ANN (Artificial Neural Network)

Aug 30, 2020

BIPLOV JHA
073BEL315

Assignment No. **#01**

Q. Perform PCA for the following data set.

(a) Using Matlab or any other computer program
(b) From manual calculation (consider variables x2, x3 and x4 only).

X1	X2	X3	X4	X5	X6
1350	79	393	161	870	165
1588	85	468	177	1110	160
1294	68	424	168	1050	152
1222	59	412	161	930	151
1585	98	439	164	1105	165
1297	82	429	169	1080	160
1796	79	449	169	1160	154
1565	55	424	163	1010	140
2664	128	452	173	1320	180
1166	55	399	157	815	140
1570	109	428	162	1060	175
1798	82	445	172	1160	158
1998	115	469	169	1370	160
1993	98	438	170	1080	167
1442	80	431	166	1129	144
1769	83	440	165	1095	165
1979	100	459	173	1120	173
1294	68	404	161	955	140

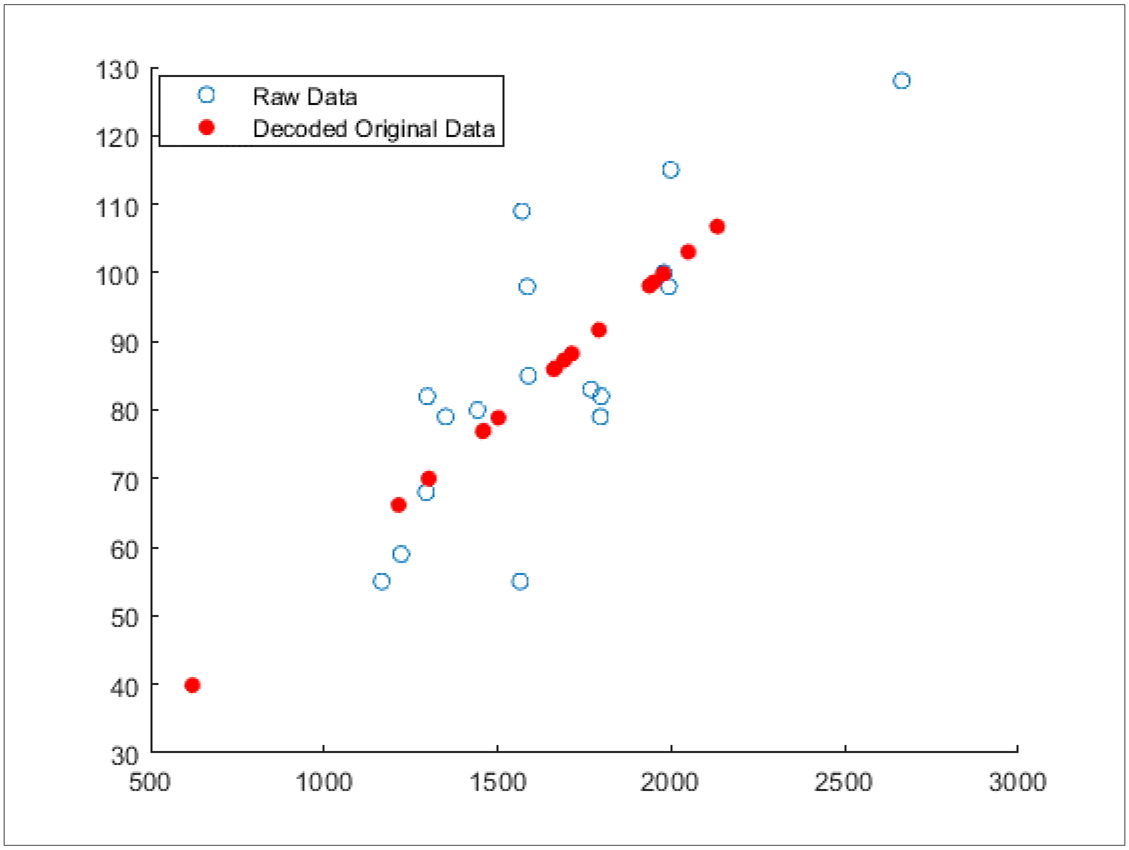
(c) How well PCA would have performed its job?

