Assignment 3

Computational Intelligence, SS2023

Team Members		
Last name	First name	Matriculation Number
Carlos Franco	Verde Arteaga	
Nedžma	Mušović	
Laura Maria	Höber	

Contents

2 k-Nearest Neighbors **2**
2.1 kNN for Classification . . . 2
2.1.1 Decision Boundaries of training data . . . 2
2.1.2 Score and decision boundary . . . 5
2.1.3 Score against K . . . 7
2.2 kNN for Regression . . . 8

3 Support Vector Machines **9**
3.1 Linear SVMs . . . 9
3.1.1 Linear decision boundary . . . 10
3.1.2 Support vectors . . . 11
3.1.3 Objective function . . . 11
3.1.4 Decision boundary with modified data . . . 12
3.2 Nonlinear SVMs . . . 14
3.3 Kernels . . . 15

2 k-Nearest Neighbors

2.1 kNN for Classification

In this exercise, we will plot the decision boundary that result from kNN-classification based on the training data, considering different numbers of neighbors K = 1,2,3,4,5.
The data consists of features vectors x_n, each containing 2 features, and their class labels, which can be either 0, 1 or 2. Overall there are 200 data points, who are separated into 180 training samples for training the classifier and 80 test samples to test its performance.

2.1.1 Decision Boundaries of training data

The classification of a new point based on its closest neighbors consists of the steps:

- Calculating the distance to every other point in the training data via euclidean distance
- Chose the k closest points
- Make a majority vote to find the class label, that occurs most often under these k points, and assign this label to the new point (If two or more classes occur the same amount of times, make a random choice between them)

The following graphs 1 to 5 show the resulting decision boundaries, obtained by classifying every point in the shown background based on the training data. Here a step size of 0.02 is used.

