	7	0	1	3	0	2	0	4	5	0	0	0	7	0	0	6	0
2	0	0	0	0	0	4	0	6	6	1	0	3	0	5	0	0	1	2	0
0	6	0	0	0	0	1	0	4	0	0	0	2	3	0	6	0	0	5	1
0	0	4	0	0	0	2	1	0	0	0	1	0	0	0	0	3	0	2	2
3	5	0	0	0	6	0	0	1	7	0	7	0	2	4	0	0	0	1	0
0	0	0	4	0	0	7	0	0	0	7	1	1	0	0	3	2	0	7	0

4.a.2. Splitting the above 25*20 table with 200 non-zero values into training set with 150 non-zero values and 50 non-zero values as testing set - Same split as 3

Training Set : 150

```
p=0.22%; //Percentage of Training in the entire data
A_ones=find(Train_set) //Finding the cells with non zero values
A_ones_change=A_ones(rand(size(A_ones))<=p) //Getting random 150 cells
Train_set(A_ones_change)=0 // Setting the other values to 0
```

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	7	0	6	1	2	3	0	0	0	0	0
0	7	0	0	2	3	4	0	0	0	0	0	5	6	0	5	1	0	0	2
0	6	0	2	0	7	0	0	0	1	0	0	4	0	0	0	0	0	0	0
0	5	0	0	0	1	0	0	6	0	0	0	0	7	0	0	2	0	0	0
0	0	0	0	0	0	0	7	0	5	1	0	0	0	0	0	0	6	0	5
0	0	7	3	0	1	5	7	0	0	0	4	5	0	0	0	0	0	0	0
3	4	0	0	1	7	0	0	0	0	0	3	6	0	0	5	7	0	3	0
0	0	6	0	0	5	0	0	0	2	3	4	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0	6	0	1	0	0	0	0	0	0	6	7
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
0	2	0	5	0	7	0	0	3	4	0	0	0	0	1	0	6	0	4	0
4	0	0	0	6	1	0	6	7	0	4	0	0	0	0	2	0	0	2	0
0	1	0	0	0	0	7	1	0	5	6	2	0	3	0	0	0	0	5	1
0	0	5	0	1	2	0	0	0	0	0	0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
5	0	0	4	0	0	0	1	0	4	0	0	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	2	0	0	0	0	7	0	0	0	0	0
7	0	0	2	1	0	3	3	0	0	0	0	0	0	0	0	0	0	0	0
1	7	0	1	3	0	0	0	0	0	0	5	0	0	2	1	0	0	0	0
0	0	0	0	0	1	3	0	2	0	4	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	4	0	0	0	1	0	3	0	5	0	0	1	2	0
0	0	0	0	0	0	1	0	4	0	0	0	0	3	0	0	0	0	5	0
0	0	4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	1	0	0	7	0	2	4	0	0	0	1	0
0	0	0	4	0	0	7	0	0	0	7	1	1	0	0	3	0	0	7	0

Testing Set : 50 - Same sets as 3

```
Test_set = Total_data - Train_set //Subtracting Entire data from training data
```

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	5	0	0	0	0	7	0
0	0	0	0	0	0	0	2	0	3	0	0	4	0	6	0	0	0	0	0
0	0	0	0	0	3	0	0	0	0	0	2	0	0	0	0	4	0	1	0

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

4.a. c. Tool Box : MatlabALS , Data : Training_Set(200 values), Latent Features : 4 , Matlab Function : 'nnmf'

```
opt=statset('Maxiter',10,'Display',"final"); //Various parameters for the nnmf method
[W0,H0]=nnmf(Train_set,4,'replicates',5,"options",opt,"algorithm","als")//Applying the ALS to training set
```

W :

0.33	0.00	9.17	0.00
9.85	2.72	0.01	0.00
8.92	0.00	0.17	0.00
6.12	0.67	0.14	0.00
0.00	0.00	5.80	6.33
0.32	0.88	1.23	10.04
12.45	0.32	0.53	0.22
2.20	0.11	2.79	2.72
1.19	2.94	7.43	0.00
0.02	1.13	0.17	0.00
7.77	1.74	2.69	0.00
0.64	2.33	0.00	8.14
0.00	10.23	4.53	0.00
0.72	0.00	0.08	3.01
0.01	0.57	0.09	0.00
0.52	0.00	2.51	3.13
0.00	0.00	4.14	0.59
0.00	1.05	0.00	5.44
4.69	0.00	3.27	0.20
0.03	4.03	0.00	0.56
0.24	3.62	1.60	1.16
0.86	4.00	0.36	0.00
0.00	0.55	0.13	2.28
0.60	0.07	5.15	0.00
0.00	12.73	0.00	0.00

H:

0.09	0.54	0.03	0.11	0.10	0.55	0.02	0.00	0.13	0.01	0.00	0.10	0.35	0.20	0.00	0.25	0.34	0.00	0.09	0.01
0.03	0.03	0.00	0.15	0.03	0.00	0.60	0.01	0.11	0.05	0.52	0.00	0.06	0.14	0.00	0.14	0.00	0.00	0.54	0.03
0.00	0.09	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.70	0.00	0.49	0.00	0.08	0.37	0.00	0.00	0.11	0.15	0.29
0.34	0.00	0.41	0.17	0.22	0.11	0.19	0.68	0.21	0.00	0.12	0.06	0.20	0.00	0.00	0.05	0.00	0.16	0.00	0.04

4.a.d. Predicting the ratings using the W , H matrices obtained above for the Test_set 50 values

```
Test_set_pred = W0*H0 //Multiplyting Latent Features to get Predicted matrix
B_ones = find(Test_set==0) //Making the cells other than test set values as zeros
Test_set_pred(B_ones)=0
Eone= immse(Test_set,Test_set_pred)//Calculating mean squared error between test set and predicted values for test set.
```

Test Set:

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	5	0	0	0	0	7	0
0	0	0	0	0	0	0	2	0	3	0	0	4	0	6	0	0	0	0	0
0	0	0	0	0	3	0	0	0	0	0	2	0	0	0	0	4	0	1	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	2	0	0	0	0	4	0	0	2	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	2	0	7	4	0	1	5	0	0	0
0	0	0	4	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0