ANSWER: The PCA performance for the dataset given is as follows:

MATLAB CODE:

```
clear, close all, clc;

% Reading data from excel file
filename=input('ENTER EXCEL FILE LOCATION WITH FILENAME\n','s');
raw_x = xlsread(filename);

%Display Raw data
disp('Displaying Raw Data:');
raw_x

%Calculating Mean, Covariance from raw data
mean_x = mean(raw_x);
cov_x = cov(raw_x);

%size from raw data (m * n) size matrix
size_x = size(raw_x);
m = size_x(1,1)
n = size_x(1,2)

%calculating sum of{ ((x1 - meanx1)(x2-meanx2)) }/(m-1)
adjust_x = zeros(m,n);
for row = 1:m;
    for col = 1:n;
        adjust_x(row, col) = raw_x(row, col) - mean_x(col);
    end
end

%calculation of eigen vector and eigen value
[eig_vector, eig_value] = eig(cov_x);

%finding PC1
diag_eig = diag(eig_value);
[emax_val, emax_index] = max(diag_eig);
feature_vector = eig_vector(:, emax_index);

%Transform Data
disp('Final Data')
final_data = adjust_x * feature_vector;
final_data = -final_data

% Decoding Data
disp('Decoded Data')
Y = final_data * feature_vector'
alsoY = (final_data * feature_vector') + mean_x;
rawOriginalData = alsoY

[coef,score]=pca(raw_x);
coef = -coef;
score = -score

%Plot Results
figure;
scatter(raw_x(:,1), raw_x(:,2));
hold on;
scatter(alsoY(:,1), alsoY(:,2),'filled','red');
%hold on;
%scatter(Y(:,1), Y(:,2),'filled','d', 'black');
```

Output:




```
final_data =

   1.0e+03 *

    0.3313
    0.0311
    0.3320
    0.4365
    0.0361
    0.3195
   -0.1808
    0.0857
   -1.0589
    0.5243
    0.0632
   -0.1828
   -0.4370
   -0.3464
    0.1672
   -0.1362
   -0.3457
    0.3607

Decoded OriginalData =

   1.0e+03 *

    1.9481    0.0986    0.4472    0.1696    1.1748    0.1651
    1.6614    0.0859    0.4348    0.1669    1.0878    0.1589
    1.9488    0.0986    0.4473    0.1696    1.1750    0.1651
    2.0486    0.1031    0.4516    0.1705    1.2053    0.1672
    1.6661    0.0861    0.4350    0.1670    1.0893    0.1590
    1.9368    0.0981    0.4468    0.1695    1.1714    0.1648
    1.4590    0.0770    0.4260    0.1651    1.0264    0.1546
    1.7135    0.0882    0.4371    0.1674    1.1037    0.1600
    0.6204    0.0399    0.3896    0.1573    0.7720    0.1366
    2.1324    0.1068    0.4553    0.1713    1.2308    0.1690
    1.6921    0.0873    0.4361    0.1672    1.0972    0.1596
    1.4571    0.0769    0.4259    0.1650    1.0259    0.1545
    1.2143    0.0661    0.4154    0.1628    0.9522    0.1493
    1.3009    0.0700    0.4191    0.1636    0.9785    0.1512
    1.7913    0.0917    0.4404    0.1681    1.1273    0.1617
    1.5016    0.0789    0.4278    0.1655    1.0394    0.1555
    1.3015    0.0700    0.4192    0.1636    0.9787    0.1512
    1.9761    0.0998    0.4485    0.1699    1.1834    0.1657
```

```
score =

   1.0e+03 *

    0.3313    0.1203   -0.0184    0.0093   -0.0038   -0.0014
    0.0311   -0.0470    0.0006   -0.0314    0.0024   -0.0033
    0.3320   -0.0703    0.0033    0.0033   -0.0064   -0.0019
    0.4365    0.0241    0.0042    0.0000   -0.0041    0.0027
    0.0361   -0.0396   -0.0154   -0.0020    0.0026    0.0027
    0.3195   -0.0989   -0.0114    0.0015   -0.0056   -0.0023
   -0.1808   -0.0303    0.0164   -0.0032   -0.0016    0.0015
    0.0857    0.0477    0.0295    0.0024    0.0001    0.0021
   -1.0589    0.0717   -0.0018    0.0163   -0.0018   -0.0018
    0.5243    0.1181    0.0076   -0.0001    0.0071    0.0009
    0.0632   -0.0003   -0.0327    0.0016    0.0020    0.0026
   -0.1828   -0.0293    0.0115   -0.0007   -0.0046   -0.0015
   -0.4370   -0.1734   -0.0015    0.0070    0.0058    0.0022
   -0.3464    0.1041   -0.0022   -0.0038    0.0022   -0.0024
    0.1672   -0.1030    0.0063    0.0104    0.0038   -0.0020
   -0.1362    0.0245    0.0032   -0.0043   -0.0045    0.0049
   -0.3457    0.0594   -0.0052   -0.0202    0.0005   -0.0002
    0.3607    0.0221    0.0061    0.0139    0.0058   -0.0027
```