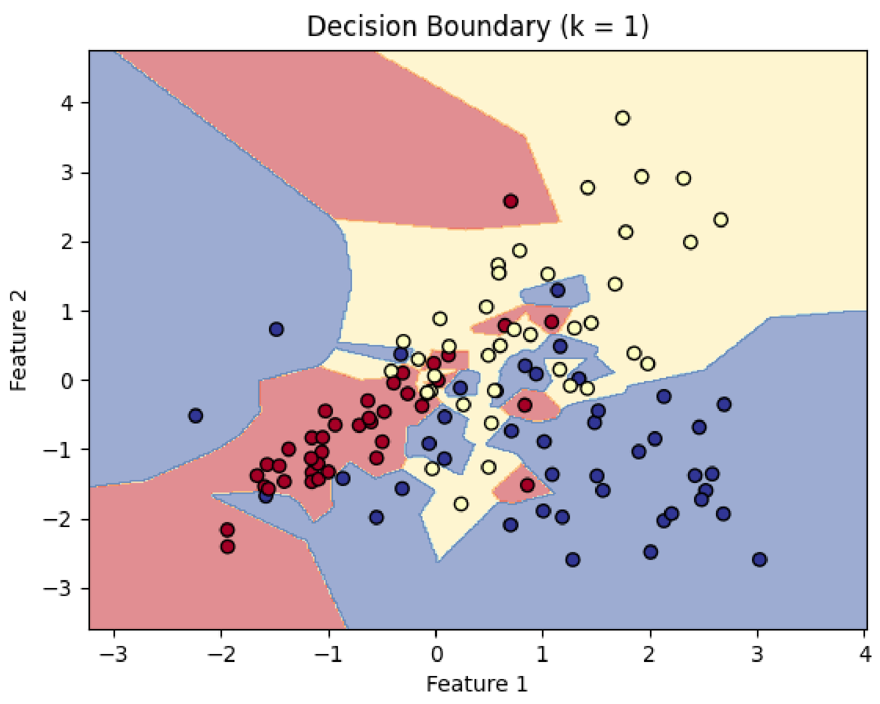

Figure 1: Decision Boundary using training data K=1

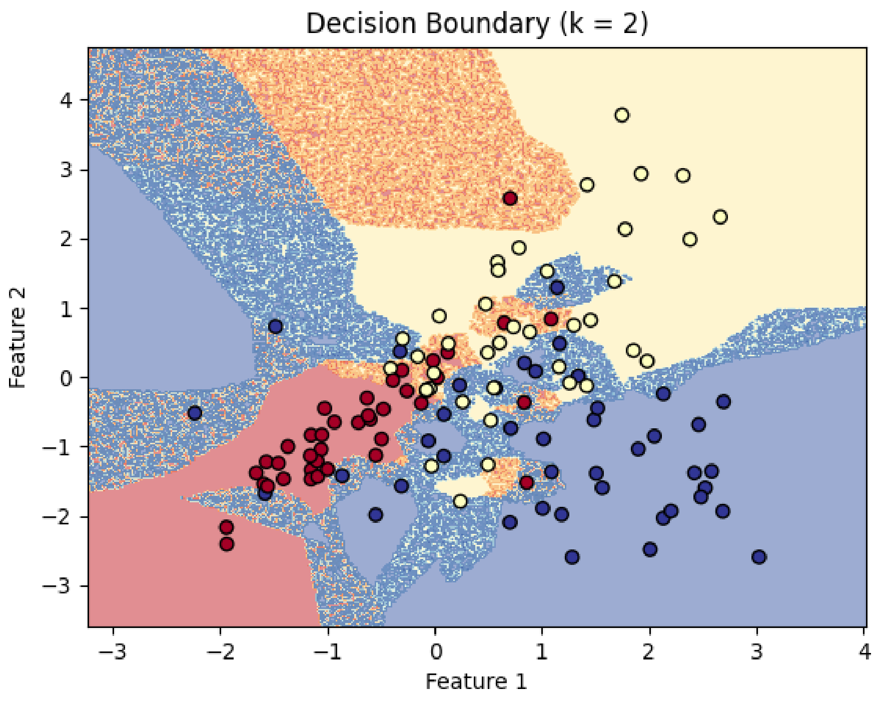

Figure 2: Decision Boundary using training data K=2

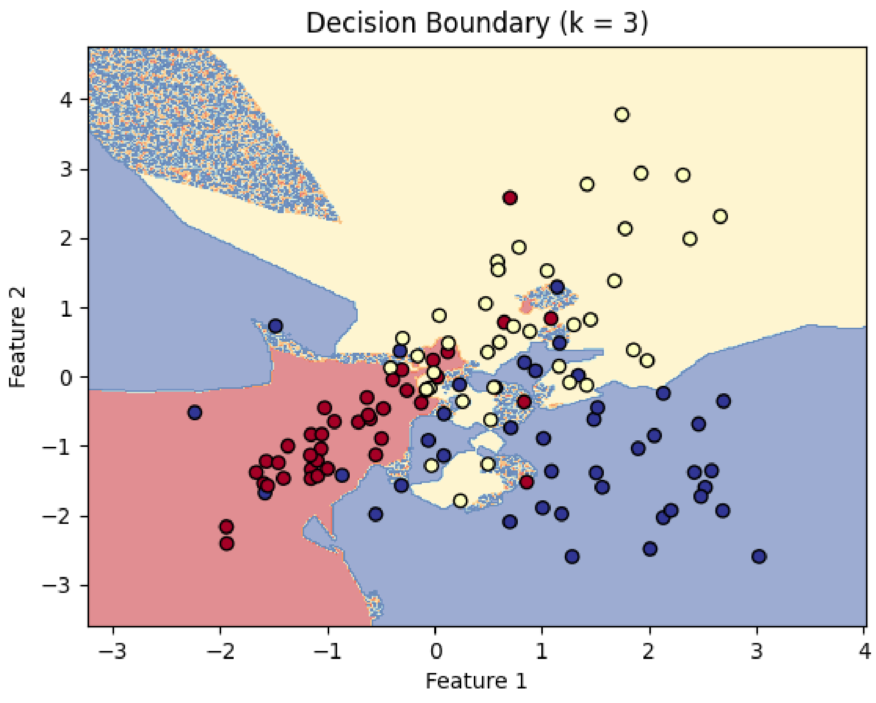

Figure 3: Decision Boundary using training data K=3

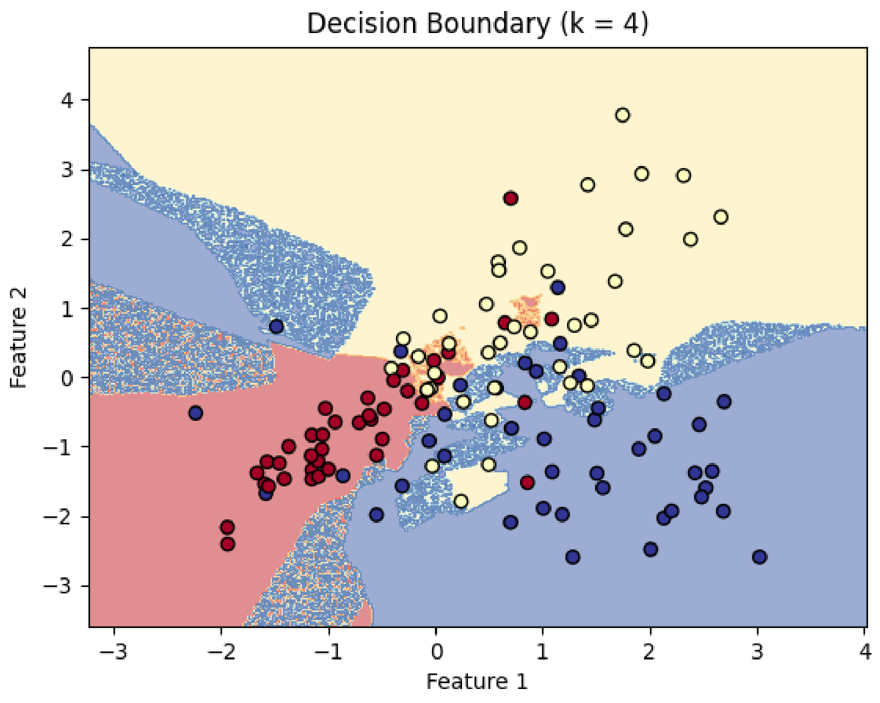

Figure 4: Decision Boundary using training data K=4

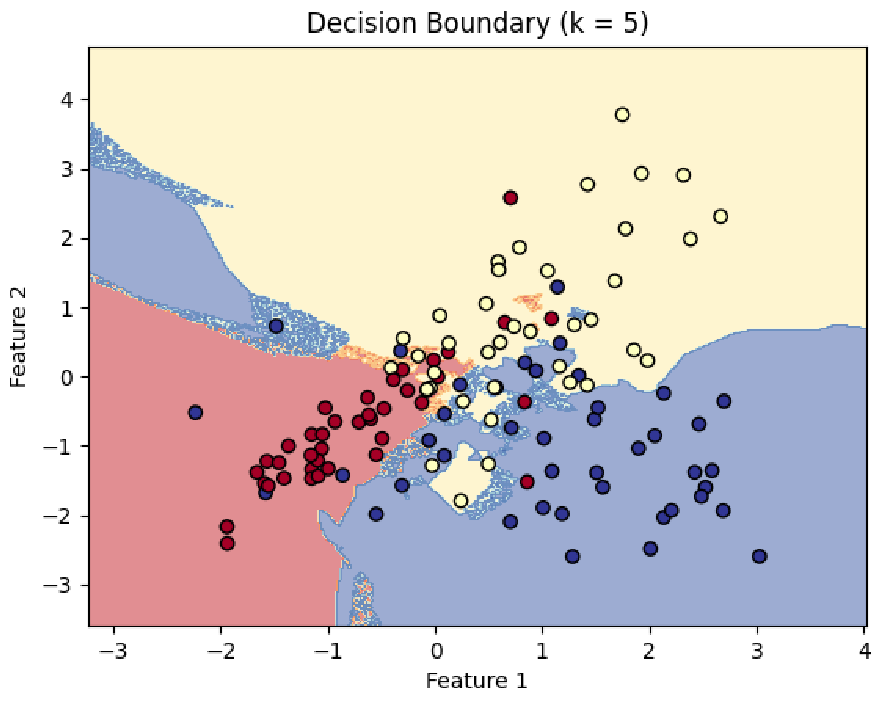

Figure 5: Decision Boundary using training data K=5

Conclusion:

The graphs above showcase, that with the lowest number neighbors K1 all the data points of the training set get classified correctly, which is of course because the point itself is its own nearest neighbor. With rising number of considered neighbors there occur more misclassifications. However, the areas begin to more resemble the general shape of the data instead of fitting every single point and little areas of outliers are ignored.

What also needs to be mentioned are the noisy patches for K>1. These can be explained with the random choice that is made in case of a tie between neighboring classes. The patches also get smaller with a higher k, because it gets less likely to reach a tie if more points are considered.

