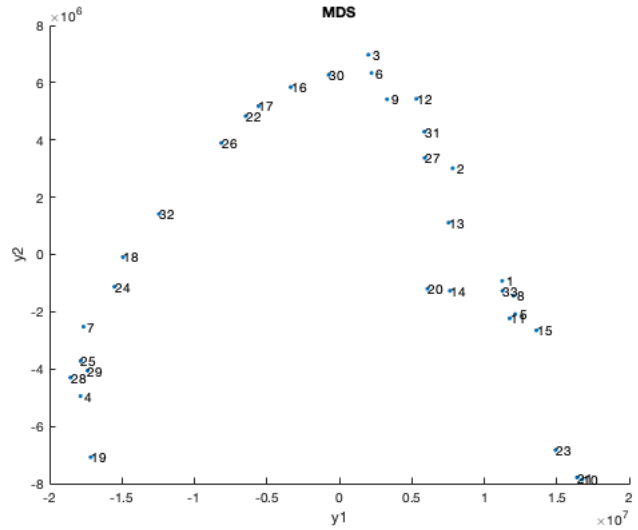
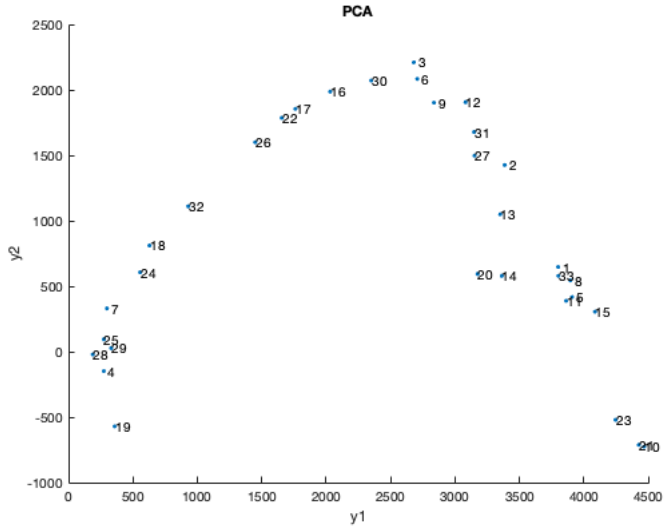
```
% Homework 7 - Question 1
X = csvread('faces.csv');
%colormap(gray); imagesc(reshape(X(10,:),112,92));
n = size(X,1);
disp("1a)")
Z = pdist(X, 'squaredeuclidean');
D = squareform(Z);

q=2;
Y = calculate_mds(D, q);
figure(1);
scatter(Y(:,1), Y(:,2), '.');
a = [1:n]'; b = num2str(a); c = cellstr(b);
dx = 0.1; dy = 0.1;
text(Y(:,1)+dx, Y(:,2)+dy, c);
title('MDS')
xlabel('y1');
ylabel('y2');
snapnow;

coeff = pca(X);
U = coeff(:,1:q);
Y = (X*U);
figure(2);
scatter(-Y(:,1), -Y(:,2), '.');
a = [1:n]'; b = num2str(a); c = cellstr(b);
dx = 0.1; dy = 0.1;
text(-Y(:,1)+dx, -Y(:,2)+dy, c);
title('PCA')
xlabel('y1');
ylabel('y2');
snapnow;
fprintf("The results of MDS with Sqaured Euclidan Distances and PCA are similar.\n");
disp("1b)")
k = 5;
idx = knnsearch(X, X, 'K', k+1);
idx = idx(:,2:end);
W = zeros(n);
for i=1:n
    W(i, idx(i,:)) = 1;
    W(idx(i,:),i) = 1;
end
D=diag(sum(W,2));
L = D - W;
[U,lambda] = eigs(L,D, q+1, 'smallestabs');
Y = U(:,2:end);

figure(3);
scatter(Y(:,1), Y(:,2), '.');
a = [1:n]'; b = num2str(a); c = cellstr(b);
dx = 0.1; dy = 0.1;
text(Y(:,1), Y(:,2), c);
title('Laplacian Eigenmaps')
xlabel('y1');
ylabel('y2');
snapnow;

disp("1c)")
D = distances(graph(W));
Y = calculate_mds(D, q);
figure(4);
scatter(Y(:,1), Y(:,2), '.');
a = [1:n]'; b = num2str(a); c = cellstr(b);
text(Y(:,1), Y(:,2), c);
title('Isomap')
xlabel('y1');
ylabel('y2');
snapnow;

disp("1d)")
figure(5)
set(gcf, 'PaperUnits','inches');
set(gcf, 'PaperSize',[10, 10]);
colormap(gray);
for i=1:n
    subplot(3, 11, i);
    imagesc(reshape(X(i,:),112,92));
    set(gca, 'XTickLabel', []);
    set(gca, 'YTickLabel', []);
end
snapnow;

fprintf("One key thing to note is that, each image is captured from different angle(head is rotated in images). Also in some images the head of th
fprintf("Looks like Laplacian Eigenmaps or Isomap is doing better than PCA and MDS. Group of the images which are very similar, show up close to e
function Y = calculate_mds(D, q)
    n = size(D,1);
    temp = eye(n) - (1/n)*ones(n);
    B = (-1/2) * temp * D * temp;
    [V, D] = eig(B);
    [D_sorted, D_order] = sort(diag(D), 'descend');
    V=V(:, D_order);
    U=V(:, 1:q);
    lambda = diag(D_sorted(1:q));
    Y = U*lambda;
end
```

