PCA with NaN

```
Dorg = [5   1   1   3;
     3   2   4   1;
     NaN 4   5   5;
     1   1   1   1;
     3   4   NaN NaN;
     5   1   1   1;
     4   2   NaN NaN;
     3   3   3   2;
     NaN NaN NaN NaN;
     1   1   3   4;
     4   2   NaN NaN;
     NaN 4   3   3];
```

Get number of users (raters) and number of artists

```
nu=size(Dorg,1);
na=size(Dorg,2);

Dpca = Dorg;
```

Run PCA

```
[coeff,score,latent, tsq, expl, mu]= pca(Dpca,'algorithm','als','centered','on','NumComponents'
coeff
```

```
coeff = 4×4
   -0.0068    0.8110   -0.4991   -0.3053
    0.2836    0.4202    0.1609    0.8468
    0.4380    0.2924    0.7327   -0.4310
    0.8530   -0.2833   -0.4338   -0.0627
```

```
score
```

```
score = 12×4
   -0.9237   -0.1798   -2.8398   -0.5706
   -1.0184    0.0621    1.3849   -0.2808
    3.3872    1.4568   -0.1542    0.2060
   -2.6024   -2.8571    0.0239    0.7758
    7.8433   -0.5882    0.2481   -0.3481
   -2.6297    0.3869   -1.9723   -0.4452
   -2.3937    1.1851    1.2707   -0.3557
   -0.3198   -0.0934    0.3794    0.9343
   -0.5813    0.4701    1.3563   -0.2323
    0.8327   -3.1224    0.1881   -0.2741
      ⋮
      ⋮
```

```
latent
```

```
latent = 4×1
    9.2548
    2.5448
    1.8675
    0.3199
```

Generate lower dimensional versions of the scores

```
score1=horzcat(score(:,1:1),zeros(nu,3));
score2=horzcat(score(:,1:2),zeros(nu,2));
score3=horzcat(score(:,1:3),zeros(nu,1));
score4=score;
```

Generate predictions

Predictions are taken by multiplying the user specific score vectors with the artist specifc coeff vector. For example for the one-factor model, we consider only the first dimension of the user specific score1 vector multiplied by the corresponding first dimension of the coeff vector.

Since we multiplying a column vector (12x1) with a row vector (1x4) we generate a full (12x4) predicted ratings matrix.

Rating_matrix = user_score_colvec * artist_coeff_rowvec + average_ratings_matrix

= [-1.7;-.05;3.7;-2.1;1.;-2.8 ...] * [-.16 .56 .61 .52] + [3.2 2.2 2.6 2.5; 3.2 2.2 ...]

For higher order (2 factor - 4 factor) models we include more dimensions of the user scores and artist coeffs.

```
Dpred1= score1*coeff'+repmat(mu,nu,1)
```

```
Dpred1 = 12×4
    3.5544    2.0157    2.8875    1.6815
    3.5551    1.9889    2.8460    1.6008
    3.5250    3.2383    4.7759    5.3588
    3.5659    1.5396    2.1521    0.2495
    3.4946    4.5020    6.7279    9.1599
    3.5661    1.5319    2.1401    0.2262
    3.5645    1.5988    2.2435    0.4275
    3.5503    2.1870    3.1520    2.1966
    3.5521    2.1128    3.0374    1.9735
    3.5424    2.5138    3.6569    3.1797
      ⋮
```

```
Dpred2= score2*coeff'+repmat(mu,nu,1)
```

```
Dpred2 = 12×4
    3.4086    1.9401    2.8349    1.7324
    3.6054    2.0149    2.8642    1.5832
    4.7065    3.8504    5.2018    4.9460
    1.2488    0.3391    1.3168    1.0590
    3.0176    4.2549    6.5559    9.3266
    3.8798    1.6944    2.2532    0.1166
    4.5256    2.0967    2.5900    0.0918
    3.4745    2.1477    3.1247    2.2231
    3.9334    2.3103    3.1749    1.8403
    1.0102    1.2019    2.7440    4.0644
      ⋮
```

```
Dpred3= score3*coeff'+repmat(mu,nu,1)
```

```
Dpred3 = 12×4
    4.8258    1.4832    0.7541    2.9642
    2.9143    2.2378    3.8790    0.9824
    4.7834    3.8255    5.0888    5.0129
    1.2368    0.3430    1.3344    1.0486
    2.8937    4.2948    6.7377    9.2190
```

```
    4.8641    1.3770    0.8081    0.9721
    3.8914    2.3012    3.5211   -0.4594
    3.2852    2.2088    3.4027    2.0585
    3.2565    2.5286    4.1687    1.2520
    0.9163    1.2321    2.8819    3.9828
     ⋮
```

```
Dpred4= score4*coeff'+repmat(mu,nu,1)
```

```
Dpred4 = 12×4
    5.0000    1.0000    1.0000    3.0000
    3.0000    2.0000    4.0000    1.0000
    4.7205    4.0000    5.0000    5.0000
    1.0000    1.0000    1.0000    1.0000
    3.0000    4.0000    6.8877    9.2408
    5.0000    1.0000    1.0000    1.0000
    4.0000    2.0000    3.6744   -0.4371
    3.0000    3.0000    3.0000    2.0000
    3.3274    2.3318    4.2688    1.2666
    1.0000    1.0000    3.0000    4.0000
     ⋮
```