Question: How does PLS differ from PCA?

Answer:

PCA tries to explain the variance-covariance structure of a data set. Aim is to increase the variance of the features itself, like the loss of information is greatly reduced. PCA is a Dimensionality Reduction algorithm. Both PLS and PCA are used for dimension reduction. In PCA, the objective is to account for the maximum portion of the variance present in the original set of variables with a minimum number of composite variables called principal components.

PLS:

Partial Least Squares, use the annotated label to maximize inter-class variance. Principal components are pairwise orthogonal. Principal components are focus on maximize correlation. Partial Least Squares (PLS) regression is based on linear transition from a large number of original descriptors to a new variable space based on small number of orthogonal factors (latent variables). In other words, factors are mutually independent (orthogonal) linear combinations of original descriptors. PLS model contains the smallest necessary number of factors.

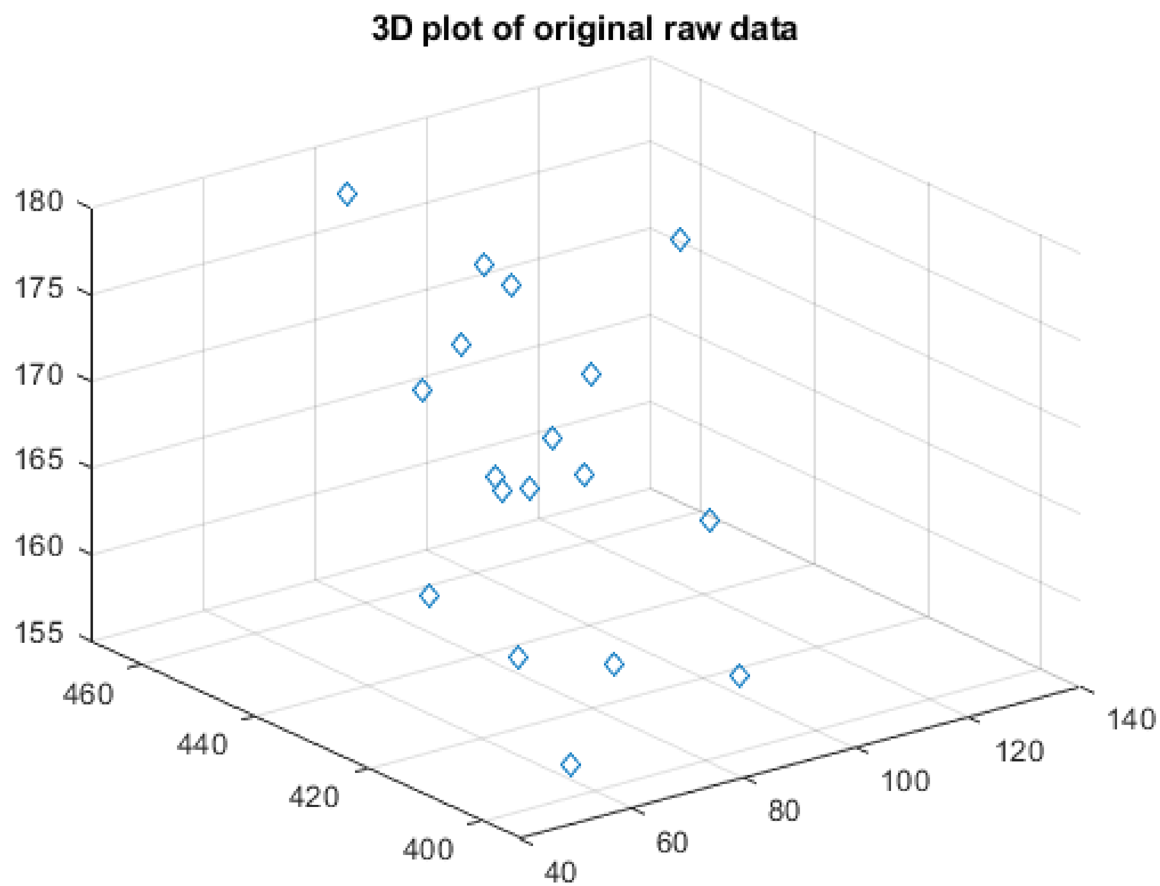
The main difference is that the PCA is unsupervised method and PLS is supervised method.