2.1.2 Score and decision boundary

In the next exercise we evaluate the decision boundaries, that resulted form the kNN-classification previously shown with the training data. Figures 6 to 7 include the training points in full color, the test points in transparent color and the reached accuracy score, which is the percentage of correctly classified test data.

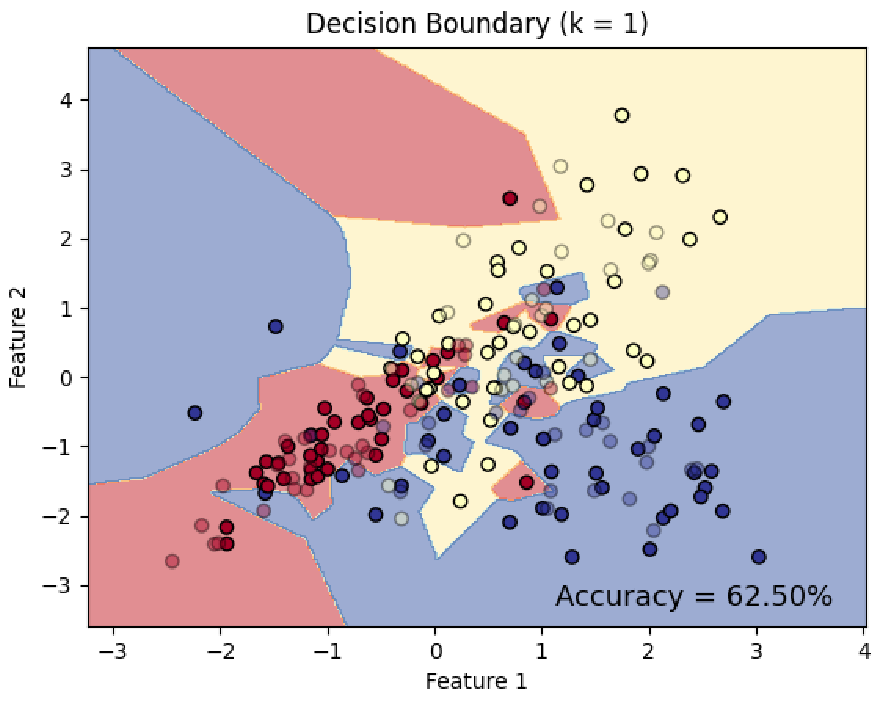

Figure 6: Decision Boundary using training data K=1

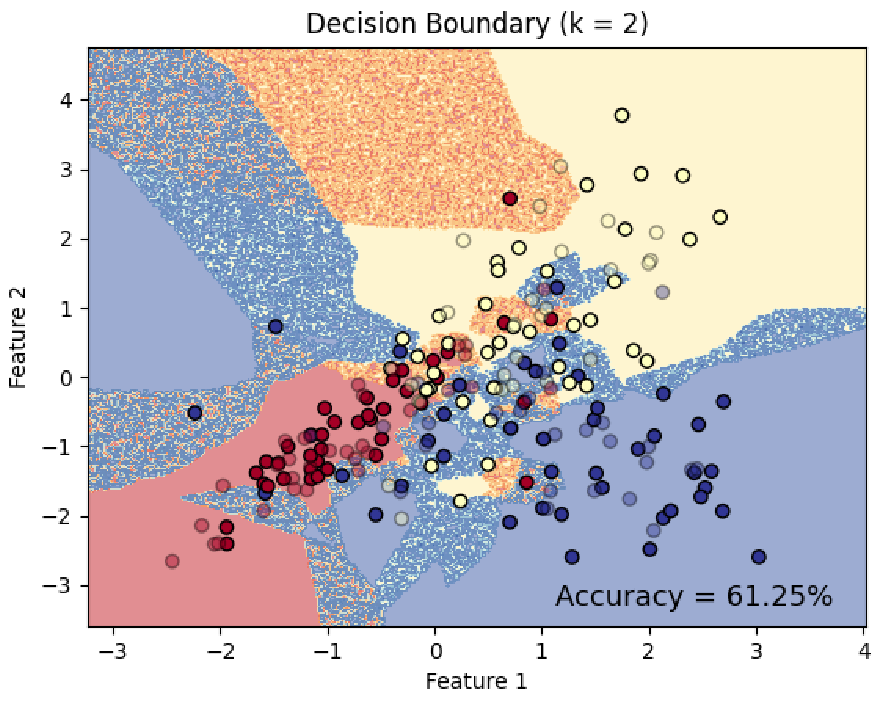

Figure 7: Decision Boundary using training data K=2

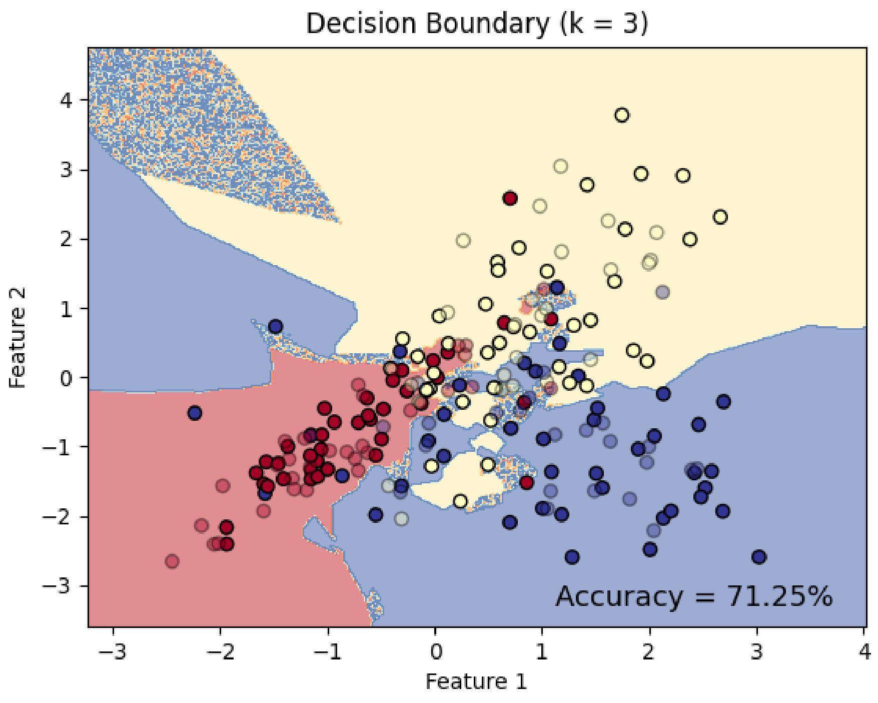

Figure 8: Decision Boundary using training data K=3

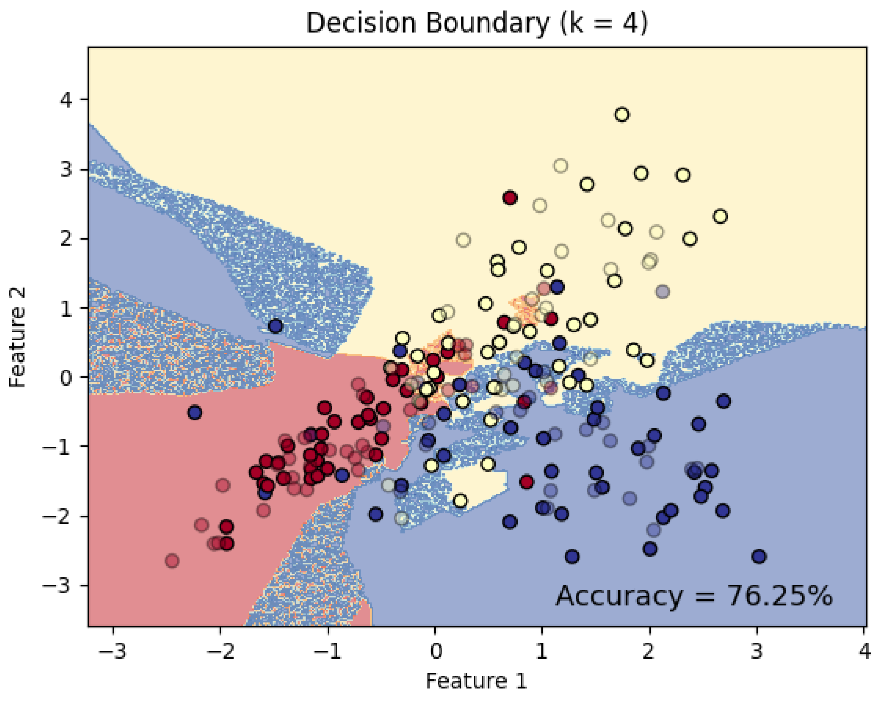

Figure 9: Decision Boundary using training data K=4

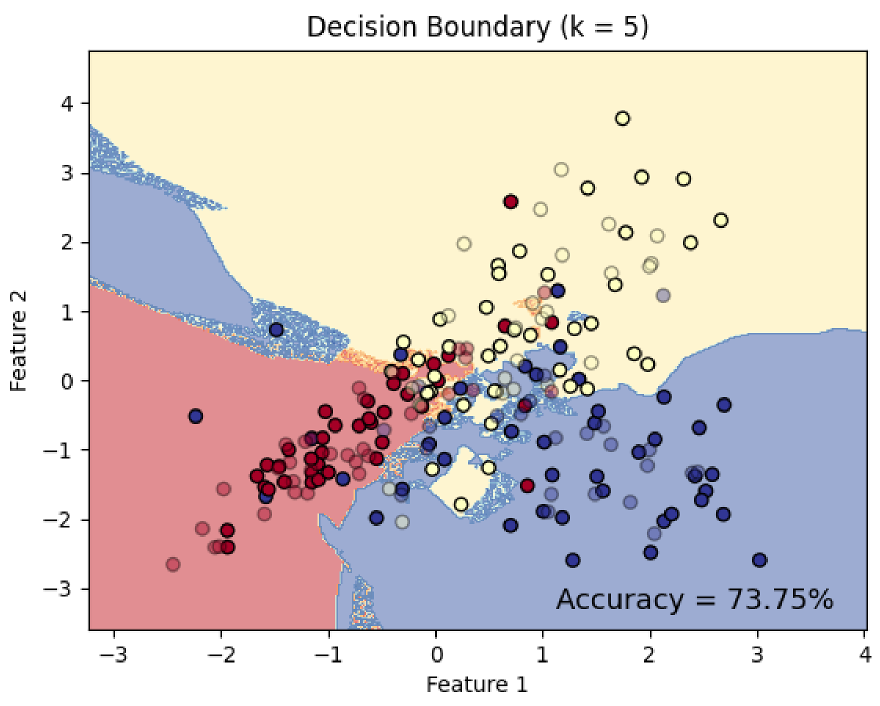

Figure 10: Decision Boundary using training data K=5

Table 1: Score for different K

K	1	2	3	4	5
Accuracy Score	0.625	0.6125	0.7125	0.7625	0.7375