Average Rating Error

```
err1=sqrt(sum(fillmissing((Dpred1-Dorg),"constant",0).^2,"all")/(nu*na))
```

```
err1 = 0.9007
```

```
err2=sqrt(sum(fillmissing((Dpred2-Dorg),"constant",0).^2,"all")/(nu*na))
```

```
err2 = 0.6301
```

```
err3=sqrt(sum(fillmissing((Dpred3-Dorg),"constant",0).^2,"all")/(nu*na))
```

```
err3 = 0.2625
```

```
err4=sqrt(sum(fillmissing((Dpred4-Dorg),"constant",0).^2,"all")/(nu*na))
```

```
err4 = 6.9611e-15
```

```

```

For each Artist build a Regression Tree from the first two Score Dimension and Known (not Nan) Target Ratings

First prepare new xydata table with NaN entry rows removed

```
xydata_art1 = rmmissing(horzcat(score,Dorg(:,1)));
xydata_art2 = rmmissing(horzcat(score,Dorg(:,2)));
xydata_art3 = rmmissing(horzcat(score,Dorg(:,3)));
xydata_art4 = rmmissing(horzcat(score,Dorg(:,4)));
```

Build a tree model for each artist rating with two PCA factors

```
rtree_1= fitrtree(xydata_art1(:,1:2),xydata_art1(:,end),"MinParentSize",4);
rtree_2= fitrtree(xydata_art2(:,1:2),xydata_art2(:,end),"MinParentSize",4);
rtree_3= fitrtree(xydata_art3(:,1:2),xydata_art3(:,end),"MinParentSize",4);
rtree_4= fitrtree(xydata_art4(:,1:2),xydata_art4(:,end),"MinParentSize",4);
```

Predict rating for each artist

```
pred_1 = predict(rtree_1, score(:,1:2));
pred_2 = predict(rtree_2, score(:,1:2));
pred_3 = predict(rtree_3, score(:,1:2));
pred_4 = predict(rtree_4, score(:,1:2));
```

Combine rating into table

```
pred = horzcat(pred_1, pred_2, pred_3, pred_4)
```

```
pred = 12×4    Rows 2:11 | Columns 1:4
    3.6667    2.0000    4.0000    1.0000
    3.0000    4.0000    4.0000    4.0000
    1.0000    1.0000    1.0000    1.0000
    3.0000    4.0000    2.3333    4.0000
    5.0000    1.0000    1.0000    1.0000
    3.6667    2.0000    1.0000    1.0000
    3.0000    3.0000    2.3333    2.5000
    3.0000    3.0000    4.0000    2.5000
    1.0000    1.0000    2.3333    4.0000
    3.6667    2.0000    1.0000    1.0000
```

```
err=sqrt(sum(fillmissing((pred-Dorg),"constant",0).^2,"all")/(nu*na))
```

```
err = 0.4040
```

```
view(rtree_1)
```

```
Decision tree for regression
1  if x2<-1.72265 then node 2 elseif x2>=-1.72265 then node 3 else 3.22222
2  fit = 1
3  if x1<-0.621748 then node 4 elseif x1>=-0.621748 then node 5 else 3.85714
4  if x1<-2.51174 then node 6 elseif x1>=-2.51174 then node 7 else 4.2
5  fit = 3
6  fit = 5
7  if x1<-0.971031 then node 8 elseif x1>=-0.971031 then node 9 else 4
8  fit = 3.66667
9  fit = 5
```

```
view(rtree_2)
```

```
Decision tree for regression
 1  if x1<-0.621748 then node 2 elseif x1>=-0.621748 then node 3 else 2.27273
 2  if x1<-2.49808 then node 4 elseif x1>=-2.49808 then node 5 else 1.5
 3  if x2<-1.85529 then node 6 elseif x2>=-1.85529 then node 7 else 3.2
 4  fit = 1
 5  if x1<-0.971031 then node 8 elseif x1>=-0.971031 then node 9 else 1.75
 6  fit = 1
 7  if x1<0.239886 then node 10 elseif x1>=0.239886 then node 11 else 3.75
 8  fit = 2
 9  fit = 1
10  fit = 3
11  fit = 4
```

```
view(rtree_3)
```

```
Decision tree for regression
1  if x1<-1.81039 then node 2 elseif x1>=-1.81039 then node 3 else 2.625
2  fit = 1
```

```
3  if x2<-0.0156483 then node 4 elseif x2>=-0.0156483 then node 5 else 3.16667
4  fit = 2.33333
5  fit = 4
```

```
view(rtree_4)
```

```
Decision tree for regression
1  if x1<0.239886 then node 2 elseif x1>=0.239886 then node 3 else 2.5
2  if x1<-0.971031 then node 4 elseif x1>=-0.971031 then node 5 else 1.6
3  fit = 4
4  fit = 1
5  fit = 2.5
```

```

```