Second Calculation:

```
%Importing all data
rawdata = xlsread('HW.xlsx')
```

```
rawdata = 18×6

        1350          79         393         161         870         165
        1588          85         468         177        1110         160
        1294          68         424         168        1050         152
        1222          59         412         161         930         151
        1585          98         439         164        1105         165
        1297          82         429         169        1080         160
        1796          79         449         169        1160         154
        1565          55         424         163        1010         140
        2664         128         452         173        1320         180
        1166          55         399         157         815         140
```

```
%Scatter Plot of raw data
scatter3(rawdata(:,2),rawdata(:,3),rawdata(:,4),'diamond');
title("3D plot of original raw data");
```

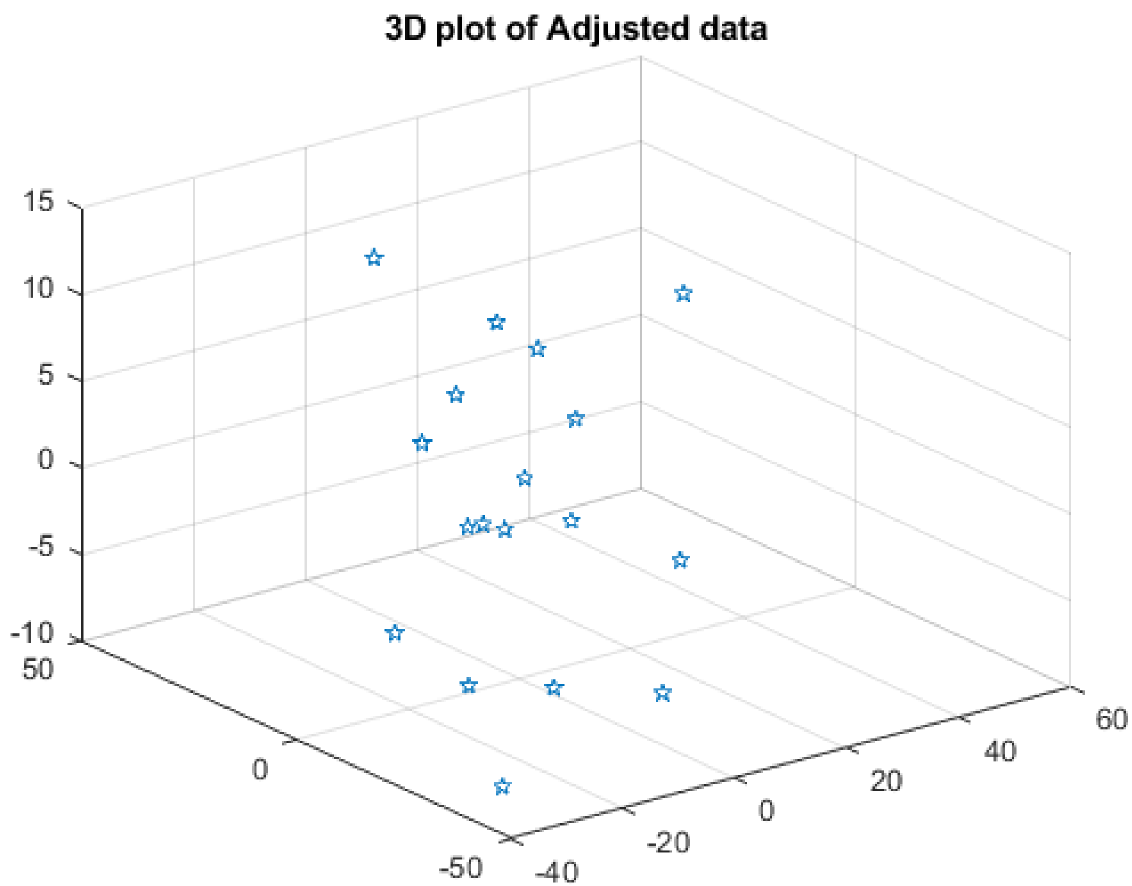



```
%Adjusted data
size_raw = size(rawdata);
row = size_raw(1,1);
col = size_raw(1,2);
adjustData = zeros(row, 3);
mean_x2 = mean(rawdata(:,2));
mean_x3 = mean(rawdata(:,3));
mean_x4 = mean(rawdata(:,4));
adjustData(:,1) = rawdata(:,2) - mean_x2;
adjustData(:,2) = rawdata(:,3) - mean_x3;
```

```
adjustData(:,3) = rawdata(:,4) - mean_x4;
adjustData
```

```
adjustData = 18×3

   -5.6111  -40.5000   -5.6667
    0.3889   34.5000   10.3333
  -16.6111   -9.5000    1.3333