Conclusion:
Analyzing the summarized table that we made, the option for the most accurate classification is a k of 4 with a score of 76.25 percent. We can also notice that the worst score is when we apply k = 2 with 61.25 percent.

2.1.3 Score against K

As a last subtask, we plotted the score against k = 1,2,...,20 to see which number of neighbors leads to the best accuracy.

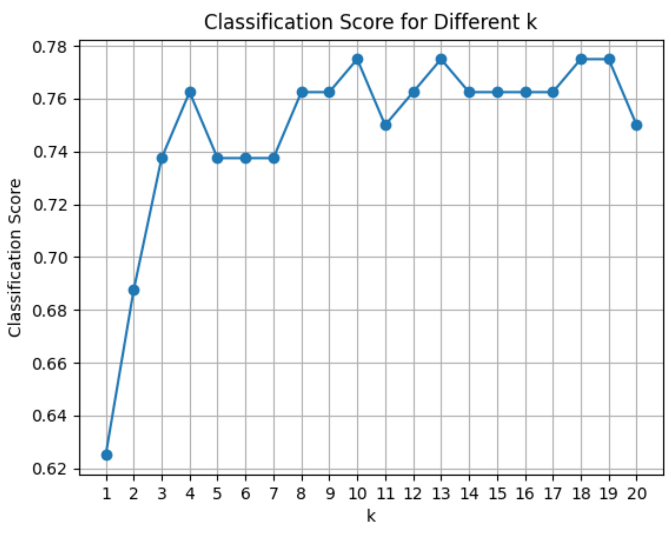

Figure 11: Score against K using test data

Conclusion:
In our graph we can see, that for k = 10 we reach the maximum score of 77.5 percent for the classification of test points. The accurracies of all subsequent numbers of k are either equal(for k = 13, 18, 19) or worse. Therefore, the previously found peak accuracy of 76.25 percent for k = 4 still seems to be the best choice, considering that the accuracy afterwards stagnates at around 76 percent, while still needing more processing power.

