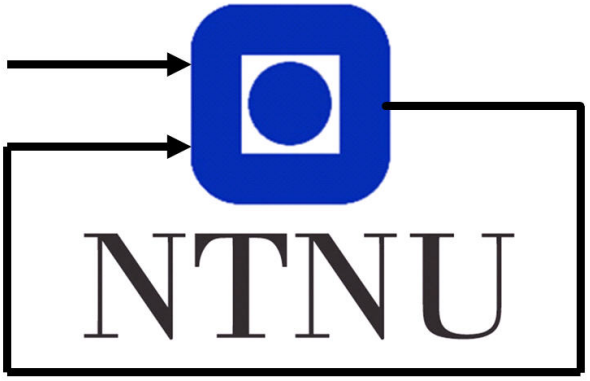
TTK4205: Prosjektoppgave 1

Per Arne Kjelsvik

November 19, 2017



Department of Engineering Cybernetics

Contents

1 Prosjektoppgave

1.1 Gjennomføring av oppgaven

Først må vi hente ut dataen og inndele i både *testsett* og *treningssett*:

Listing 1: `MATLAB` kode for å hente ut data

```
%% 2 - Datasett
% Open files and extact data (ready to use with Classifcation Learner app)
[train_1, test_1, true_1] = data2Table('ds-1.txt');
[train_2, test_2, true_2] = data2Table('ds-2.txt');
[train_3, test_3, true_3] = data2Table('ds-3.txt');
```

Funksjonen `data2Table` splitter datasettene og setter inn i tabeller (full funksjon i appendix REF). Om datasettene inneholder objekter i vilkårlig rekkefølge sorteres de etter klassetilhørighet. Vi kan nå begynne å hente ut data:

Listing 2: Bruk naermeste nabo til å klassifisere og finn beste egenskapskombinasjon

```
%% 5 - Gjenommforing av oppgaven
% Run closest neighbor classifcator, get best property combination
[NN_1, prop_1, C_NN_1, T_1] = closestNeighbor(train_1, test_1, true_1);
[NN_2, prop_2, C_NN_2, T_2] = closestNeighbor(train_2, test_2, true_2);
[NN_3, prop_3, C_NN_3, T_3] = closestNeighbor(train_3, test_3, true_3);
```

Variablene `T_1`, `T_2` og `T_3` inneholder sortert rekkefølge av feilratene for de forskjellige kombinasjonene, vist i tabell 1, tabell 2, og tabell 3, hvor v_i er de forskjellige egenskapene, og hvor 1 betyr *på*, mens 0 betyr *av*. `prop_1`, `prop_2` og `prop_3` er vektorer som består av de beste kombinasjonene som brukes videre til de andre klassifikatorene. `NN_1`, `NN_2`, og `NN_3` er beste nærmeste-nabo klassifisering.

feilrate	**v1**	**v2**	**v3**	**v4**
0.093	1	1	1	1
0.100	1	1	0	1
0.127	1	0	1	1
0.147	1	1	1	0
0.167	1	0	0	1
0.180	1	1	0	0
0.193	1	0	1	0
0.213	0	1	1	1
0.227	0	1	0	1
0.240	1	0	0	0
0.300	0	0	1	1
0.320	0	1	1	0
0.360	0	1	0	0
0.387	0	0	0	1
0.433	0	0	1	0

Table 1: Feilrate for forskjellige kombinasjoner av egenskaper, datasett 1

feilrate	v1	v2	v3
0.013	1	1	0
0.020	1	1	1
0.180	1	0	0
0.193	1	0	1
0.280	0	1	0
0.287	0	1	1
0.493	0	0	1

Table 2: Feilrate for forskjellige kombinasjoner av egenskaper, datasett 2

feilrate	v1	v2	v3	v4
0.075	0	1	1	1
0.095	0	1	1	0
0.095	1	1	1	1
0.100	1	1	1	0
0.150	1	0	1	1
0.170	1	0	1	0
0.190	0	0	1	1
0.200	1	1	0	1
0.215	1	1	0	0
0.240	0	1	0	1
0.285	1	0	0	1
0.310	0	1	0	0
0.330	1	0	0	0
0.345	0	0	1	0
0.395	0	0	0	1

Table 3: Feilrate for forskjellige kombinasjoner av egenskaper, datasett 3

Koden for implementeringen av nærmeste-nabo klassifikatoren kan finnes i Appendix A i kodesnutt A.1. Full kode for å se hvordan denne brukes til å finne beste egenskapskombinasjon kan sees i Appendix B B.1. Videre skal vi bruke minimum feilrate klassifikator til å klassifisere testsettene:

Listing 3: Bruk minimum feilrate til å klassifisere testsett

```
% Run min error classificator
[minErr_1, err_minErr_1, C_minErr_1] = minErrorClassificator(...
    train_1, test_1, prop_1, true_1);
[minErr_2, err_minErr_2, C_minErr_2] = minErrorClassificator(...
    train_2, test_2, prop_2, true_2);
[minErr_3, err_minErr_3, C_minErr_3] = minErrorClassificator(...
    train_3, test_3, prop_3, true_3);
```

Koden for implemteringen av minimum feilrate klassifikator kan finnes i Appendix A i kodesnutt A.3. Og til slutt har vi minste kvadraters metode:

Listing 4: Bruk minste kvadraters metode til å klassifisere testsett

```
% Run least squares method classificator
[LS_1, err_LS_1, C_LS_1] = leastSquaresMethod(...
    train_1, test_1, prop_1, true_1);
[LS_2, err_LS_2, C_LS_2] = leastSquaresMethod(...
    train_2, test_2, prop_2, true_2);
[LS_3, err_LS_3, C_LS_3] = leastSquaresMethod(...
    train_3, test_3, prop_3, true_3);
```

Og implemteringen av minste kvadraters metode kan også finnes i Appendix A i kodesnutt A.2. For hvert datasett er det altså brukt disse kombinasjonene:

Datasett 1			
1	1	1	1
Datasett 2			
1	1	0	x
Datasett 3			
0	1	1	1

Table 4: Beste egenskapskombinasjoner

Og i tabell 5 nedenfor er det listet opp hvordan hver klassifikator presterte med disse egenskapskombinasjonene. Som vi kan se, så er minste kvadraters metode best for datasett 1, men dårligst for datasett 2 og 3, hvor nærmeste-nabo er best. Minimum feilrate havner i midten på klassifisering for hvert datasett.

Datasett 1	
feilrate	**klassifikator**
0.073	Minste kvadrater
0.080	Minimum feilrate
0.093	Nærmeste-nabo
Datasett 2	
feilrate	**klassifikator**
0.013	Nærmeste-nabo
0.020	Minimum feilrate
0.120	Minste kvadrater
Datasett 3	
feilrate	**klassifikator**
0.075	Nærmeste-nabo
0.130	Minimum feilrate
0.160	Minste kvadrater

Table 5: Beste klassifikatorer

1.2 Kortfattet svar på spørsmål

Til slutt skal det svares kortfattet på noen spørsmål:

1. **Hvorfor er det fornuftig å benytte nærmeste-nabo klassifikatoren til å finne gunstige egenskapskombinasjoner?**

   Fordi nærmeste-nabo baserer seg på om egenskapene befinner seg nærme hverandre. Så hvis du har datasett hvor egenskap 1 og 2 f. eks. er blandet sammen, mens egenskap 1 og 3 er tydelig lokalisert i hvert sitt område, så vil nærmeste-nabo oppdage dette. Siden den gjør lite antagelser og er ikke-parametrisk, bør den gi en grei feilrate, uavhengig av fordelingen til datasettet.

2. **Hvorfor kan det i en praktisk anvendelse være fornuftig å finne en lineær eller kvadratisk klassifiaktor til erstatning for nærmeste-nabo klassifikatoren?**

   Nærmeste-nabo krever en del regnekraft, og gjør ingen antakelser om settet. Ved anveldelse kan det godt tenkes at dette er informasjon som vi kan bruke til vår fordel, og da lønner det seg kanskje heller med klassifikatorer som tar nytte av informasjonen.

3. **Hvorfor er det lite gunstig å bruke samme datasett både til trening og evaluering av en klassifikator?**

   Fordi evalueringen vår vil da basere seg på data vi allerede har brukt til å trene modellen. Dette gir et tydelig bias. Derfor er det bedre å trene en klassifikator med et sett, og se hvordan klassifikatoren preseterer med et testsett. På den måten kan vi objektiv sammenlikne hvor gode forskjellige klassifikatorer er.

4. **Hvorfor gir en lineær klassifikator dårlige resultater for datasett 2?**

   Om vi ser på dataen fra datasett 2 (figur 1), så kan vi se at det vanskelig lar seg gjøre å skille objektene i lineære områder. Dette fordi variabel 1 på en måte bretter seg rundt variabel 2, som er samlet som en kjerne. Formen blir som en halvsirkel som går rundt en sirkel.