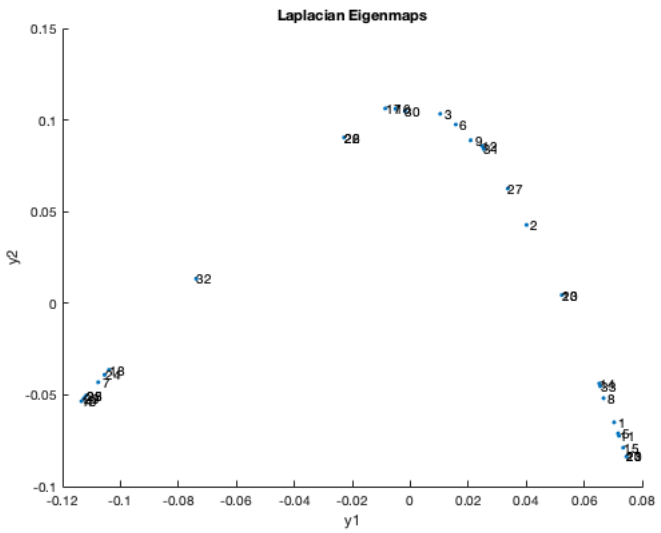
1a)



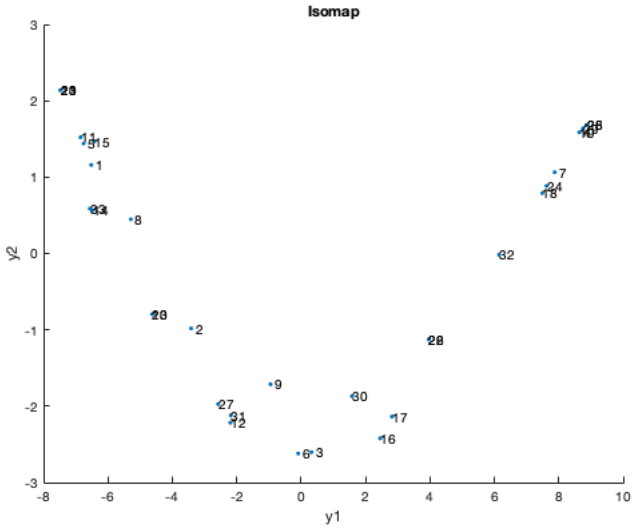


The results of MDS with Sqaured Euclidan Distances and PCA are similar.

