Group 5:

Tikhon Riazantsev, Agastya Heryudhanto, Tabea Volpert,

Nadja Rothberg.

Task 2: Nearest Neighbor Classification (10 Points)

Add the following tasks to your MATLAB script *myIntroduction*.
You received the following training data labeled with the classes 0 or 1:

x_1	2	2	2	3	3	4	4	5	7	7	8	9
x_2	5	6	8	4	6	5	9	9	7	9	8	7
y	0	0	0	0	0	0	1	1	1	1	1	1

You want to classify new datapoints U using nearest neighbor classification.

a) Import the data into your matlab script and visualize it in a scatter plot. Mark the classes in different colors, name the axes and prepare a legend with 'class 0' and 'class 1'. (Hint: use *doc gscatter*)

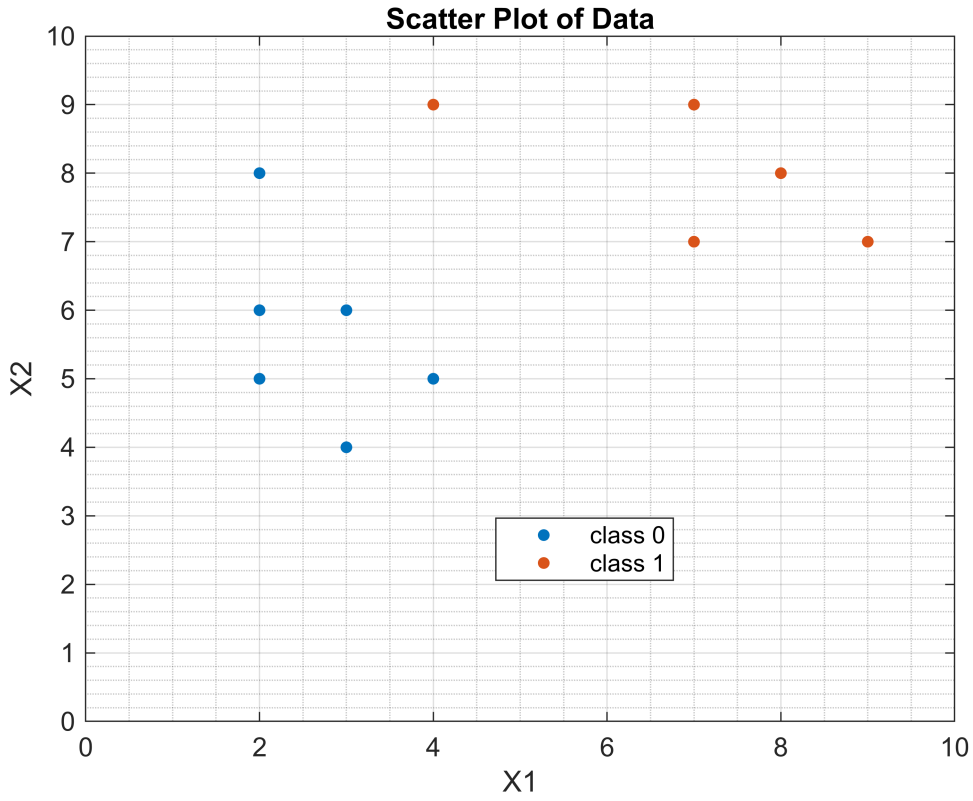
b) Create the function `[ v, pred ] = bruteForce( X, Y, U )` and implement a brute force algorithm to classify U using a for-loop. The function returns the distance vector v that contains all distances between U and dataset X and the final classification pred. Use the function to classify the datapoints $U_1 = (4, 7)$ and $U_2 = (7, 5)$.

c) Use the MATLAB function *knnsearch()* to perform a nearest neighbor classification of the datapoints U_1 and U_2 using a kd-tree.

d) What is the benefit of the kd-tree based approach in contrast to a brute-Force approach in Nearest Neighbor classification? (Hint: You can answer this question in a comment in your MATLAB script.)

a)

```
close all;
clear;
x1 = [2; 2; 2; 3; 3; 4; 4; 4; 7; 7; 8; 9];
x2 = [5; 6; 8; 4; 6; 5; 9; 9; 7; 9; 8; 7];
Y = [0; 0; 0; 0; 0; 0; 1; 1; 1; 1; 1; 1];
f1 = figure;
scatterplot = gscatter(x1,x2,Y);
xlabel('X1');
ylabel('X2');
legend('class 0', 'class 1', "Position",...
    [0.45,0.25,0.25,0.13]);
title('Scatter Plot of Data');
grid on;
grid minor;
xlim([0, 10]);
ylim([0, 10]);
```



b)

```
X = [2 2; 2 3; 2 4; 3 4; 3 5; 4 5; 4 9; 5 9; 7 7;...
    7 9; 8 8; 9 7];
U1 = [4, 7];
U2 = [7, 5];

[v1, pred1] = bruteForce(X, Y, U1);
[v2, pred2] = bruteForce(X, Y, U2);

fprintf('U1 (%d, %d), is classified as class %d\n', ...
    U1(1), U1(2), pred1);
```

```
U1 (4, 7), is classified as class 0
```

```
fprintf('U2 (%d, %d), is classified as class %d\n', ...
    U2(1), U2(2), pred2);
```

```
U2 (7, 5), is classified as class 1
```

c)

```
kdtree = createns(X, 'NSMethod', 'kdtree');

idx_U1 = knnsearch(kdtree, U1, 'K', 1);
idx_U2 = knnsearch(kdtree, U2, 'K', 1);
```

```
class_U1 = Y(idx_U1);
class_U2 = Y(idx_U2);ass_U2 = Y(idx_U2(1));

fprintf('U1 (%d, %d) is classified as class %d\n', ...
    U1(1), U1(2), class_U1);
```

```
U1 (4, 7) is classified as class 0
```

```
fprintf('U2 (%d, %d) is classified as class %d\n', ...
    U2(1), U2(2), class_U2);
```

```
U2 (7, 5) is classified as class 1
```

d)

```
time_bruteforce = timeit(@() bruteForce(X, Y, U1));
time_knnsearch = timeit(@() knnsearch(kdtree, U1, 'K', 1));
fprintf('Time to classify for bruteForce classifier: %s\n', ...
    time_bruteforce);
```

```
Time to classify for bruteForce classifier: 6.467100e-06
```

```
fprintf('Time to classify for knnsearch classifier: %s\n', ...
    time_knnsearch);
```

```
Time to classify for knnsearch classifier: 3.701710e-05
```

```
fprintf(['As we can see, knnsearch is slower in classifying small' ...
    'datasets, but much faster in larger ones.\n' ...
    'If we are classifying data with n dimensions and m samples' ...
    'Time to compute of knnsearch is O(nlog(m))\n' ...
    'Time to compute bruteForce is O(nm)'])
```

```
As we can see, knnsearch is slower in classifying smalldatasets, but much faster in larger ones.
If we are classifying data with n dimensions and m samplesTime to compute of knnsearch is O(nlog(m))
Time to compute bruteForce is O(nm)
```