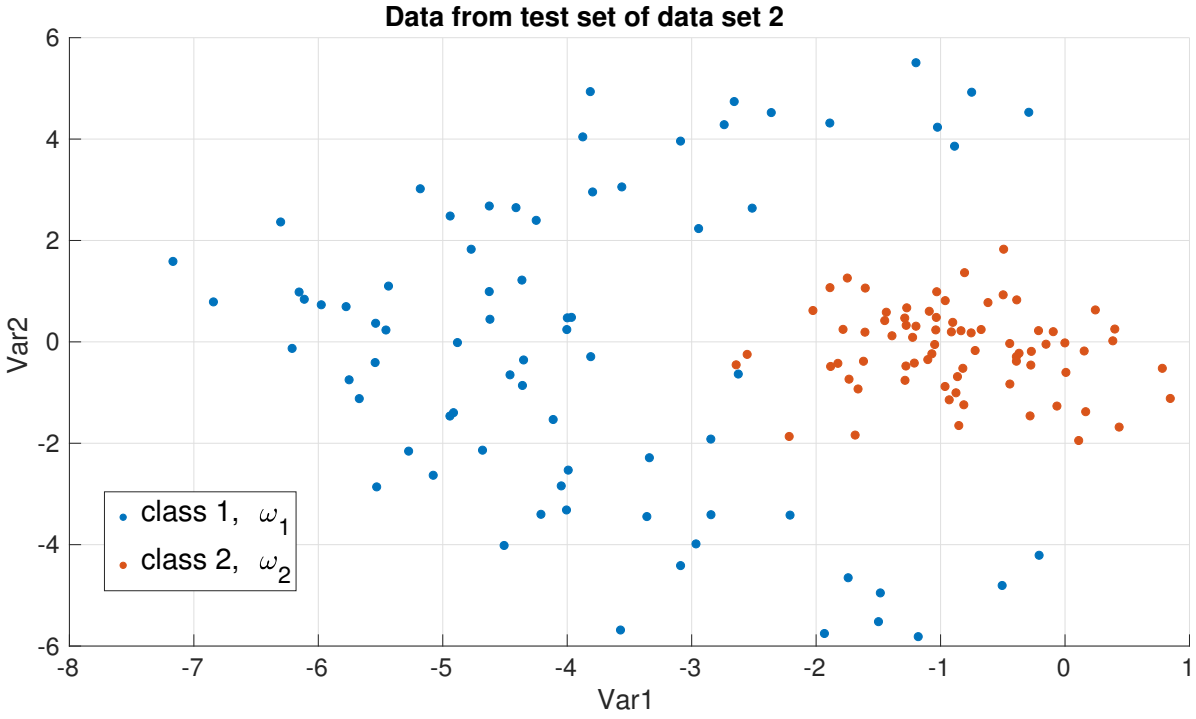


Figure 1: Scatter plot av egenskap 1 og egenskap 2 for testsettet i datasett 2

Har også valgt å legge ved forvirringsmatrisene for de beste klassifikatorene:

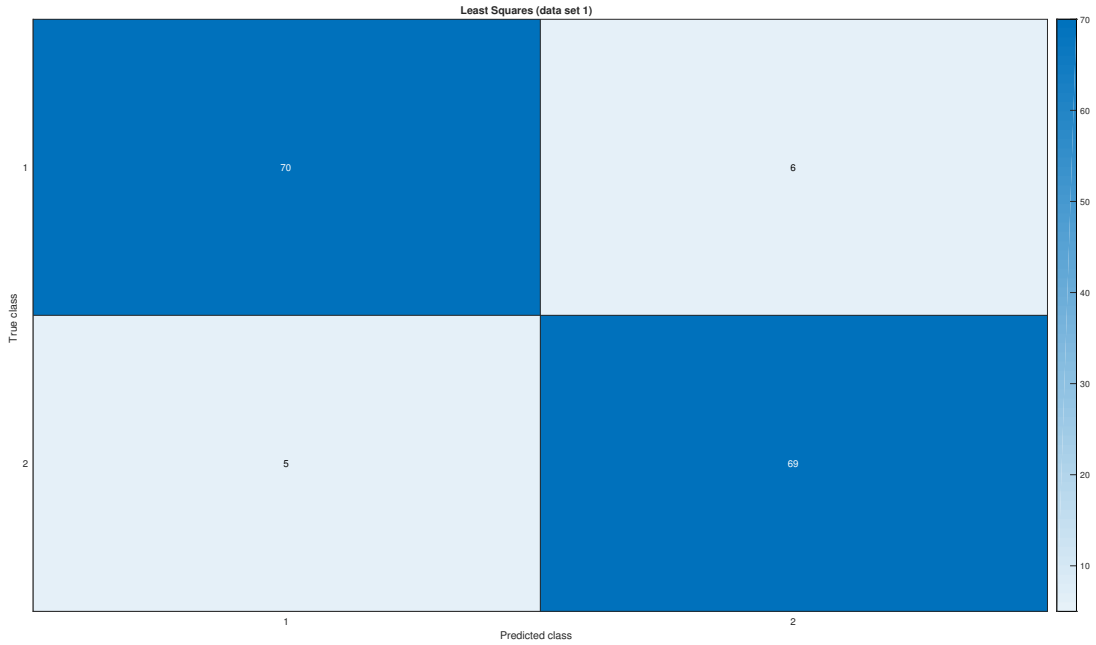


Figure 2: Forvirringsmatrise minste kvadraters datasett 1

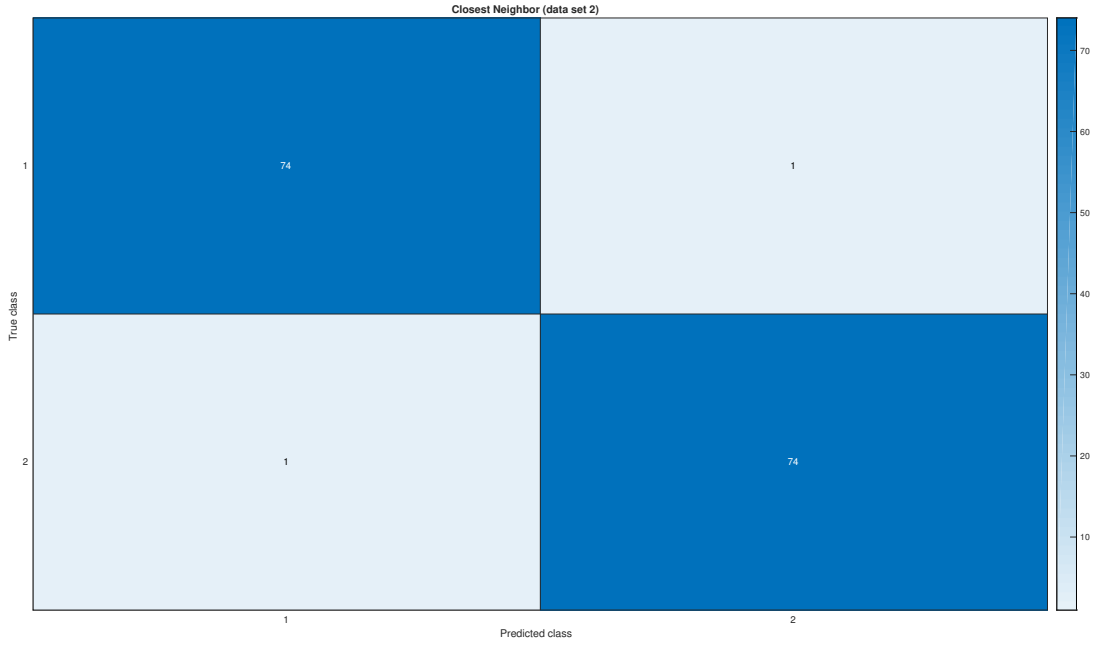


Figure 3: Forvirringsmatrise nærmeste nabo datasett 2

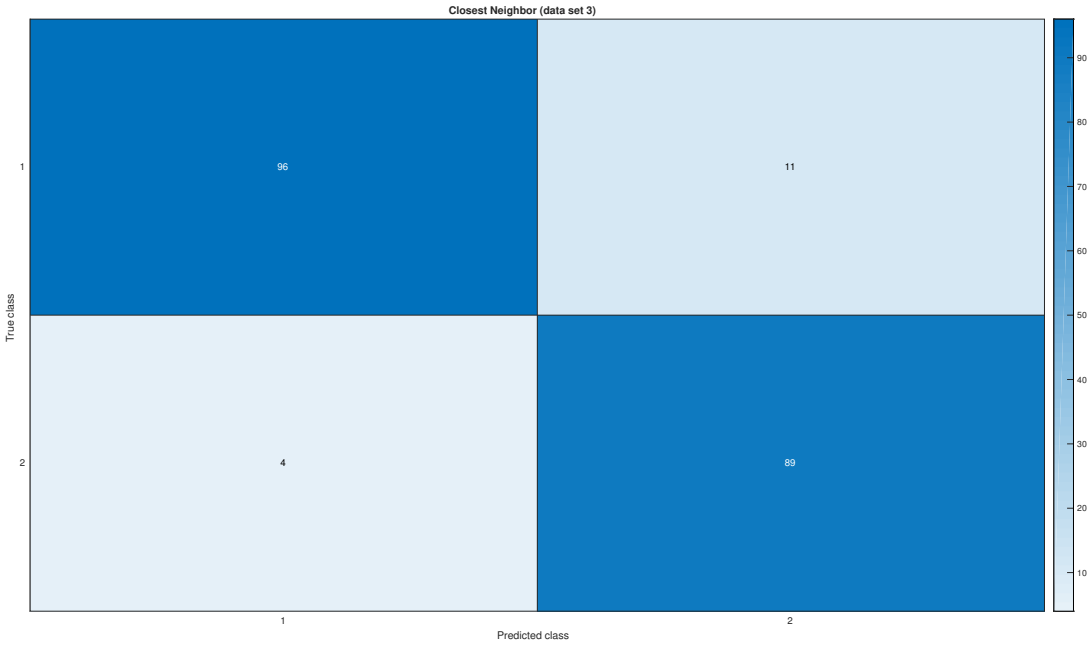


Figure 4: Forvirringsmatrise nærmeste nabo datasett 3

A Appendix A - MATLAB-kode for implementering av klassifikatorer

A.1 Nærmeste nabo klassifikator (kode)

Listing 5: MATLAB-kode nærmeste nabo klassifikator

```
function [inProgress] = closestNeighborClassificator(train,test, active)
n = size(train.class(:),1);

% Extract x
[x_1, x_2, x_test] = extractX(train, test, active);
x = [x_1 x_2];
b = zeros(n,1);

% Classify objects
for obj=1:n
    min_norm = inf;
    for neighbor=1:n
        obj_norm = norm(x_test(:,obj) - x(:,neighbor));
        if obj_norm < min_norm
            min_norm = obj_norm;
            if neighbor<=length(x_1)
                b(obj) = 1;
            else
                b(obj) = 2;
            end
        end
    end
end

inProgress = b;
```

A.2 Minste kvadraters metode (kode)

Listing 6: MATLAB-kode for minste kvadraters metode

```
function [inProgress, errRate, confusion] = leastSquaresMethod(train,...
    test, active, true_class)
n = size(train.class(:),1);

% Extract x and preallocate variables
[x_1, x_2, x_test] = extractX(train, test, active);
[d, n_1] = size(x_1);
Y = zeros(n,d+1);
b = zeros(n,1);

% Expand Y and fill out b
for i=1:n_1
    % For x_1
    Y(i,:) = [1 x_1(:,i)'];
    b(i) = 1;

    % For x_2
```

```
    Y(n_1+i,:) = [1 x_2(:,i)'];
    b(n_1+i) = -1;
end

% Calculate expanded weight vector a
a = inv(Y'*Y)*Y'*b;

% Classify
classify = zeros(n,1);

for i=1:n
   y = [1 x_test(:,i)']';
   g = a'*y;

   if g >= 0
       classify(i,1) = 1;
   else
       classify(i,1) = 2;
   end
end

% Return result
inProgress = classify;
[errRate, confusion] = getErrRate(classify, true_class);
end
```

A.3 Minimum feilrate klassifikator (kode)

