Maral Nourimand
30.11.2023
Data Mining Exercise Week 05

Question #1

```
% Load the Iris dataset
iris_data = dlmread('Iris.txt');

% Extract relevant data
features = iris_data(:, 2:5);  % Columns 2 to 5 are real data
classes = iris_data(:, 6);      % Class column

% Run PCA
% coeff contains the principal component coefficients.
% score contains the principal component scores.
% latent contains the eigenvalues of the covariance matrix.
% explained contains the percentage of variance explained by each principal
component.
[coeff, score, latent, ~, explained] = pca(features);
```

```
Percentage of variance explained by the first component: 92.46%
Percentage of variance explained by the second component: 5.30%
```

Question #2

```
% k 3:10 (number of nearest neighbors)
k_values = 3:10;

% Initialize a cell array to store feature rankings for each k
feature_rankings_cell = cell(1, length(k_values));

% Perform ReliefF algorithm for different values of k
for k_idx = 1:length(k_values)
    k = k_values(k_idx);

    % Use ReliefF algorithm
    [ranked_features, weights] = relieff(features, classes, k);

    % Store the feature rankings for this k
    feature_rankings_cell{k_idx} = ranked_features;

    % Display the ordered importance of variables for the current k
    fprintf('Ordered Importance of Variables for k = %d:\n', k);
    disp(ranked_features);
    fprintf('\n');
```

```
end
```

```
Ordered Importance of Variables for k = 3:
     4     2     1     3


Ordered Importance of Variables for k = 4:
     4     3     1     2


Ordered Importance of Variables for k = 5:
     4     1     3     2


Ordered Importance of Variables for k = 6:
     4     3     1     2


Ordered Importance of Variables for k = 7:
     4     3     1     2


Ordered Importance of Variables for k = 8:
     4     3     1     2


Ordered Importance of Variables for k = 9:
     4     3     2     1


Ordered Importance of Variables for k = 10:
     4     3     2     1
```

Question #3

```
% Separate training and test data
training_data = [];
training_labels = [];
test_data = [];
test_labels = [];

for i = 1:3
    class_data = features(classes == i, :);

    % Training data: First 40 cases
    training_data = [training_data; class_data(1:40, :)];
    training_labels = [training_labels; repmat(i, 40, 1)];
```

```
    % Test data: Remaining 10 cases
    test_data = [test_data; class_data(41:end, :)];
    test_labels = [test_labels; repmat(i, 10, 1)];
end

% Train Naïve Bayesian classifier
nb_model = fitcnb(training_data, training_labels);

% Predict on the test set
predicted_labels = predict(nb_model, test_data);

% Display the confusion matrix
confusionMat = confusionmat(test_labels, predicted_labels);
disp('Confusion Matrix:');
disp(confusionMat);
```

```
Confusion Matrix:
    10     0     0
     0    10     0
     0     0    10
```

Question #4

```
% use the first two principal components of PCA
selected_features = training_data * coeff(:, 1:2);

% Train Naïve Bayesian classifier
nb_model = fitcnb(selected_features, training_labels);

% Apply PCA transformation to test data
test_data_pca = test_data * coeff(:, 1:2);

% Predict on the test set
predicted_labels = predict(nb_model, test_data_pca);

% Display the confusion matrix
confusionMat = confusionmat(test_labels, predicted_labels);
disp('Confusion Matrix:');
disp(confusionMat);
```

```
Confusion Matrix:
    10     0     0
     0    10     0
     0     2     8
```

Question #5

Assuming that m=20 and the given amino acids: {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}
For GGKAPAMM sequence we'd have:

G:00000100000000000000
G:00000100000000000000
K:00000000100000000000
A:10000000000000000000
P:00000000000010000000
A:10000000000000000000
M:00000000001000000000
M:00000000001000000000

The disadvantage of encoding (one-hot encoding):

- potential high dimensionality of the input space. As *m* increases, the neural network has to handle a larger number of input nodes.
- increase in computational complexity
- memory requirements.

For binary data, Manhattan distance can be suitable as it measures the "distance" as the total number of bit differences between two binary vectors. It is the count of positions at which the corresponding bits are different.

For binary data, Euclidean distance may not be the most suitable choice, as it assumes a continuous scale and can be sensitive to the magnitude of differences. Binary data doesn't have magnitudes; it's more about presence or absence.

For binary data, Chebyshev distance can be suitable as it considers only the maximum difference between corresponding bits. It represents the "worst-case" scenario in terms of bit differences between two binary vectors.

If each position in the amino acid sequence GGKAPAMM can be any of the 20 possible amino acids, there would be 20^8 different symbol sequences.
$20^8 = 2.56 \times 10^{10}$

Question #6

Considering the text analysis in the provided example, one possible application would be to develop a model to automatically classify factory reports into different categories based on the provided text

descriptions. The goal is to predict the type or cause of the reported issue, enabling proactive maintenance interventions.