  -25.6111  -21.5000   -5.6667
   13.3889    5.5000   -2.6667
   -2.6111   -4.5000    2.3333
   -5.6111   15.5000    2.3333
  -29.6111   -9.5000   -3.6667
   43.3889   18.5000    6.3333
  -29.6111  -34.5000   -9.6667
```

```
%Scatter Plot of Adjusted data
scatter3(adjustData(:,1),adjustData(:,2),adjustData(:,3),'pentagram');
title("3D plot of Adjusted data");
```




```
%Calculating covariance matrix
cov_x = cov(adjustData)
```

```
cov_x = 3×3

  415.1928  288.9118   56.3922
  288.9118  488.7353   99.7647
   56.3922   99.7647   28.2353
```

```
%Calculating eigen vector and eigen value
[eig_vector, eig_value] = eig(cov_x)
```

```
eig_vector = 3×3

    0.0146   -0.7608   -0.6488
   -0.2114    0.6319   -0.7457
    0.9773    0.1480   -0.1516

eig_value = 3×3

    7.5007         0         0
         0  164.2593         0
         0         0  760.4034
```

```
diag_eig = diag(eig_value)
```

```
diag_eig = 3×1

    7.5007
  164.2593
  760.4034
```

```
%Calculating Principle Component
e1 = diag_eig(1) %first eigen value
```

```
e1 = 7.5007
```

```
e2 = diag_eig(2) %second eigen value
```

```
e2 = 164.2593
```

```
e3 = diag_eig(3) %third eigen value
```

```
e3 = 760.4034
```

```
pvar1 = (e1/(e1+e2+e3))*100 %percentage variance of e1
```

```
pvar1 = 0.8047
```

```
pvar2 = (e2/(e1+e2+e3))*100 %percentage variance of e2
```

```
pvar2 = 17.6213
```

```
pvar3 = (e3/(e1+e2+e3))*100 %percentage variance of e3
```

```
pvar3 = 81.5740
```