Listing 7: MATLAB-kode for minimum feilrate klassifikator

```
function [inProgress, errRate, confusion] = minErrorClassificator(train,...
    test, active, true_class)
n = size(train.class(:),1);
[x_1, x_2, x_test] = extractX(train, test, active);
[d_1, n_1] = size(x_1);
[d_2, n_2] = size(x_2);

% Mean vectors
mu_1 = 1/n_1 * sum(x_1');
mu_2 = 1/n_2 * sum(x_2');
mu_1 = mu_1';
mu_2 = mu_2';

% Covariance matrices
sigma_1 = zeros(d_1);
sigma_2 = zeros(d_2);

for k=1:n_1
    sigma_1 = sigma_1 + (x_1(:,k)-mu_1)*(x_1(:,k)-mu_1)';
end
sigma_1 = 1/n_1 * sigma_1;

for k=1:n_2
    sigma_2 = sigma_2 + (x_2(:,k)-mu_2)*(x_2(:,k)-mu_2)';
end
```

```
sigma_2 = 1/n_2 * sigma_2;

% Discriminant function parameters
W_1 = -0.5*inv(sigma_1);
W_2 = -0.5*inv(sigma_2);
w_1 = inv(sigma_1)*mu_1;
w_2 = inv(sigma_2)*mu_2;
w_1_0 = -0.5*mu_1'*inv(sigma_1)*mu_1 - 0.5*log(det(sigma_1)) + log(n_1/n);
w_2_0 = -0.5*mu_2'*inv(sigma_2)*mu_2 - 0.5*log(det(sigma_2)) + log(n_2/n);


% Classifcation
classify = zeros(n,1);

for i=1:n
   g_1 = x_test(:,i)'*W_1*x_test(:,i) + w_1'*x_test(:,i) + w_1_0;
   g_2 = x_test(:,i)'*W_2*x_test(:,i) + w_2'*x_test(:,i) + w_2_0;

   g = g_1 - g_2;

   if g >= 0
       classify(i,1) = 1;
   else
       classify(i,1) = 2;
   end
end

% Return result
inProgress = classify;
[errRate, confusion] = getErrRate(classify, true_class);

end
```

B Appendix B - MATLAB-funksjoner brukt i utførelsen av oppgaven

B.1 Nærmeste nabo script (kode)

Listing 8: MATLAB-funksjon for å kjøre nærmeste nabo og finne feilrater osv.

```
function [inProgress, properties, C, T] = closestNeighbor(train,test,...
    true_class)
n = size(train.class(:),1);
d = size(train.Properties.VariableNames(:),1)-1;
T = table();

% Make all possible combinations of properties
combinations = getcondvects(d);
combinations(1,:) = [];
classified = zeros(n,size(combinations,1));

% Run closest neighbor and find combination with least errors
min_err = inf;
errRates      = zeros(1,size(combinations,1));

for combo=1:size(combinations,1)
    % Run closest neighbor classificator
    classified(:,combo) = ...
        closestNeighborClassificator(train,test,combinations(combo,:));

    % Find the error rate of the classification
    [errRates(combo), confusion] = getErrRate(classified(:,combo),...
        true_class);

    % Update best classification
    if errRates(combo) < min_err
        min_err = errRates(combo);
        best_combination = combo;
        C = confusion;
    end
end

% Return best combination and accompying classification
properties = combinations(best_combination,:);
inProgress = classified(:, best_combination);

% Sort the combinations of properties and put in Table with error rates
[~, order] = sort(errRates);
errRates = sort(errRates);
combinations = combinations(order', :);
T{:,1} = errRates';
T.Properties.VariableNames{1} = 'Error';

for j=2:d+1
   T{:,j} = combinations(:,j-1);
   T.Properties.VariableNames{j} = insertAfter('var','r',num2str(j-1));
end

end
```

B.2 data2Table-funksjon

Listing 9: MATLAB-funksjon for å hente ut data

```
function [train, test, true_class] = data2Table(set)
    fid = fopen(set);
    train = {};
    test  = {};
    row1 = 0;
    row2 = 0;
    iteration = 0;
    true_class = [];

    while ~feof(fid)
        iteration = iteration + 1;
        tline = fgetl(fid);
        [token, remain] = strtok(tline);
        remain = str2num(remain);
        if mod(iteration,2) == 1        % Training set (odd numbered)
            row1 = row1+1;
            train{row1,1} = insertAfter('class_','_',token);
            for i=1:length(remain)
               train{row1,i+1} = remain(i);
            end
        else                            % Test set (even numbered)
            row2 = row2 + 1;
            test{row2,1} = insertAfter('class_','_',token);
            for i=1:length(remain)
               test{row2,i+1} = remain(i);
            end
            true_class(row2, 1) = str2num(token);
        end
    end

    fclose(fid);

    % Convert cells to tables
    train = cell2table(train);
    test  = cell2table(test);
    train.Properties.VariableNames{1} = 'class';
    test.Properties.VariableNames{1} = 'class';
    for i=2:length(test.Properties.VariableNames)
       train.Properties.VariableNames{i} = insertAfter('var',...
           'r',num2str(i-1));
       test.Properties.VariableNames{i} = insertAfter('var',...
           'r',num2str(i-1));
    end

    % Sort sets so that class 1 comes first, then class 2
    train       = sortrows(train,1);
    test        = sortrows(test,1);
    true_class  = sortrows(true_class);
end
```