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

Predicted Set

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Col18	Col19	Col20
0	0	0	0	0	0	0	0	0	0	0	0.00	0	0	0	0	0	0	0	0.00
0.96	0	0	1.52	0	0	0	0	0	0	0	0	0	0	0	0	0	0.001	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0.94	0	1.829	0	0	0	0	0.815	0.00
0	0	0	0	0	0	0	0.01	0	0.199	0	0	2.154	0	0.053	0	0	0	0	0.00
0	0	0	0	0	0.68	0	0	0	0	0	3.22	0	0	0	0	0	0	0.89	0.00
3.423	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0.512	0	0	0	0	0.195	0	0	0.091	0	0.00
0	0	0	0	0.83	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	1.528	0	0.572	1.275	0	0.716	0.405	0	0	0.00
0	0	0	0.17	0.03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	1.18	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.482	0	0.00
0	0	0	0.59	0	0	0	2.05	0	0	0	0	0	0	0	0.324	0	0	0.075	0.00
0	0	0	0	0.02	0	0	0.01	0	0	0	0	0	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0.72	0	0	1.47	0	0	0	0	0	0	0.433	0.85
0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.22
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.414	0	0	0	0.23
0	0	0	0	0	2.6	0	0	0	0	0	0	0	0	0	0	0	0.38	0	0.00
0	0	0	0.69	0	0	0	0	0	0	0	0.04	0	0	0	0.61	0	0	2.172	0.00
0	0	0	0	0	0	0	0	0.67	1.317	0	0	0	0	0	0	0	0	0	0.00
0	0.612	0	0	0	0	0	0	0	0	0	0	0.516	0	0	0.785	0	0	0	0.24
0	0	0	0	0	0	0.77	0	0	0	0	0.2	0	0	0	0	0	0	0	0.14
0.056	0.773	0	0	0	0.33	0	0	0	3.599	0	0	0.00	0	0	0	0	0	0	0.00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00

Generated Test Error : 2.13

4b. Comments on the predicted values and error between the models having latent feauters as 2 Vs 4

Latent features are the aggregated observable features in the dataset also called as hidden features

As the number of latent feauters increase more than 2, there appears to be a raise in the test error as we see in our analysis. Although the training error descreases as the latent features increase, because of the over fitting , the test error eventually increases. Few approaches like regularization are used to decrease the effect of this over fitting in such scenarios.

5a. Eigen Values and Eigen Vectors

ToolBox : Matlab , Function: SVD, Output : U, S, V :: U and V left and right sigular vectors are the set of orthonormal eigen vectors of AA* & A*A

S : Eigen values are the squares of the singular values obtained as the diagonal elements of S.

[U,S,V] = svd(sv)

U:

-0.19	0.02	0.14	-0.26	-0.15	-0.09	0.29	-0.51	0.16	-0.10	-0.04	0.37	0.08	0.07	-0.28	-0.16	0.05	0.17	-0.29	0.00	-0.04	-0.10	0.00	-0.06	-0.25	0.09
-0.19	-0.02	0.20	-0.37	-0.17	0.01	0.12	0.02	-0.50	0.17	-0.02	-0.12	-0.17	0.33	0.21	0.21	0.04	0.32	0.06	-0.15	-0.25	0.15	0.02	-0.02	0.08	-0.04
-0.24	-0.11	0.18	-0.27	-0.06	-0.45	-0.38	0.03	0.17	0.04	-0.04	0.15	-0.13	0.05	0.25	-0.28	-0.09	-0.29	0.07	0.11	-0.19	-0.14	0.01	0.27	0.16	-0.04
-0.27	-0.06	0.24	-0.42	0.02	0.01	0.01	0.28	0.19	-0.03	0.14	0.01	0.23	-0.19	0.00	0.30	0.26	-0.12	0.03	0.04	0.47	0.13	0.04	-0.20	0.07	0.11
-0.21	-0.03	0.22	-0.25	0.20	0.52	0.11	0.14	-0.14	-0.11	-0.06	-0.26	-0.10	-0.18	-0.15	-0.30	-0.25	-0.24	-0.06	0.04	-0.10	-0.21	-0.07	0.06	-0.18	-0.15