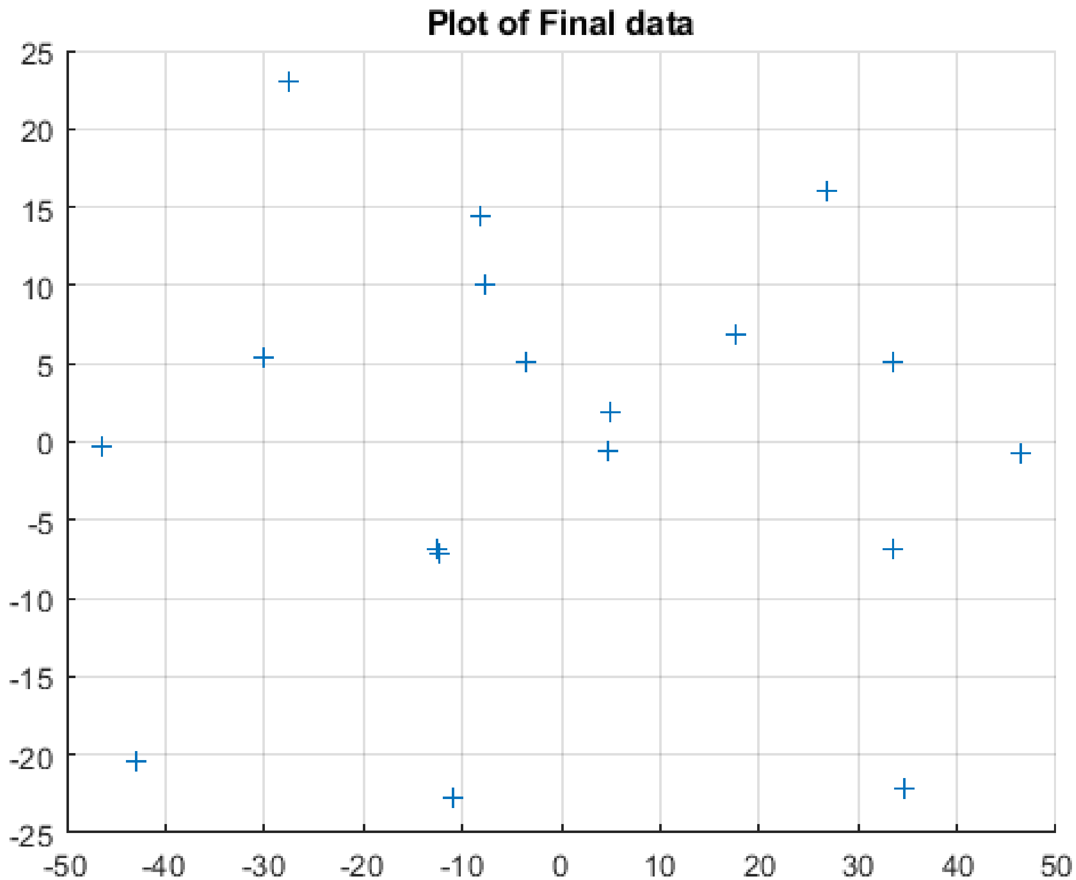
```
%Note:We know eigen vector with the highest eigen value is considered as
%the principal component of data set. Here, eigen vector with the largest
%eigen value was the third column. Also taking into account the second
%column of eigen vector as its eigen value is significant
feature_vector = [eig_vector(:,3), eig_vector(:,2)]
```

```
feature_vector = 3×2

   -0.6488   -0.7608
   -0.7457    0.6319
   -0.1516    0.1480
```

```
%Now we obtain the final data
final_data = adjustData * feature_vector
```

```
final_data = 18×2

   34.6997  -22.1614
  -27.5446   23.0340
   17.6597    6.8320
   33.5084    5.0602
  -12.3842   -7.1055
    4.6961   -0.5115
   -8.2711   14.4086
   26.8524   15.9821
  -42.9072  -20.3824
   46.4039   -0.7033
```

```
%Scatter Plot of Final data
scatter(final_data(:,1),final_data(:,2),'+');
title("Plot of Final data");
grid on;
```