2.2 kNN for Regression

In this subtask, we apply kNN for regression on a dataset establishing a relation between the blood pressure and diabetes progression quantifier. The classical kNN algorithm (for two neighbors) was applied, whereby the average value of the nearest neighbors from the trained set was selected as the prediction for the new data point.
Plotting the regression function versus the trained data points resulted as follows:

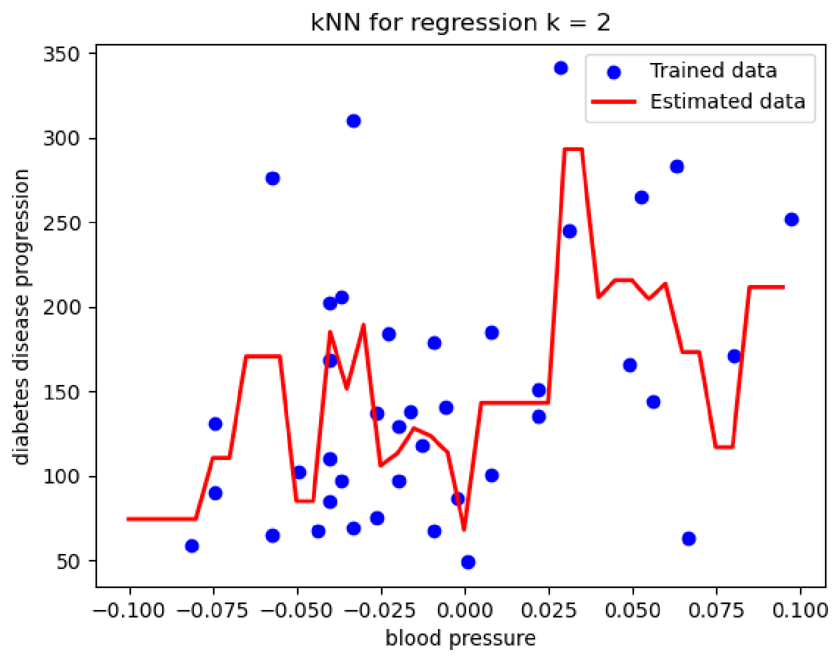

Figure 12: kNN regression for diabetes dataset

As observable from the visualization of the regression function, it fits the data only partially well. On the one side, it mostly follows the given trend, but it indubitably does not entirely mirror it. However, it needs to be taken into account that the kNN is a local method, estimating targets based on the nearest neighbors and that a variant and outliers prone data set is the object of this observation. Nevertheless, selecting k = 2 seems to offer a fairly high variance capturing individual points but not being too sensitive against outliers. For a smoother fit, the number of neighbors needs to be increased.

3 Support Vector Machines

In this task support vector machines (SVMs) are used to fit linear or polynomial decision boundaries between two classes. The considered data consists of the N feature vectors $\mathbf{x}_n$ with n = 1...N (each containing two feature values) and N corresponding targets $\mathbf{t}_n$.
The aim of the SVM algorithm is to find the parameters α_n and b, that define a decision boundary, which has the greatest distance to points of either class. The points, that are closest to the decision boundary on either side are called "support vectors". There are two approaches to achieve find the parameters: solving either the primal problem or the dual problem. Both approaches lead to the same solution, but for this exercise the dual problem is chose, which looks as follows:

$$\begin{gathered} argmax\tilde{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n \alpha_m t_n t_n k(\mathbf{x}_n, \mathbf{x}_m) \\ \text{with} \quad 0 \leq \alpha_n \leq C, \quad \forall n \quad \text{and} \quad \sum_{n=1}^{N} \alpha_n t_n = 0 \end{gathered} \tag{1}$$

The regularization parameter C is chosen as 1. This method has the advantage, that it uses a kernel function k($\mathbf{x}_n$,$\mathbf{x}_m$) instead of the dot product ${\mathbf{x}_n}^T\mathbf{x}_m$ and once the problem is optimized, only the support vectors are needed to classify newly available data points.

Once the optimal parameters have been found, the classification of new samples is done with the function:

$$y(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n t_n k(\mathbf{x}, \mathbf{y}) + b \tag{2}$$

If this function outputs positive values, the predicted class is 1, and negative outputs indicate the predicted class -1. An output value of 0 would mean the point lies directly on the decision boundary.

3.1 Linear SVMs

In the first part of this exercise a linear decision boundary is searched, so the kernel function is defined as:

$$k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y} \tag{3}$$

Parameter Fitting with the simplified SMO Algorithm
To obtain the optimal parameters for the linear decision boundary, these steps have to be followed:

1.) Initialize α as a vector of zeros $(0, ...0)^T$ and b also as 0.
2.) Iterate over all values α_n with n = 1 ... N and check the so-called KKT conditions, until an α is found that violates one of them:

$$\begin{aligned} \alpha_n = 0 &\Longrightarrow t_n \cdot y(\mathbf{x}_n) \geq 1 \\ \alpha_n = C &\Longrightarrow t_n \cdot y(\mathbf{x}_n) \geq 1 \\ 0 < \alpha_n < C &\Longrightarrow t_n \cdot y(\mathbf{x}_n) = 1 \end{aligned} \tag{4}$$

3.) If α_i violates a condition, another α_j is randomly chosen with i≠j and is used to update α_i in the following way:

$$\begin{aligned} \alpha_{i,old} &= \alpha_i, \quad \alpha_{j,old} = \alpha_j \\ E_i &= y(\mathbf{x}_i) - t_i \\ E_j &= y(\mathbf{x}_j) - t_j \\ \eta &= 2k(\mathbf{x}_i, \mathbf{x}_j) - k(\mathbf{x}_i, \mathbf{x}_i) - k(\mathbf{x}_j, \mathbf{x}_j) \\ if \quad t_i \neq t_j &: L = max(0, \alpha_j - \alpha_i), H = min(C, C + \alpha_j - \alpha_k) \\ else \quad &: L = max(0, \alpha_j + \alpha_i - C), H = min(C, \alpha_j + \alpha_k) \end{aligned} \tag{5}$$

Update α_j and clipping it into the right interval:

$$\alpha_j' = \alpha_{j,old} - \frac{t_j(E_i - E_j)}{\eta}$$
$$\alpha_j = f(a,b) = \begin{cases} \text{H}, & \alpha_j' > H, \\ \text{L}, & \alpha_j' < L, \\ \alpha_j', & otherwise \end{cases} \tag{6}$$

Update α_j:

$$\alpha_i = \alpha_{i,old} + t_i \cdot t_j \cdot (\alpha_{j,old} - \alpha_j) \tag{7}$$

Update b:

$$b = f(a,b) = \begin{cases} b_1, & 0 < \alpha_i < C, \\ b_2, & 0 < \alpha_j < C, \\ \frac{b_1+b_2}{2}, & otherwise \end{cases} \tag{8}$$

4.) Repeat step 2 and 3 until the maximum number of iterations, in this case $max\_iter = 500$, is reached or until none of the weights α_i were changed during the last run.