-0.11	0.33	0.18	0.17	0.06	-0.34	0.18	0.13	-0.28	-0.22	0.39	0.04	-0.04	-0.22	-0.31	-0.15	0.23	-0.08	0.23	-0.12	-0.21	-0.06	0.04	-0.12	0.13	-0.06
-0.10	0.34	0.20	0.12	0.13	-0.31	0.22	0.09	0.00	0.04	-0.56	-0.04	-0.20	-0.32	0.17	0.13	-0.16	0.14	-0.04	-0.08	0.19	0.01	-0.17	0.08	-0.15	0.01
-0.16	0.34	0.19	0.13	-0.15	0.43	-0.22	-0.10	0.13	0.00	-0.19	0.31	-0.10	-0.01	-0.12	0.29	0.10	0.08	0.29	0.18	-0.08	-0.17	0.19	0.14	0.23	-0.10
-0.11	0.31	0.15	0.15	-0.10	0.19	-0.14	-0.22	0.11	0.02	-0.12	-0.24	0.09	0.00	0.30	-0.28	0.28	-0.18	-0.24	-0.09	-0.16	0.28	-0.12	-0.33	0.18	0.19
-0.11	0.32	0.17	0.16	0.03	-0.09	-0.10	0.06	0.12	0.00	0.22	-0.13	0.28	0.42	0.21	0.18	-0.13	-0.07	-0.07	-0.08	0.04	-0.30	0.09	-0.10	-0.45	-0.24
-0.11	0.26	0.18	0.10	0.03	0.04	-0.11	0.06	-0.05	0.15	0.30	-0.01	0.05	0.16	-0.22	-0.15	-0.36	0.18	-0.17	0.14	0.26	0.34	-0.09	0.38	0.13	0.29
-0.26	0.09	-0.33	-0.14	0.04	-0.15	-0.45	-0.06	-0.09	-0.30	-0.22	-0.26	0.05	0.01	-0.33	0.12	-0.19	0.12	0.10	0.06	-0.10	0.04	0.05	-0.26	-0.12	0.24
-0.20	0.12	-0.32	-0.09	-0.01	0.07	-0.11	-0.27	-0.19	-0.18	0.04	0.04	0.28	-0.14	0.18	-0.16	0.10	0.11	0.16	-0.22	0.28	0.11	-0.23	0.30	0.02	-0.44
-0.19	0.14	-0.30	-0.03	0.20	-0.02	0.18	0.37	0.02	0.36	-0.32	0.09	0.27	0.30	-0.17	-0.32	0.26	0.07	0.03	0.13	-0.04	-0.07	0.08	-0.01	0.07	-0.03
-0.20	0.14	-0.29	-0.10	0.12	-0.06	0.25	-0.11	0.42	0.01	0.20	-0.41	-0.27	-0.01	-0.06	0.26	0.12	0.01	-0.22	0.03	-0.18	0.00	0.06	0.28	0.19	-0.14
-0.18	0.14	-0.29	-0.05	0.08	0.10	0.30	-0.14	-0.07	0.09	0.18	0.14	0.13	-0.12	0.38	0.06	-0.27	-0.15	0.28	-0.01	-0.07	-0.19	0.08	-0.01	0.02	0.51
-0.18	0.13	-0.29	0.01	0.16	0.09	-0.22	0.12	-0.18	0.14	0.19	0.42	-0.49	-0.04	0.08	0.05	0.05	-0.10	-0.37	-0.03	0.11	0.03	-0.05	-0.23	-0.15	-0.08
-0.28	-0.22	-0.02	0.25	-0.16	0.08	0.01	0.30	0.03	-0.20	-0.11	0.18	0.19	0.01	-0.08	0.18	-0.03	-0.11	-0.26	-0.56	-0.21	0.01	-0.04	0.25	0.06	0.16
-0.23	-0.14	-0.03	0.22	-0.15	-0.04	0.13	0.12	-0.16	-0.25	0.00	0.03	0.18	-0.20	0.24	0.00	-0.05	0.11	-0.25	0.57	-0.13	0.20	0.34	0.02	-0.06	-0.16
-0.24	-0.10	-0.04	0.15	-0.21	-0.05	0.21	-0.14	0.09	0.17	-0.08	-0.01	-0.13	0.13	-0.16	-0.04	-0.38	-0.32	0.21	-0.15	0.19	0.31	0.24	-0.28	0.13	-0.29
-0.23	-0.14	-0.01	0.12	-0.15	0.05	0.07	0.20	0.32	0.01	0.16	0.08	-0.06	0.00	0.04	-0.04	-0.11	0.29	0.25	0.16	-0.20	0.05	-0.64	-0.23	-0.08	-0.08
-0.22	-0.16	0.01	0.18	-0.31	-0.05	-0.17	-0.11	-0.13	0.55	0.10	-0.25	0.08	-0.37	-0.11	-0.02	0.05	0.16	-0.12	-0.04	0.10	-0.39	0.04	-0.05	0.01	0.02
-0.23	-0.17	-0.02	0.25	-0.18	0.01	0.10	-0.05	-0.12	-0.23	-0.05	-0.18	-0.35	0.32	0.02	-0.10	0.35	-0.13	0.16	0.14	0.37	-0.14	-0.07	0.18	-0.17	0.26
-0.13	-0.19	0.11	0.16	0.39	-0.07	-0.01	-0.31	-0.23	0.22	-0.06	0.04	0.20	0.05	-0.15	0.36	0.10	-0.36	0.01	0.23	-0.16	0.14	-0.33	0.08	-0.05	-0.03
-0.16	-0.22	0.14	0.17	0.46	-0.02	0.00	-0.13	-0.01	-0.21	0.00	0.01	0.00	0.16	0.11	-0.05	-0.13	0.30	-0.12	-0.05	0.18	-0.31	0.02	-0.24	0.50	-0.05
-0.17	-0.21	0.15	0.13	0.39	0.06	-0.13	-0.08	0.20	0.13	0.08	0.00	-0.10	-0.12	0.07	-0.14	0.16	0.27	0.27	-0.22	-0.11	0.29	0.35	0.09	-0.37	0.07