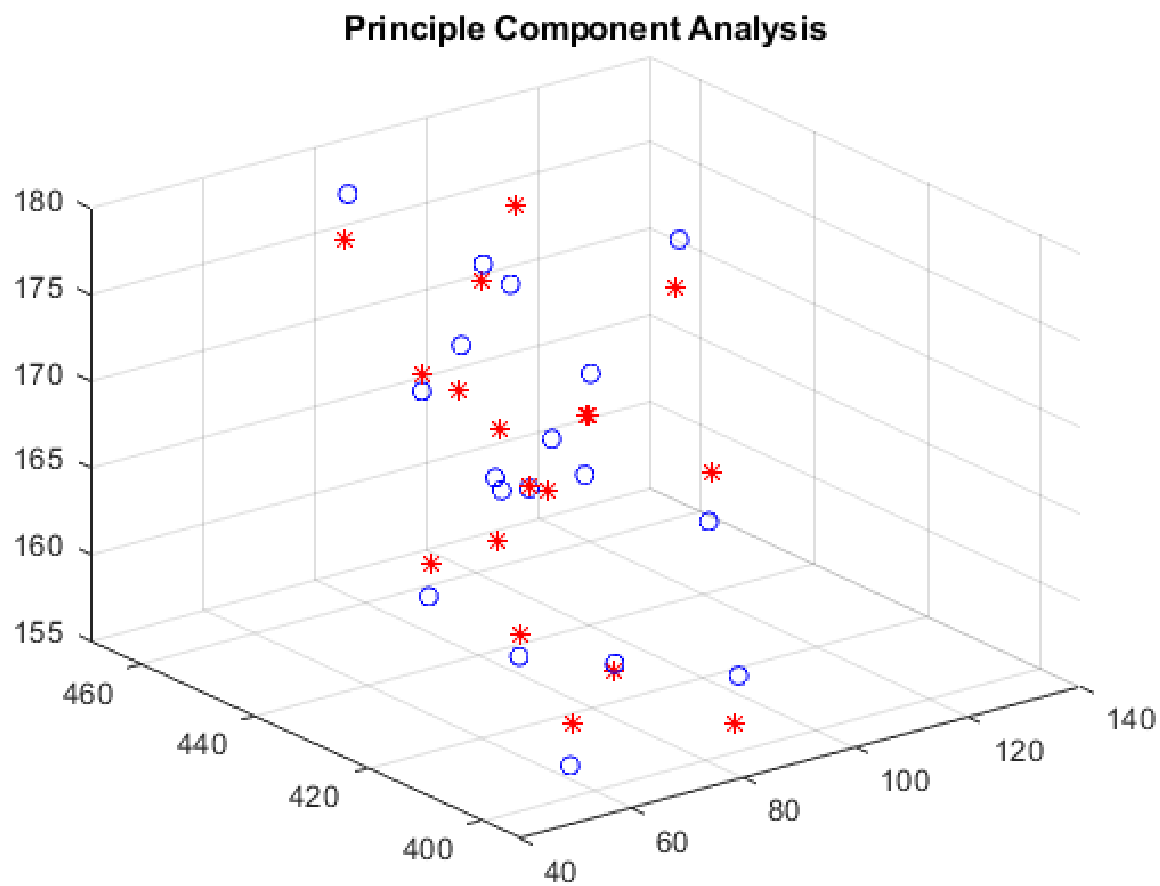



```
%Decoding data
decodedInput = (final_data * feature_vector');
decodedInput(:,1) = decodedInput(:,1) + mean_x2;
decodedInput(:,2) = decodedInput(:,2) + mean_x3;
decodedInput(:,3) = decodedInput(:,3) + mean_x4;
decodedInput
```