3.1.1 Linear decision boundary

Figure 13 shows the linear decision boundary resulting from the above explained SMO algorithm. The dashed lines indicate the maximum distance from the boundary, where no data points from either class can be found. This distance is called margin.

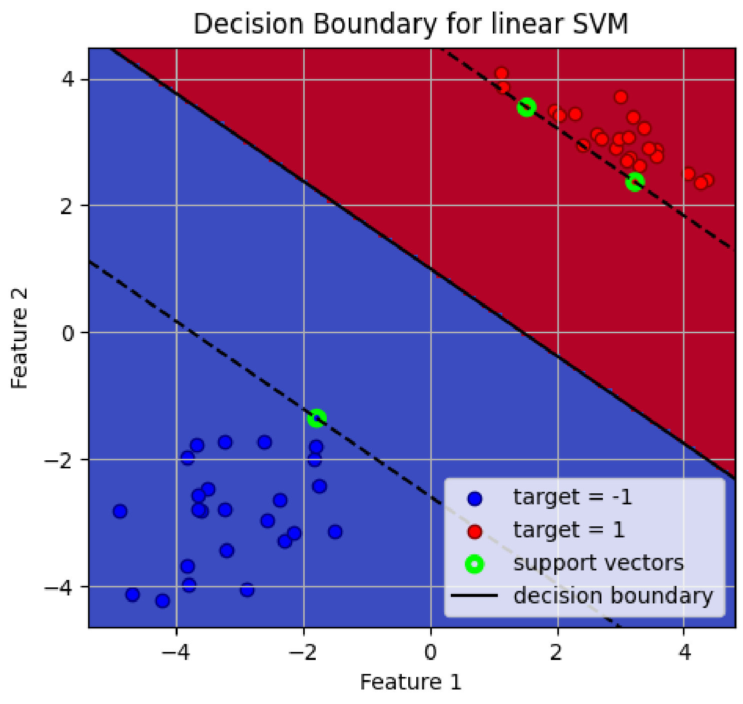

Figure 13: Decision boundary fit with the linear SVM

3.1.2 Support vectors

The points marked in green of figure 13 are the support vectors that belong to the obtained optimal boundary. A minimum of 2 support vectors, one on either side, are necessary to describe the SVM classifier. In this case a third one is coincidentally also present, because two of the points of the red class have the same distance to the boundary.
What makes these points identifiable, is that they are the only points with weights α unequal to 0. In this case, that is true for the following points:

Table 2: Support vectors and corresponding weights

i	α_i	feature 1	feature 2
3	0.0573	-1.7756	-1.3523
23	0.0015	3.2340	2.3679
26	0.0558	1.5311	3.5417

b resulted in -0.2817

These points, here named x_s together with their weights α and formula 2 can be used to obtain the analytical form of the decision boundary. The derivation of the slope k looks as such:

$$
\begin{aligned}
y(\mathbf{x}) &= \sum_{n=1}^{N} \alpha_n t_n k(\mathbf{x}, x_{s,n}) + b = 0 \\
0 &= \alpha_3 \cdot t_3 \mathbf{x}^T x_{s,3} + \alpha_{23} \cdot t_{23} \mathbf{x}^T x_{s,23} + \alpha_{26} \cdot t_{26} \mathbf{x}^T x_{s,26} + b \\
0 &= \alpha_3 \cdot t_3 \left(x^{(1)}\ x^{(2)}\right) \begin{pmatrix} x_{s,3}^{(1)} \\ x_{s,3}^{(2)} \end{pmatrix} + \alpha_{23} \cdot t_{23} \left(x^{(1)}\ x^{(2)}\right) \begin{pmatrix} x_{s,23}^{(1)} \\ x_{s,23}^{(2)} \end{pmatrix} + \alpha_{26} \cdot t_{26} \left(x^{(1)}\ x^{(2)}\right) \begin{pmatrix} x_{s,26}^{(1)} \\ x_{s,26}^{(2)} \end{pmatrix} + b \\
0 &= \left(x^{(1)}\ x^{(2)}\right) \begin{pmatrix} \alpha_3 t_3 x_{s,3}^{(1)} + \alpha_{23} t_{23} x_{s,23}^{(1)} + \alpha_{26} t_{26} x_{s,26}^{(1)} \\ \alpha_3 t_3 x_{s,3}^{(2)} + \alpha_{23} t_{23} x_{s,23}^{(2)} + \alpha_2 6 t_{26} x_{s,26}^{(2)} \end{pmatrix} + b \\
x^{(2)} &= -x^{(1)} \cdot \frac{\alpha_3 t_3 x_{s,3}^{(2)} + \alpha_{23} t_{23} x_{s,23}^{(2)} + \alpha_2 6 t_{26} x_{s,26}^{(2)}}{\alpha_3 t_3 x_{s,3}^{(1)} + \alpha_{23} t_{23} x_{s,23}^{(1)} + \alpha_{26} t_{26} x_{s,26}^{(1)}} - \frac{b}{\alpha_3 t_3 x_{s,3}^{(1)} + \alpha_{23} t_{23} x_{s,23}^{(1)} + \alpha_{26} t_{26} x_{s,26}^{(1)}} \\
x^{(2)} &= -x^{(1)} \cdot k + c
\end{aligned} \tag{9}
$$

Offset: $c = \frac{-b}{\alpha_3 t_3 x_{s,3}^{(1)} + \alpha_{23} t_{23} x_{s,23}^{(1)} + \alpha_{26} t_{26} x_{s,26}^{(1)}} = 1.0104$

Slope: $k = \frac{\alpha_3 t_3 x_{s,3}^{(2)} + \alpha_{23} t_{23} x_{s,23}^{(2)} + \alpha_{26} t_{26} x_{s,26}^{(2)}}{\alpha_3 t_3 x_{s,3}^{(1)} + \alpha_{23} t_{23} x_{s,23}^{(1)} + \alpha_{26} t_{26} x_{s,26}^{(1)}} = -0.6893$

3.1.3 Objective function

Figure 14 shows the values, that were obtained for the objective function for every iteration of the SMO algorithm. At first it oscillates between positive and negative values, but eventually converges to 0 relatively quick, after about 50 iterations. For every execution of the code this looks a little bit different, because α_j is randomly chosen, but the end result stays the same.