S:

Eigen Vectors are :	**9431.25**	**3392.97**	**2632.14**	**1739.27**	**998.231**	**332.355**	**231.007**	**187**	**118**	**109**	**100**	**78.9**	**67.4**	**54.4**	**41.2**	**33.3**	**17.6**	**11**	**6.71**	**5.58**	**3.54**	**1.46**

97.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	58.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	51.30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	41.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	31.59	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	18.23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	15.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	13.69	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.86	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.88	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.37	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.42	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.77	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.20	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.31	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.59	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.36	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.88	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.21
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

V:

-0.2462	-0.0947	0.31334	-0.3286	0.47683	0.13077	-0.0409	0.06372	-0.207	0.03217	-0.17	-0.4	0.06	0.1	-0.3	0.08	-0.4	0.05	0.03	-0.2	0.1	-0
-0.1888	0.01036	-0.0201	-0.338	0.00331	-0.1692	0.25306	-0.2656	0.65382	-0.0953	0.15	0.01	0.21	0.08	-0.2	0.08	0.16	0.09	0.05	-0.1	0.17	0.18
-0.1046	0.03685	0.12443	-0.0879	0.00933	-0.0045	0.02184	0.26904	-0.0844	-0.0728	0.44	0.24	0.14	0.18	0.03	0.01	-0.2	0.33	-0.4	0.51	0.11	0.14
-0.0954	-0.1002	0.09099	0.06021	0.23653	-0.1929	-0.1856	-0.4946	0.05189	0.5261	0.15	0.28	0.06	0.01	-0.1	-0	-0.2	-0.2	0.02	0.21	-0.3	-0.2
-0.1605	-0.0881	0.25892	-0.4288	0.0508	-0.0299	0.03562	0.04518	-0.2291	0.09073	0.06	0.06	-0.1	0.26	0.4	0.04	0.53	-0.2	0.09	0.01	-0.2	0.19
-0.0435	-0.016	0.07179	-0.1411	0.0015	0.00978	-0.0111	0.09787	0.05282	-0.0318	0.03	0.01	0.03	-0.1	0.04	-0.1	0.1	-0.6	-0.1	0.23	0.59	-0.4
-0.3286	-0.3818	0.15647	0.32695	0.2919	0.01916	-0.0573	-0.0008	0.28289	-0.3228	0.07	0.03	-0.4	-0	0.25	-0.3	0.04	0.19	0.06	-0	0.01	-0.2
-0.0917	0.40902	0.25685	0.25155	-0.028	0.11397	-0.2671	-0.0873	0.24091	0.04113	-0.2	-0.1	0.19	0.31	0.41	0.39	-0	0.11	-0	-0	0.12	-0.1
-0.1366	0.33775	0.01841	0.18061	0.14181	-0.3512	0.53066	0.32859	0.13891	0.37235	-0.1	-0.2	-0.2	-0	0.12	-0.1	-0.1	-0.1	-0	0.04	-0	0.07
-0.085	0.35705	0.20654	0.22879	0.02378	-0.1407	-0.113	0.11496	-0.1952	-0.1136	0.53	0.1	-0.1	0.13	-0.3	-0	0.14	-0.1	0.29	-0.3	0.04	-0.1
-0.1343	0.1403	-0.3722	-0.1026	0.20827	-0.0306	-0.0881	0.06924	-0.1199	0.18287	-0.21	0.02	0.09	0.15	-0.2	-0.2	0.43	0.45	-0.1	0.05	0.08	-0.4
-0.1223	0.12016	-0.3394	-0.0921	0.16483	0.01345	-0.0938	-0.1419	-0.0537	0.0286	0.23	0.02	-0.4	-0.1	0.14	0.3	-0	-0	-0.6	-0.4	0.07	0.02
-0.1847	0.26738	-0.122	-0.0665	0.12611	-0.4369	-0.45	0.03253	0.03333	-0.3344	-0.13	-0.2	-0	-0.2	-0.1	0.02	0.07	-0.1	0.02	0.34	-0.2	0.21
-0.1603	-0.1013	0.14565	-0.1418	-0.1383	0.03495	-0.158	0.38501	0.15544	0.288	0.17	-0	0.2	-0.6	0.12	0.18	0.06	0.19	0.09	-0.1	-0.1	-0.2
-0.3283	0.07854	-0.0673	0.08291	-0.1981	0.46977	-0.3156	0.13299	0.19319	0.32549	-0.05	0	-0.2	0.09	-0.2	-0.3	0.05	-0.2	-0	0.04	0.08	0.39
-0.3041	-0.2025	0.0675	-0.0807	-0.5614	-0.4939	-0.1547	-0.0407	-0.1673	0.05361	-0.18	0.05	-0.1	0.17	0.04	-0.1	-0.2	0.11	-0.1	-0.2	0.18	-0.1
-0.1398	0.11996	-0.3662	-0.081	0.14723	0.03371	-0.0041	-0.1217	-0.1938	0.04502	0.11	0.17	0.11	-0.1	0.39	-0.1	-0.2	0.08	0.52	-0	0.37	0.24
-0.346	-0.131	-0.2145	0.19893	0.144	0.02362	0.10344	0.23203	-0.0524	-0.1746	-0.14	0.31	0.54	0.11	0.02	0.02	-0	-0.3	-0.2	-0.2	-0.2	0.04
-0.3405	0.00011	-0.2229	-0.1073	-0.2073	0.16236	0.20416	0.05948	0.0061	-0.1194	-0.05	0.12	-0.3	0.11	-0.2	0.47	-0.1	-0	0.31	0.26	-0.2	-0.3
-0.2854	0.10127	-0.098	0.03493	-0.2392	0.19121	0.1817	-0.3065	-0.1342	-0.0621	0.31	-0.6	0.23	-0	0.16	-0.3	-0	-0	-0.1	0.08	-0.2	-0.2
-0.2036	0.37668	0.3598	-0.0701	-0.0662	0.17186	0.22507	-0.2885	-0.174	-0.1808	-0.28	0.38	-0	-0.4	-0.1	-0.2	0.04	0.11	-0.1	-0	-0	0.02
-0.2064	-0.2649	0.06531	0.43225	0.0188	-0.0873	0.15943	-0.1581	-0.2696	0.14119	-0.02	-0.2	0.07	-0.2	-0.2	0.39	0.34	0.08	-0	0.16	0.3	0.25

5b. Selecting Eigen values to preserve 85% of information.

check = 0.85*sum(diag(S.^2))

Taking the sum of the eigen values in order from the top we get the value <= check.

Hence we get first 4. They account to a little more than 85%. Eigen values are : **9431.25, 3392.97,2632.14,1739.27**

9431.25	**3392.97**	**2632.14**	**1739.27**	**998.231**	**332.355**	**231.007**	**187.285**	**117.985**	**109**	**100**	**78.9**	**67.4**	**54.4**	**41.2**	**33.3**	**17.6**	**11**	**6.71**	**5.58**	**3.54**	**1.46**

Total :	19593	
85%	16654	
Sum of first 4 :	17195.6	87.7642

And hence choosing the corresponding Eigen vectors from U and V :

U_new:

-0.19	0.02	0.14	-0.26
-0.19	-0.02	0.20	-0.37
-0.24	-0.11	0.18	-0.27
-0.27	-0.06	0.24	-0.42
-0.21	-0.03	0.22	-0.25
-0.11	0.33	0.18	0.17
-0.10	0.34	0.20	0.12
-0.16	0.34	0.19	0.13
-0.11	0.31	0.15	0.15
-0.11	0.32	0.17	0.16
-0.11	0.26	0.18	0.10
-0.26	0.09	-0.33	-0.14
-0.20	0.12	-0.32	-0.09
-0.19	0.14	-0.30	-0.03
-0.20	0.14	-0.29	-0.10
-0.18	0.14	-0.29	-0.05
-0.18	0.13	-0.29	0.01
-0.28	-0.22	-0.02	0.25
-0.23	-0.14	-0.03	0.22
-0.24	-0.10	-0.04	0.15
-0.23	-0.14	-0.01	0.12
-0.22	-0.16	0.01	0.18
-0.23	-0.17	-0.02	0.25
-0.13	-0.19	0.11	0.16
-0.16	-0.22	0.14	0.17
-0.17	-0.21	0.15	0.13

S_new:

97.11	0.00	0.00	0.00
0.00	58.25	0.00	0.00
0.00	0.00	51.30	0.00
0.00	0.00	0.00	41.70

V_new:

-0.25	-0.19	-0.10	-0.10	-0.16	-0.04	-0.33	-0.09	-0.14	-0.09	-0.13	-0.12	-0.18	-0.16	-0.33	-0.30	-0.14	-0.35	-0.34	-0.29	-0.20	-0.21
-0.09	0.01	0.04	-0.10	-0.09	-0.02	-0.38	0.41	0.34	0.36	0.14	0.12	0.27	-0.10	0.08	-0.20	0.12	-0.13	0.00	0.10	0.38	-0.26
0.31	-0.02	0.12	0.09	0.26	0.07	0.16	0.26	0.02	0.21	-0.37	-0.34	-0.12	0.15	-0.07	0.07	-0.37	-0.21	-0.22	-0.10	0.36	0.07
-0.33	-0.34	-0.09	0.06	-0.43	-0.14	0.33	0.25	0.18	0.23	-0.10	-0.09	-0.07	-0.14	0.08	-0.08	-0.08	0.20	-0.11	0.03	-0.07	0.43

5c. Interpreting Dropped Eigen Vector :

The vectors in U matrix represent the document - to - concept similarity and the vectors in V matrix denote the term - to - concept similarity.

The vectors are dropped as their singular values or the eigen values which represent the concept strength in the entire data is low. Now, each time a vector drops, we lose the absolute representation of the final dataset.