1b)



1c)



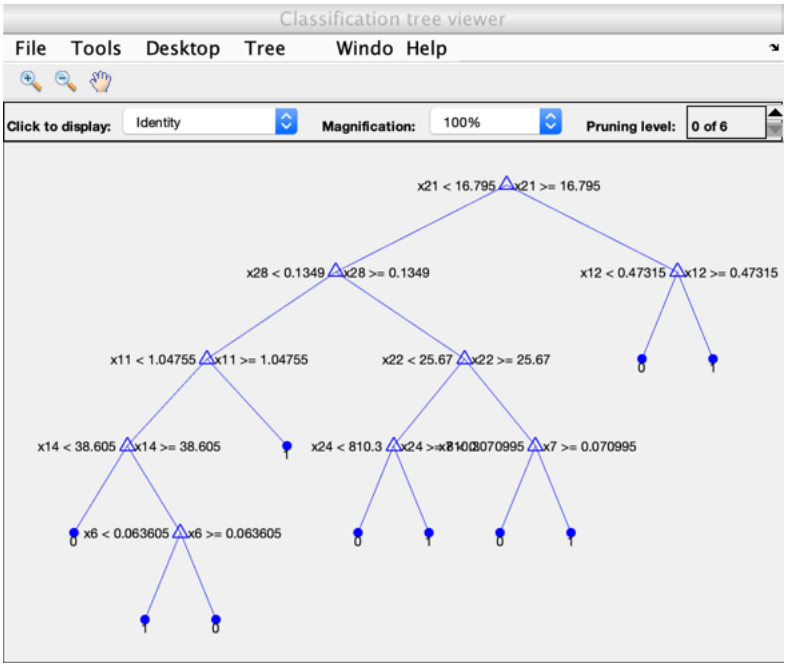
1d)

One key thing to note is that, each image is captured from different angle(head is rotated in images). Also in some images the head of the woman is tilted, however this is not very clear. These are the two properties could be captured in the dimensions.
Looks like Laplacian Eigenmaps or Isomap is doing better than PCA and MDS. Group of the images which are very similar, show up close to each other in Isomap and Laplacian Eigenmaps - this could to be attributed to the fact that they perform non-linear dimensionality reduction.

Published with MATLAB® R2018a

```
% Homework 7 - Question 2
data = readtable('bc_wisc.csv');
data = data.Variables;
train = data(1:400,:);
test = data(401:end,:);

disp('2a)');
X_train = train(:,3:end);
y_train = train(:,2);

X_test= test(:,3:end);
y_test = test(:,2);

tree = fitctree(X_train,y_train);
y_pred = predict(tree,X_test);
accuracy = sum(y_pred==y_test)/size(y_test,1);
fprintf("Fraction of test points correctly classified by decision tree = %f \n", accuracy);

disp('2b)');
view(tree, 'Mode', 'Graph')
snapnow;
top_features = tree.CutPredictor(1:7);
fprintf("Features that are considered as important by the decision tree : ");
for i=1:7
    val = top_features{i};
    if(~isempty(val))
        f = str2num(val(2:end));
        fprintf("%d,",f);
    end
end
fprintf("\n")

disp('2c)');
num_samples = 300;
y_pred_s = zeros(size(X_test,1),100);
top_features = zeros(30,1);
for i=1:100
    [~, idx] = datasample(X_train, num_samples);
    tree = fitctree(X_train(idx,:), y_train(idx,:));
    y_pred_s(:,i) = predict(tree,X_test);
    top_features = top_features + important_features(tree);
end
y_pred = mode(y_pred_s,2);
accuracy = sum(y_pred==y_test)/size(y_test,1);
fprintf("Fraction of test points correctly classified by bagging of decision trees = %f, which is better that the accuracy in 2b \n", accuracy);
[top_features_sorted , top_features_order] = sort(top_features,'descend');
tf = top_features_order(1:5);

disp('2d)');
fprintf("Features that are considered as important overall, ranked according to number of times they appear as important \n amonge 100 trees in th
for i=1:5
    fprintf("%d,", tf(i));
end
fprintf("\n");

function top_featues = important_features(tree)
    top_featues = zeros(30,1);
    cut_pred = tree.CutPredictor(1:7);
    for i=1:7
        val = cut_pred{i};
        if(~isempty(val))
            top_featues(str2num(val(2:end))) = 1;
        end
    end
end
```

```
2a)
Fraction of test points correctly classified by decision tree = 0.943750
2b)
```




```
Features that are considered as important by the decision tree : 21,28,12,11,22,
2c)
Fraction of test points correctly classified by bagging of decision trees = 0.962500, which is better that the accuracy in 2b
2d)
Features that are considered as important overall, ranked according to number of times they appear as important
 amonge 100 trees in the ensemble  : 28,21,2,23,11,
```

Published with MATLAB® R2018a