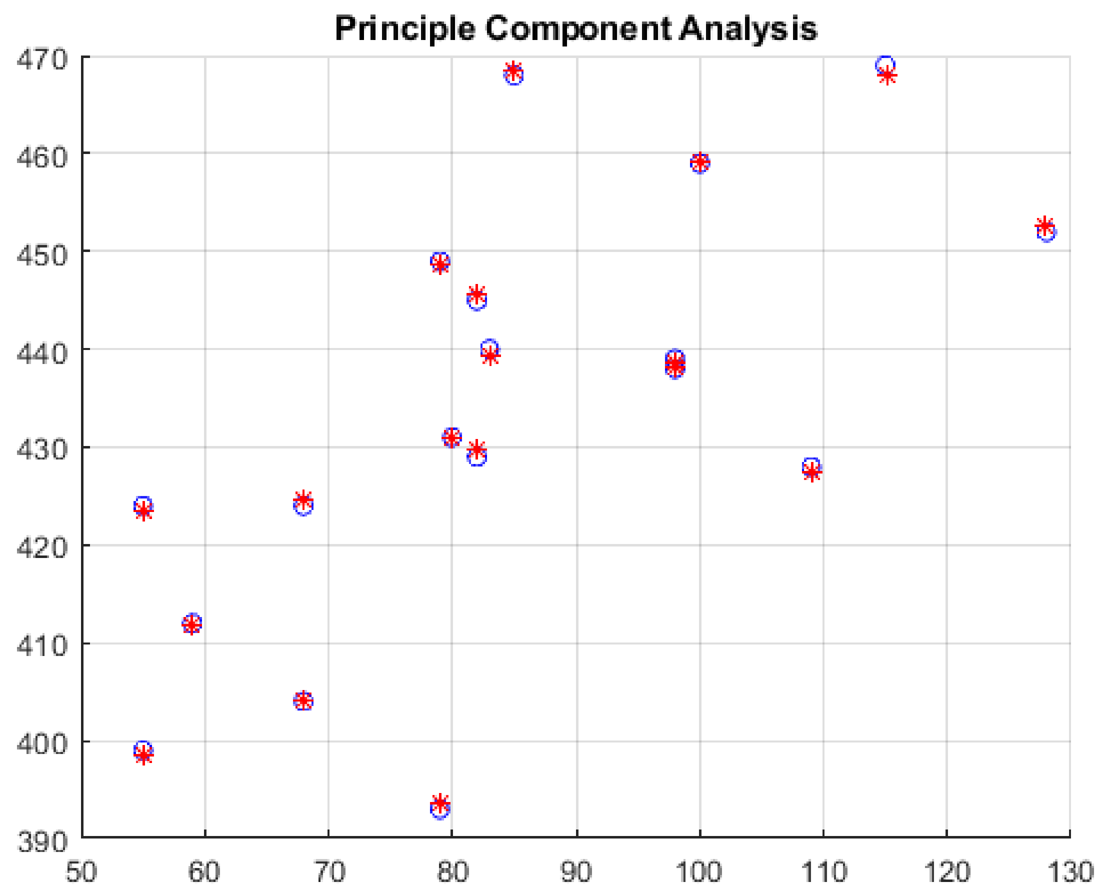
```
decodedInput = 18×3
```

```
   78.9570  393.6216  158.1261
   84.9589  468.5944  174.2519
   67.9552  424.6486  165.0013
   59.0200  411.7109  162.3366
   98.0522  438.2447  167.4919
   81.9533  429.6750  165.8791
   79.0158  448.7721  170.0535
   55.0293  423.5756  164.9624
  127.9574  452.6158  170.1531
   55.0378  398.4531  159.5287
```

```
%Plot showcasing Principle Component Analysis
figure;
scatter3(rawdata(:,2),rawdata(:,3),rawdata(:,4),'blue','o');
hold on;
scatter3(decodedInput(:,1),decodedInput(:,2),decodedInput(:,3),'red','*');
grid on;
title("3D - Principle Component Analysis");
```




```
%2DPlot showcasing Principle Component Analysis
figure;
scatter(rawdata(:,2),rawdata(:,3),'blue','o');
hold on;
scatter(decodedInput(:,1),decodedInput(:,2),'red','*');
grid on;
title("2D - Principle Component Analysis");
```


Principle Component Analysis
470
460
450
440
430
420
410
400
390
50
60
70
80
90
100
110
120
130