Let's consider the dropped vector of the singular value : 31.59 as the strength of this concept is not so insignificant, we m ay lose some relevant representation. However, if we

U :

-0.15
-0.17
-0.06
0.02
0.20
0.06
0.13
-0.15
-0.10
0.03
0.03
0.04
-0.01
0.20
0.12
0.08
0.16
-0.16
-0.15
-0.21
-0.15
-0.31
-0.18
0.39
0.46
0.39

V:

-0.1605	-0.0881	0.25892	-0.4288	0.0508	-0.0299	0.03562	0.04518	-0.23	0.09	0.06	0.06	-0.1	0.26	0.4	0.04	0.53	-0.2	0.09	0.01	-0.2	0.19
data	program	complexi	concept	algorithn	processo	game	gene	drug	diseas	electi	vote	candi	optim	result	trainii	camp	win	run	best	expre	sport

S: 31.59

These represent the terms that have the greatest similarity to the concept

All these documents have a greater similarity to the concept represented by this vector

As this latent vector(hidden feature) is a combination of the some of the columns in the data, we may lose the information any documents would be projecting in combination of these.

Also if there is a concept being represented by the terms , then those would become irrelevant.

5d. Interpreting the chosen eigen values and Concept Vectors: (Below as shown in the tables)

The concept vectors in U represent the document to concept similarity , so each column is an axis and the documents are proje cted to the concept to fit in. When we choose the top ones, only those documents having representations of the highest eigen valued concept are fit into the model and the others d eviate or may have higher variance.

The eigen value in the S represents the strength of each concept. Choosing the top 4 gave those concepts that have the highes t strength or dominate the dataset representation.

The concept vectors in V represent the term to concept similarity, so each row is an axis and the terms are projected to the concept to fit it. When we choose the top ones, those terms that pertain to the top concepts sustain and the rest become irrelevant.

U_new:

Concept1	Concept2	Concept3	Concept4	
-0.19	0.02	0.14	-0.26	doc1
-0.19	-0.02	0.20	-0.37	doc2
-0.24	-0.11	0.18	-0.27	doc3
-0.27	-0.06	0.24	-0.42	doc4
-0.21	-0.03	0.22	-0.25	doc5
-0.11	0.33	0.18	0.17	doc6
-0.10	0.34	0.20	0.12	doc7
-0.16	0.34	0.19	0.13	doc8
-0.11	0.31	0.15	0.15	doc9
-0.11	0.32	0.17	0.16	doc10
-0.11	0.26	0.18	0.10	doc11
-0.26	0.09	-0.33	-0.14	doc12
-0.20	0.12	-0.32	-0.09	doc13
-0.19	0.14	-0.30	-0.03	doc14
-0.20	0.14	-0.29	-0.10	doc15
-0.18	0.14	-0.29	-0.05	doc16
-0.18	0.13	-0.29	0.01	doc17
-0.28	-0.22	-0.02	0.25	doc18
-0.23	-0.14	-0.03	0.22	doc19
-0.24	-0.10	-0.04	0.15	doc20
-0.23	-0.14	-0.01	0.12	doc21
-0.22	-0.16	0.01	0.18	doc22
-0.23	-0.17	-0.02	0.25	doc23
-0.13	-0.19	0.11	0.16	doc24
-0.16	-0.22	0.14	0.17	doc25

S_new:

97.11	0.00	0.00	0.00
0.00	58.25	0.00	0.00
0.00	0.00	51.30	0.00
0.00	0.00	0.00	41.70

V_new:

Concept1	-0.25	-0.19	-0.10	-0.10	-0.16	-0.04	-0.33	-0.09	-0.14	-0.09	-0.13	-0.12	-0.18	-0.16	-0.33	-0.30	-0.14	-0.35	-0.34	-0.29	-0.20	-0.21
Concept2	-0.09	0.01	0.04	-0.10	-0.09	-0.02	-0.38	0.41	0.34	0.36	0.14	0.12	0.27	-0.10	0.08	-0.20	0.12	-0.13	0.00	0.10	0.38	-0.26
Concept3	0.31	-0.02	0.12	0.09	0.26	0.07	0.16	0.26	0.02	0.21	-0.37	-0.34	-0.12	0.15	-0.07	0.07	-0.37	-0.21	-0.22	-0.10	0.36	0.07
Concept4	-0.33	-0.34	-0.09	0.06	-0.43	-0.14	0.33	0.25	0.18	0.23	-0.10	-0.09	-0.07	-0.14	0.08	-0.08	-0.08	0.20	-0.11	0.03	-0.07	0.43

-0.17	-0.21	0.15	0.13	doc26

5e. MSE between Original and the Predicted matrix using only 4 Eigen values.

A_new = U_new * S_new * V_new'(transposed)

Error = immse(A, A_new) = 4.191

New_SVD Matrix:

10.3001	7.02245	3.81375	1.64838	9.45023	2.85367	3.18714	1.14722	0.94405	0.86201	0.99	0.88	3.45	5.45	4.65	6.74	0.88	2.44	5.76	4.2	7.45	-0.8
13.0763	8.5822	4.56159	1.93787	12.4423	3.7632	3.26907	-0.0876	-0.4832	-0.3059	0.13	0.1	2.91	6.85	4.13	7.96	-0	1.44	5.82	3.72	8.1	-1.8
12.9585	8.03903	4.36693	3.03261	11.5176	3.36658	7.91411	-0.8705	-0.7151	-0.9048	0.12	0.13	2.38	7.35	5.77	10	0.18	4.86	7.28	4.88	6.53	2.28
16.4052	10.6482	5.66803	2.90936	15.2434	4.56125	6.14911	-0.4347	-0.621	-0.6251	0.33	0.3	3.57	8.85	6.06	11	0.24	3.46	8.15	5.32	9.55	-0.5
12.234	7.18591	4.43542	2.55296	10.8666	3.20845	5.83492	1.5333	0.61692	1.14585	-0.59	-0.5	2.71	6.62	5.07	8.25	-0.6	2.91	5.66	4.29	8.42	0.98
1.27346	-0.4173	2.32417	0.28411	-0.6779	-0.19	-0.3495	13.0471	9.41352	11.3409	-0.05	-0.2	5.51	0.02	4.84	-0.8	-0.2	0.39	0.67	4.23	12.3	0.57
1.94164	0.08329	2.52072	0.14234	0.20687	0.10112	-1.197	12.7901	9.06146	11.0817	-0.2	-0.3	5.47	0.26	4.41	-0.8	-0.4	-0.4	0.46	3.9	12.6	-0.4
3.07603	1.03057	3.02748	0.70052	0.80919	0.24963	0.86971	13.3023	9.92614	11.5718	0.71	0.5	6.57	1.07	6.42	0.89	0.57	1.82	2.52	5.65	13.6	1.01
1.4148	0.02076	2.23425	0.33863	-0.458	-0.1258	0.04001	11.8878	8.82187	10.3526	0.48	0.3	5.47	0.21	5.03	-0.2	0.36	1.06	1.38	4.43	11.3	0.73
1.42859	-0.2064	2.30927	0.39028	-0.5433	-0.1576	0.1251	12.5095	9.13914	10.8844	0.16	0.01	5.47	0.18	5.04	-0.4	0.04	0.88	1.09	4.4	11.8	0.87
2.56597	0.48787	2.40409	0.54864	0.85583	0.25988	0.41191	10.5915	7.493	9.17515	-0.31	-0.4	4.59	0.84	4.33	0.34	-0.4	0.48	1.03	3.75	10.8	0.54
2.36215	7.23629	1.25654	-0.0402	1.75833	0.64098	1.63036	-1.3733	3.89144	-0.8255	11.2	10.1	8.63	1.89	9.41	5.96	11	10.6	13.1	9.26	1.47	0.08
0.3595	5.41136	0.62163	-0.5167	-0.0133	0.11841	0.07071	-0.5495	4.00376	-0.1291	10.1	9.11	7.7	0.64	7.78	3.82	9.85	8.65	10.7	7.78	0.99	-0.5
-0.5432	4.3768	0.4468	-0.5144	-1.1043	-0.2269	0.1852	0.67111	4.70564	0.93745	9.49	8.59	7.55	0.13	7.65	3.11	9.31	8.41	9.91	7.58	1.36	0.1
0.71516	5.52616	0.84321	-0.5731	0.33339	0.23462	-0.4011	0.17882	4.38807	0.48221	9.84	8.9	7.93	0.72	7.75	3.65	9.6	8.13	10.5	7.76	1.92	-0.9
-0.2954	4.53529	0.50161	-0.5948	-0.731	-0.1032	-0.2601	0.48242	4.45277	0.75399	9.34	8.45	7.42	0.2	7.32	3	9.13	7.91	9.68	7.31	1.42	-0.4
-1.1873	3.57588	0.21343	-0.4127	-1.8588	-0.4787	0.70014	0.87443	4.68192	1.11837	8.94	8.09	7.02	-0.2	7.37	2.8	8.8	8.38	9.27	7.23	0.95	0.82
4.05649	1.49639	1.27997	4.34158	0.71463	-0.1651	16.8433	-0.4958	1.18933	-0.1829	1.23	1.23	1.02	3.96	8.74	9.83	1.85	13.3	8.32	6.85	-0.5	13.3
2.80085	1.0947	1.07247	3.37235	-0.0379	-0.3068	13.2556	0.78523	2.07491	0.91071	1.56	1.49	1.65	2.9	7.69	7.64	2.03	11.1	7.06	6.14	0.44	10.7
3.55818	2.28014	1.38534	2.98105	1.03754	0.07621	11.5733	0.64775	2.21043	0.77669	2.47	2.3	2.54	3.13	7.8	7.62	2.84	10.5	7.73	6.44	1.23	8.89
4.26202	2.3372	1.45875	3.14449	1.85621	0.31012	11.8468	-0.1769	1.18246	0.01382	1.54	1.47	1.64	3.5	7.03	7.84	1.93	9.81	7.06	5.69	0.82	8.83
3.74663	1.32934	1.24667	3.45272	1.10005	0.03582	13.0742	0.06551	1.05238	0.22397	0.58	0.61	0.81	3.33	6.78	7.75	1.05	9.95	6.28	5.28	0.35	10.1
2.50353	0.46089	0.85903	3.62221	-0.4435	-0.4653	14.275	0.36658	1.55242	0.54537	0.86	0.87	0.82	2.85	7.36	7.73	1.4	11.2	6.55	5.72	-0.4	11.7
3.86025	-0.0352	1.08756	3.28284	1.67868	0.20931	11.6107	-0.1335	-0.6128	-0.0914	-2.62	-2.3	-1.7	3.1	3.6	6.06	-2.1	6.07	2.45	2.29	0.12	8.91
5.08418	0.3605	1.44968	3.89943	2.54614	0.42335	13.6339	-0.3126	-0.8535	-0.2547	-3.06	-2.7	-1.9	3.92	4.3	7.41	-2.5	7.1	3.09	2.77	0.37	10.2
5.90238	1.08309	1.76052	3.84409	3.44469	0.71493	13.152	-0.2694	-0.8021	-0.2357	-2.83	-2.5	-1.5	4.28	4.46	7.68	-2.3	6.87	3.45	2.97	1.07	9.53

5f. Four Interpretations as shown by the Faloutsos-p1-SVD slide set.

Interpretation - One:

Main Idea : Matrix U represents the document-to-concept similarity, Matrix V represents the term-to-concept similarity and S represents the Concept strength.

U is calculated as A*A'(transpose)- (26*22) (22*26) = (26*26) ->document- by document matrix. This is further fitted into to get U matrix