B.3 extractX-funksjon

Listing 10: MATLAB-funksjon for å hente ut treningsett og testsett

```
function [v1, v2, v3] = extractX(train, test, active)
d = size(train.Properties.VariableNames(:),1)-1;
v1 = zeros();
v2 = zeros();

% Making property vectors and test vector
d = size(train.Properties.VariableNames(:),1)-1;
v3 = test{:,2:end};

for i=1:size(train.class(:),1)
   if train.class{i} == 'class-1'
        v1(end+1,1:d) = train{i,2:end};  % Class 1
   else
        v2(end+1,1:d) = train{i,2:end};  % Class 2
   end
end

% Delete first empty row and transpose
v1(1,:) = [];
v2(1,:) = [];
v1 = v1';
v2 = v2';
v3 = v3';

% Remove inactive properties
inactive= active == zeros(1,d);
[~,del] = find(inactive);
if ~isempty(del)                % Only remove if 1 or more is inactive
    for var=length(del):-1:1
        v1(del(var), :) = [];
        v2(del(var), :) = [];
        v3(del(var), :) = [];
    end
end

end
```

B.4 getcondvects-funksjon

Listing 11: MATLAB-funksjon for å generere en sannhetstabell med alle mulige egenskaskombinasjoner

```
function [condvects] = getcondvects(i)
    % GETCONDVECTS returns a matrix of binary condition vectors.
    % GETCONDVECTS(I) returns a matrix of all possible binary condition
    % vectors for a logical system with (I) inputs.
    % INPUT: (I) shall be an integer >= 1
    % OUTPUT: CONDVECTS is a binary matrix of size [2^I,I]
    % METHOD: The method uses three nested FOR loops that work to target
    % only those cells whose value should be true (which of course is 50%
    % of the matrix).  Therefore, the algorithm scales as [0.5*i*2^i]
```

```
    % which is optimal.  Furthermore, the algorithm has low memory
    % footprint and overhead.
    % Copyright 2011, Paul Metcalf
    % Acknowledgements: James Tursa and Nico Schlmer

    g = 2;
    i2 = 2^i;
    condvects = false(i2,i);
    for m = 1 : 1 : i
        m2 = 2^m;
        m3 = (m2/2)-1;
        i3 = i-m+1;
        for g = g : m2 : i2
            for k = 0 : 1 : m3
                condvects(g+k,i3) = true;
            end
        end
        g = m2+1;
    end
end
```

B.5 getErrRate-funksjon

Listing 12: MATLAB-funksjon som beregner feilraten til en klassifikator og finner forvirringsmatrisen

```
function [errRate, confusion] = getErrRate(classified, true_class)
% Set up confusion matrix and other variables
n       = size(classified, 1);      % Number of objects
c       = 2;                        % Number of classes
C       = zeros(c,c);               % Confusion matrix of size cxc
x       = true_class;               % Sorted: [1 1 ... 1 2 2 ... 2]'

% Find number of class 1 and 2 objects
[n_x1, ~]   = find(true_class == 1);
n_x1        = n_x1(end);
n_x2        = length(x)-n_x1;

% Find wrongly classified objects
[wrong, ~]  = find(classified ~= x);        % Wrongly classified objects
errRate     = 1-sum(classified == x)/n;     % Error rate

% Make confusion matrix
C(:,1) = [n_x1 - sum(wrong<=n_x1) ;        sum(wrong<n_x1) ];
C(:,2) = [       sum(wrong>n_x1)  ; n_x2 - sum(wrong>=n_x1)];

confusion = C;

end
```

B.6 plotConfusion-funksjon

Listing 13: MATLAB-funksjon for å plotte forvirringsmatriser som varmekart

```
function plotConfusion(C, meta)
%{
Plot out heatmap from confusion matrix - data constructed like this:

                     _   Data-1     Data-2      Data-3  _
  Nearest Neighbor  |    (...)      (...)       (...)    |
  Mimimum Error     |    (...)      (...)       (...)    |
  Least Squares     |_   (...)      (...)       (...)   _|

%}

n_1 = size(C,1);
n_2 = size(C,2);

fprintf('Confusion matrices of the different classificators');
set = 1;

for col=1:2:n_1
    for row=1:2:n_2
        figure
        h = heatmap(C(row:row+1,col:col+1));
        h.XLabel = 'Predicted class';
        h.YLabel = 'True class';
        h.Title = meta{row,col};
    end
    set = set + 1;
end
end
```