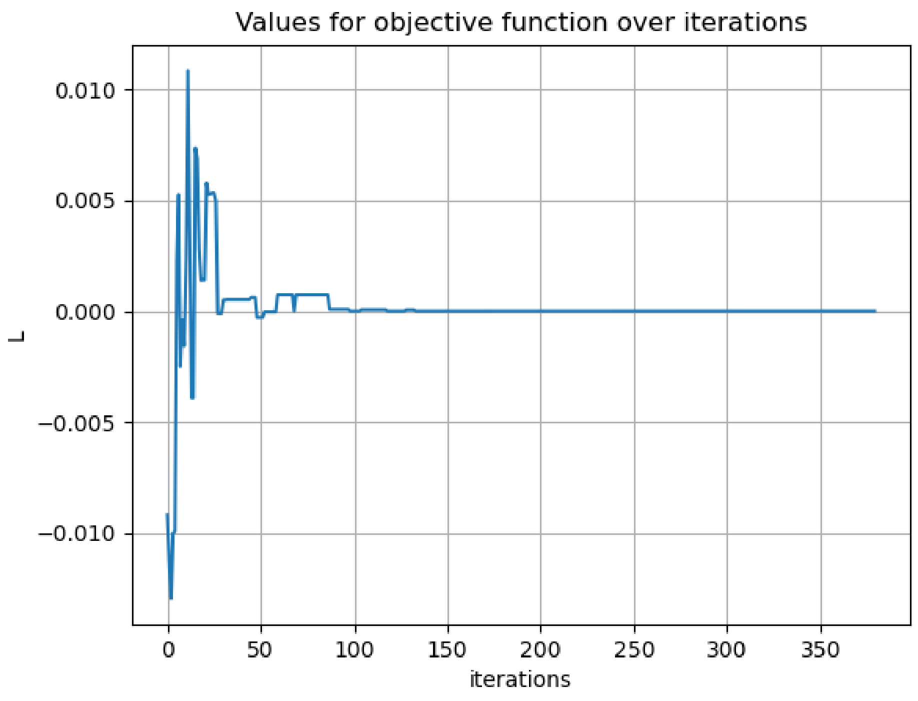

Figure 14: Values for objective function over iterations

3.1.4 Decision boundary with modified data

Now only one point is added to the data to see how it influences the resulting SVM decision boundary. The support vectors can be seen in the following table:

Table 3: Support vectors and corresponding weights

index	α	feature 1	feature 2
16	0.376	1.1417	3.8575
51	0.376	-1	3

b resulted in -1.1619

Figure 15 shows the severe impact of the new point on the decision boundary, even though it deviates from the remaining points of its class. This highlights, that the classification with SVMs is highly sensitive to outliers.

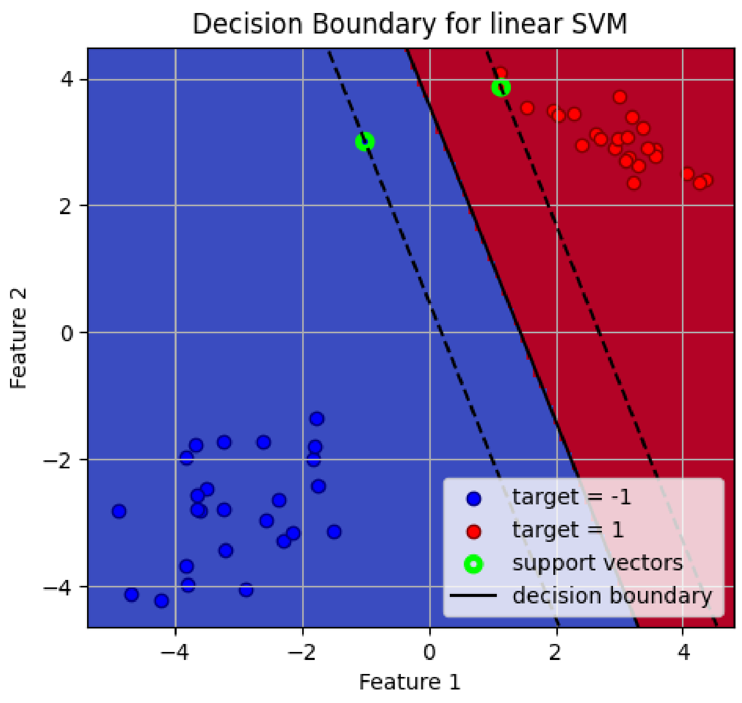

Figure 15: Decision boundary for modified data

One solution to make the classifier less sensitive to a few outliers are so-called "soft margin SVMs", which include slack variables and a penalty term weighted by C. The lower the value of C, the more outliers are allowed. It wants to find the optimal boundary, that minimizes the number of outliers and at the same time has the biggest margin to the remaining points. In our case, C can be adapted to change the constraint on the weights α_i, as seen in equation . For example, a value of C = 0.1 results in the decision boundary of figure 16 and seems to solve the problem.

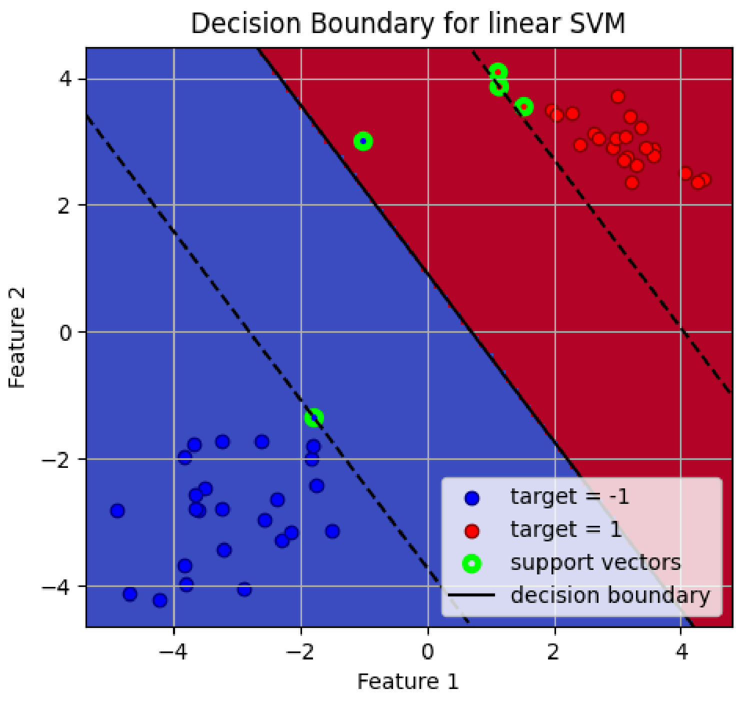

Figure 16: Decision boundary for modified data with C = 0.1

3.2 Nonlinear SVMs

The aim of this subtask was to fit the decision boundary for not linearly separable data, which was achieved by slightly adjusting the previously implemented SMO such that it involves the polynomial kernel from equation 10 instead of the linear one from equation 3.