624	603	616	731	514	196	200	322	221	188	203	382	306	203	305	264	163	360	313	395	345	330	299	164	198	227
603	783	712	880	646	152	160	288	176	159	199	372	257	181	240	208	144	375	311	368	350	358	299	165	204	247
616	712	927	950	622	150	160	234	144	168	186	533	316	250	309	223	235	587	470	504	510	512	451	336	421	468
731	880	950	1193	878	193	208	343	196	215	253	540	351	315	386	299	248	572	447	480	521	449	395	327	429	490
514	646	622	878	824	182	212	364	232	206	260	368	256	258	284	244	218	444	344	354	386	306	324	338	446	478
196	152	150	193	182	710	642	590	528	604	520	167	149	203	204	189	192	74	142	132	84	66	93	42	49	40
200	160	160	208	212	642	711	592	530	579	487	152	116	224	201	172	164	20	85	101	41	12	35	52	58	53
322	288	234	343	364	590	592	859	676	633	566	297	277	245	255	264	279	244	234	270	228	223	213	18	25	76
221	176	144	196	232	528	530	676	593	558	478	211	212	186	208	204	191	128	158	184	132	152	142	0	0	32
188	159	168	215	206	604	579	633	558	617	515	202	173	216	215	195	199	115	146	155	121	102	111	46	58	66
203	199	186	253	260	520	487	566	478	515	465	167	141	175	168	158	178	108	127	144	119	114	98	60	72	93
382	372	533	540	368	167	152	297	211	202	167	1088	853	767	810	723	744	566	492	530	468	430	450	154	194	213
306	257	316	351	256	149	116	277	212	173	141	853	750	646	684	667	633	417	388	421	353	324	342	72	86	101
203	181	250	315	258	203	224	245	186	216	175	767	646	763	682	643	638	388	341	375	327	236	286	126	150	158
305	240	309	386	284	204	201	255	208	215	168	810	684	682	776	672	624	364	340	409	351	259	309	92	126	144
264	208	223	299	244	189	172	264	204	195	158	723	667	643	672	664	601	355	343	384	318	250	289	82	95	102
163	144	235	248	218	192	164	279	191	199	178	744	633	638	624	601	671	375	332	350	314	253	298	120	147	164
360	375	587	572	444	74	20	244	128	115	108	566	417	388	364	355	375	1050	842	787	780	790	848	472	578	580
313	311	470	447	344	142	85	234	158	146	127	492	388	341	340	343	332	842	714	657	632	652	704	372	452	442
395	368	504	480	354	132	101	270	184	155	144	530	421	375	409	384	350	787	657	687	621	655	677	328	379	393
345	350	510	521	386	84	41	228	132	121	119	468	353	327	351	318	314	780	632	621	622	611	636	323	411	436
330	358	512	449	306	66	12	223	152	102	114	430	324	236	259	250	253	790	652	655	611	746	670	338	361	406
299	299	451	395	324	93	35	213	142	111	98	450	342	286	309	289	298	848	704	677	636	670	754	387	470	459
164	165	336	327	338	42	52	18	0	46	60	154	72	126	92	82	120	472	372	328	323	338	387	561	619	582
198	204	421	429	446	49	58	25	0	58	72	194	86	150	126	95	147	578	452	379	411	361	470	619	749	699
227	247	468	490	478	40	53	76	32	66	93	213	101	158	144	102	164	580	442	393	436	406	459	582	699	692

V is calculated as A'(transpose)*A- (22*26) (26*22) = (22*22) ->term-to- term matrix. This is further fitted into to get V matrix

1307	584	381	386	879	244	953	142	174	126	124	91	331	539	563	593	96	623	575	431	654	126
584	634	224	153	502	154	386	15	183	12	308	288	384	338	500	580	308	503	679	506	398	12
381	224	209	87	305	90	280	174	163	193	42	39	177	231	293	298	50	244	280	247	356	193
386	153	87	311	203	38	550	35	51	40	28	36	88	154	205	260	39	329	163	143	117	40
879	502	305	203	794	224	485	0	3	1	0	0	164	464	347	607	1	276	434	296	484	1
244	154	90	38	224	76	108	0	0	0	0	0	58	144	102	162	0	66	122	80	144	0
953	386	280	550	485	108	1885	0	126	23	72	87	189	562	883	1016	120	1298	846	659	233	23
142	15	174	35	0	0	0	985	629	786	12	0	424	42	424	0	0	54	96	339	898	786
174	183	163	51	3	0	126	629	792	598	315	254	524	43	442	97	280	375	371	435	659	598
126	12	193	40	1	0	23	786	598	759	38	36	409	31	333	11	30	83	96	299	758	759
124	308	42	28	0	0	72	12	315	38	686	585	526	12	474	136	634	584	615	435	78	38
91	288	39	36	0	0	87	0	254	36	585	564	466	12	429	128	580	505	578	418	68	36
331	384	177	88	164	58	189	424	524	409	526	466	760	149	585	346	486	536	610	545	539	409
539	338	231	154	464	144	562	42	43	31	12	12	149	462	481	638	28	424	472	362	313	31
563	500	293	205	347	102	883	424	442	333	474	429	585	481	1224	909	483	1064	1129	980	665	333
593	580	298	260	607	162	1016	0	97	11	136	128	346	638	909	1447	181	929	1034	820	408	11
96	308	50	39	1	0	120	0	280	30	634	580	486	28	483	181	642	603	643	484	90	30
623	503	244	329	276	66	1298	54	375	83	584	505	536	424	1064	929	603	1446	1173	904	268	83
575	679	280	163	434	122	846	96	371	96	615	578	610	472	1129	1034	643	1173	1327	1016	490	96
431	506	247	143	296	80	659	339	435	299	435	418	545	362	980	820	484	904	1016	970	612	299
654	398	356	117	484	144	233	898	659	758	78	68	539	313	665	408	90	268	490	612	1300	758
385	104	115	356	121	0	1237	28	123	52	1	9	46	308	582	725	47	899	541	486	78	52

Interpretation - Two:

Main Idea : Best Axis is found to fit the data values and the dimensionality reduction is done by ignoring or removing the concepts with less strength.

-0.19
-0.19
-0.24
-0.27
-0.21
-0.11
-0.10
-0.16
-0.11
-0.11
-0.11
-0.26
-0.20
-0.19

97.11

variance of the concept on V1 axis

-0.20
-0.18
-0.18
-0.28
-0.23
-0.24
-0.23
-0.22
-0.23
-0.13
-0.16
-0.17

-0.25	-0.19	-0.10	-0.10	-0.16	-0.04	-0.33	-0.09	-0.14	-0.09	-0.13	-0.12	-0.18	-0.16	-0.33	-0.30	-0.14	-0.35	-0.34	-0.29	-0.20	-0.21

Interpretation - Three:

Main Idea : Finding Non Zero Blobs in the data - Sample of them are shown below:

data	program	complexi	concept	algorithn	processo	game	gene	drug	disease	electic	vote	candi	optim	result	trainii	camp	win	run	best	expre	sport
12	5	5	0	15	2	0	0	0	0	0	0	0	6	4	12	0	0	8	6	8	0
10	8	4	6	12	4	9	0	0	0	0	0	8	8	5	16	0	4	5	2	4	0
16	12	7	1	15	6	6	0	0	0	0	0	4	12	6	9	0	5	8	4	8	0
18	4	4	0	10	4	8	0	0	0	0	0	0	6	8	0	0	4	6	6	10	0
1	0	4	0	0	0	0	8	12	16	0	0	8	0	0	0	0	1	0	4	12	2
4	0	0	0	0	0	0	12	15	8	1	0	8	0	0	0	0	0	0	0	14	1
1	0	2	0	0	0	0	16	4	10	0	0	4	2	12	0	0	1	6	5	16	0
0	0	0	0	0	0	0	14	6	8	0	0	5	0	8	0	0	0	0	8	12	0
2	1	4	1	0	0	0	15	9	12	0	0	6	0	4	0	0	2	0	5	8	0
4	0	5	2	0	0	0	10	7	11	0	0	4	1	6	0	0	0	0	4	9	0
5	6	0	0	0	0	2	0	0	0	12	10	15	1	9	8	10	10	12	8	0	0
0	5	0	0	0	0	0	0	0	0	10	9	8	0	7	4	12	9	9	10	3	0
3	4	1	0	0	0	0	0	10	0	14	6	6	0	8	0	9	12	8	4	0	0
2	10	0	0	0	0	2	0	8	0	9	11	8	0	6	0	8	6	11	9	0	0
0	5	1	1	0	0	0	0	6	0	8	9	4	0	5	0	12	9	10	9	3	0
0	0	2	2	0	0	3	0	4	3	10	12	7	0	9	0	10	8	9	3	1	0
2	1	2	0	0	0	15	0	0	0	0	0	0	6	10	12	0	16	10	6	0	12
0	0	3	0	0	0	12	0	2	0	0	0	2	4	6	10	2	12	8	8	2	11
1	4	0	3	0	0	9	0	4	0	0	1	0	2	9	12	2	9	10	8	2	9
1	4	3	1	1	0	12	0	2	0	0	0	0	6	9	9	1	9	9	6	0	7
0	0	0	5	0	0	8	0	0	0	0	0	0	8	10	14	0	6	6	9	0	12
0	0	0	0	1	0	14	0	1	1	0	0	0	0	8	11	0	9	9	8	0	12
10	0	0	10	3	0	12	0	0	0	0	0	0	0	0	0	0	8	0	0	0	12
12	1	2	7	3	0	19	0	0	0	0	0	0	0	0	0	0	9	0	0	0	10
11	2	1	8	4	0	18	0	0	0	0	0	0	4	4	0	0	7	0	0	0	9
11	2	1	8	4	0	18	0	0	0	0	0	0	4	4	0	0	7	0	0	0	9

Interpretation - Four:

Main Idea : This talks about 2 properties : 1. Fixed Point Operation And 2. Convergence. For u1 and v1 : shown below:

-0.25
-0.19
-0.10
-0.10
-0.16
-0.04
-0.33
-0.09
-0.14
-0.09
-0.13
-0.12
-0.18
-0.16
-0.33
-0.30
-0.14
-0.35
-0.34
-0.29
-0.20
-0.21

× **A** = **97.1** ×

-0.19
-0.19
-0.24
-0.27
-0.21
-0.11
-0.10
-0.16
-0.11
-0.11
-0.11
-0.26
-0.20
-0.19
-0.20
-0.18
-0.18
-0.28
-0.23
-0.24
-0.23
-0.22
-0.23
-0.13
-0.16
-0.17

5g. SVD analysis between Original and the Predicted matrix using only 2 Eigen values.

As per the below analysis, the document 8 is more realted to both the concepts as per 'U' and hence it has the lowest mean_sq uared error.

U_new:

-0.19	0.02
-0.19	-0.02
-0.24	-0.11
-0.27	-0.06
-0.21	-0.03
-0.11	0.33
-0.10	0.34
-0.16	0.34
-0.11	0.31
-0.11	0.32
-0.11	0.26
-0.26	0.09
-0.20	0.12
-0.19	0.14
-0.20	0.14
-0.18	0.14
-0.18	0.13
-0.28	-0.22
-0.23	-0.14
-0.24	-0.10
-0.23	-0.14
-0.22	-0.16
-0.23	-0.17
-0.13	-0.19
-0.16	-0.22
-0.17	-0.21

S_new:

97.11	0.00
0.00	58.25

V_new:

-0.25	-0.19	-0.10	-0.10	-0.16	-0.04	-0.33	-0.09	-0.14	-0.09	-0.13	-0.12	-0.18	-0.16	-0.33	-0.30	-0.14	-0.35	-0.34	-0.29	-0.20	-0.21
-0.09	0.01	0.04	-0.10	-0.09	-0.02	-0.38	0.41	0.34	0.36	0.14	0.12	0.27	-0.10	0.08	-0.20	0.12	-0.13	0.00	0.10	0.38	-0.26
data	program	complexi	concept	algorit	proce	game	gene	drug	diseas	electi	vote	candi	optim	result	trainii	camp	win	run	best	expre	sport

	data	program	complexi	concept	algorit	proce	game	gene	drug	diseas	electi	vote	candi	optim	result	trainii	camp	win	run	best	expre	sport	
Original	6	12	3	4	8	2	2	0	0	0	0	0	1	2	3	10	0	2	7	6	12	0	
U*S*V	4.40194	3.45065	1.93952	1.6478	2.85	0.78	5.64	2.04	2.8	1.87	2.58	2.34	3.61	2.83	6.06	5.36	2.66	6.19	6.21	5.3	4.05	3.52	
Error	2.55379	73.0915	1.12462	5.53282	26.6	1.49	13.3	4.17	7.82	3.51	6.64	5.47	6.81	0.69	9.33	21.5	7.06	17.5	0.63	0.5	63.1	12.4	13.2
Original	1	0	2	0	0	0	0	16	4	10	0	0	4	2	12	0	0	1	6	5	16	0	
U*S*V	1.92753	3.10949	2.33382	-0.5021	0.74	0.35	-2.4	9.45	8.74	8.33	4.83	4.24	8.1	0.48	6.6	0.7	4.51	2.75	5.24	6.38	10.5	-2	
Error	0.8603	9.6689	0.11144	0.25212	0.55	0.13	5.98	42.9	22.5	2.8	23.3	18	16.8	2.33	29.2	0.49	20.3	3.07	0.57	1.92	29.8	4.12	10.7
Original	0	5	1	1	0	0	0	0	6	0	8	9	4	0	5	0	12	9	10	9	3	0	
U*S*V	3.64949	3.46105	2.16544	0.90664	2.17	0.65	2.83	4.91	5.14	4.37	3.52	3.15	5.44	2.06	6.5	3.83	3.46	5.15	6.09	5.92	6.65	1.58	
Error	13.3188	2.36836	1.35825	0.00872	4.71	0.42	8.02	24.1	0.74	19.1	20	34.2	2.08	4.24	2.25	14.6	72.9	14.9	15.3	9.51	13.3	2.49	12.7

5h. MSE between Original and the Predicted matrix using only 2 Eigen values.

```
S_new([3:26],:) =[]
S_new(:,[3:22]) = []
U_new(:,[3:26]) = []
V_new(:,[3:22])=[]
svd_new =U_new*S_new*V_new'
A = U*S*V'
E=immse(svd_new,sv)
A_new = U_new * S_new * V_new'(transposed)
```

Error = immse(A, A_new) = 11.833

Comments: The error is expected to increase as we are dropping some significant eigen values that potentially represent the data and the concepts. Hence due to the loss of information and underfitting the error increases.