B.7 sortErrorRates-funksjon

Listing 14: MATLAB-funksjon for å sortere feilrater i tabell

```
function [T]= sortErrorRates(errorRates, meta)
meta(:,2:2:end) = [];
meta(2:2:end,:) = [];
T = table(...
    errorRates(:,1), meta(:,1), ...
    errorRates(:,2), meta(:,2), ...
    errorRates(:,3), meta(:,3)  ...
    );

for i=1:3
    [~, order] = sort(errorRates(:,i));
    errorRates(:,i) = sort(errorRates(:,i));
    meta(:,i) = meta(order', i);
end

T(:,1:2:end) = num2cell(errorRates);
T(:,2:2:end) = meta;
T.Properties.VariableNames = {...
    'Data_set_1', 'Classificator_1', ...
    'Data_set_2', 'Classificator_2', ...
    'Data_set_3', 'Classificator_3'  ...
    };
```

```
23
24 end
```

C Appendix C - MATLAB-script til å gjøre hele oppgaven

C.1 Script (kode) for å gjennomføre oppgaven

Listing 15: MATLAB-script som utfører hele oppgaven

```
%% init
close all
clc
clear variables

% Add paths to functions and data
addpath('data')
addpath('functions')
addpath('classificators')

%% 2 – Datasett
% Open files and extact data (ready to use with Classifcation Learner app)
[train_1, test_1, true_1] = data2Table('ds–1.txt');
[train_2, test_2, true_2] = data2Table('ds–2.txt');
[train_3, test_3, true_3] = data2Table('ds–3.txt');

%% 5 – Gjenommforing av oppgaven
% Run closest neighbor classifcator, get best property combination
[NN_1, prop_1, C_NN_1, T_1] = closestNeighbor(train_1, test_1, true_1);
[NN_2, prop_2, C_NN_2, T_2] = closestNeighbor(train_2, test_2, true_2);
[NN_3, prop_3, C_NN_3, T_3] = closestNeighbor(train_3, test_3, true_3);
err_NN_1 = T_1{1,1};
err_NN_2 = T_2{1,1};
err_NN_3 = T_3{1,1};

% Run min error classificator
[minErr_1, err_minErr_1, C_minErr_1] = minErrorClassificator(...
    train_1, test_1, prop_1, true_1);
[minErr_2, err_minErr_2, C_minErr_2] = minErrorClassificator(...
    train_2, test_2, prop_2, true_2);
[minErr_3, err_minErr_3, C_minErr_3] = minErrorClassificator(...
    train_3, test_3, prop_3, true_3);

% Run least squares method classificator
[LS_1, err_LS_1, C_LS_1] = leastSquaresMethod(...
    train_1, test_1, prop_1, true_1);
[LS_2, err_LS_2, C_LS_2] = leastSquaresMethod(...
    train_2, test_2, prop_2, true_2);
[LS_3, err_LS_3, C_LS_3] = leastSquaresMethod(...
    train_3, test_3, prop_3, true_3);

% Confusion matrices gathered and plotted
C = [...
    C_NN_1 C_NN_2 C_NN_3; ...
    C_minErr_1 C_minErr_2 C_minErr_3; ...
    C_LS_1 C_LS_2 C_LS_3 ...
    ];
meta = {...
    'Closest Neighbor (data set 1)', '',    ...
    'Closest Neighbor (data set 2)', '',    ...
```

```
    'Closest Neighbor (data set 3)', '';    ...
    '', '', '', '', '', '';                 ...
    'Minimum Error (data set 1)', '',       ...
    'Minimum Error (data set 2)', '',       ...
    'Minimum Error (data set 3)', '';       ...
    '', '', '', '', '', '';                 ...
    'Least Squares (data set 1)', '',       ...
    'Least squares (data set 2)', '',       ...
    'Least Squares (data set 3)', '';       ...
    '', '', '', '', '', ''                  ...
    };
plotConfusion(C,meta);

% Compare error rates
ERR = [...
    err_NN_1        err_NN_2        err_NN_3;       ...
    err_minErr_1    err_minErr_2    err_minErr_3;   ...
    err_LS_1        err_LS_2        err_LS_3        ...
    ];
best_classificators = sortErrorRates(ERR,meta)
```