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^3 \tag{10}$$

The decision boundary on the data set is given in the following plot:

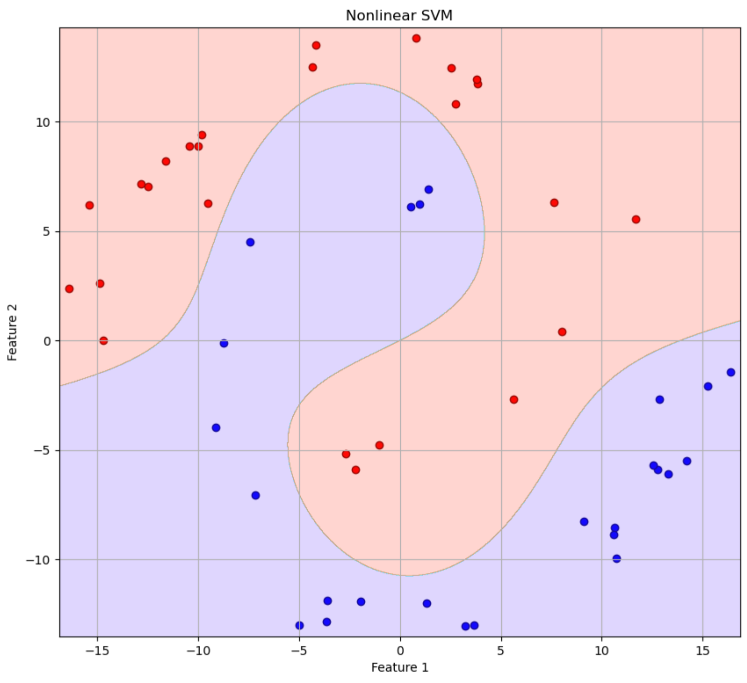

Figure 17: Decision boundary fit with the nonlinear SVM

Figure 18 shows the values, that were obtained for the objective function for every iteration of the SMO algorithm. It very slowly converges towards 0.006 (and expectable to 0 at some point) after 10000 iterations, explicitly indicating higher complexity of the nonlinear data set and the internally used polynomial kernel.

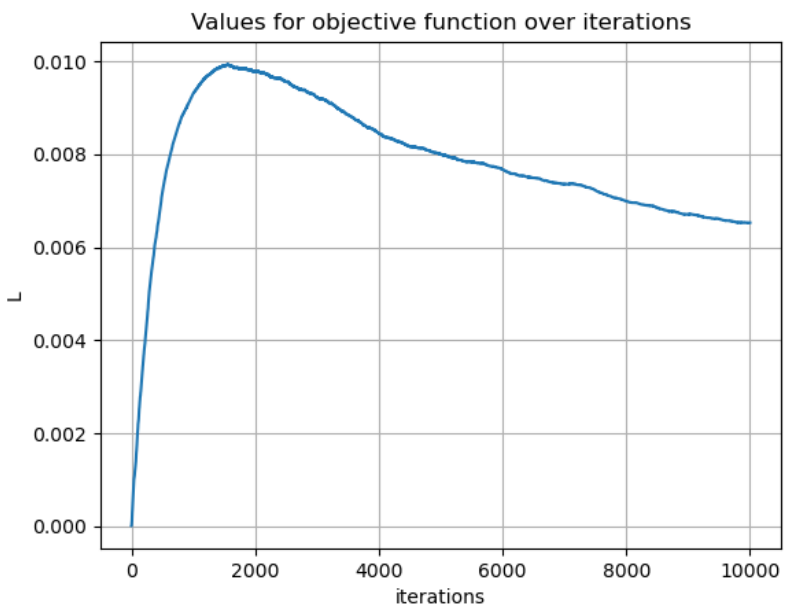

Figure 18: Values for objective function over iterations for nonlinear data

3.3 Kernels

Since a kernel is considered valid if it can be written as a dot product between two feature mappings on the input data, the proofs for the validity of the four kernels rely on this characteristic and are given as follows.

1. $\boldsymbol{k_1(x, y) = c \cdot k_A(x, y)}$

Using the previously mentioned characteristic of a valid kernel and having in mind that the $k_A(x, y)$ is a valid kernel and c a constant greater 0, $k1(x, y)$ can be evaluated as follows:

$$c \cdot k_A(x, y) = c \cdot \Phi_A(x)^T \Phi_A(y) = \sqrt{c} \cdot \Phi_A(x)^T \cdot \sqrt{c} \cdot \Phi_A(y) = (\sqrt{(c)} \cdot \Phi_A(x))^T \cdot (\sqrt{c} \cdot \Phi_A(y)) = \Phi_1(x)^T \Phi_1(y)$$

2. $\boldsymbol{k_2(x, y) = f(x) \cdot k_A(x, y) \cdot f(y)}$

Considering $k_A(x, y)$ a valid kernel and $f(\cdot)$ a real-valued function (can be transposed without any influence), the k_2 kernel expression expands as follows:

$$f(x) \cdot k_A(x, y) \cdot f(y) = f(x) \cdot \Phi_A(x)^T \Phi_A(y) \cdot f(y) = (f(x) \cdot \Phi_A(x))^T \cdot (f(y) \cdot \Phi_A(y)) = \Phi_2(x)^T \Phi_2(y)$$

3. $\boldsymbol{k_3(x, y) = k_A(x, y) + k_B(x, y)}$

If $k_A(x, y)$ and $k_B(x, y)$ are valid kernels, $k_3(x, y)$ can be rewritten again showing the expected dot form for a valid kernel.

$$k_A(x, y) + k_B(x, y) = \Phi_A(x)^T \Phi_A(y) + \Phi_B(x)^T \Phi_B(y) = \begin{bmatrix} \Phi_A(x) \\ \Phi_B(x) \end{bmatrix}^T \cdot \begin{bmatrix} \Phi_A(y) \\ \Phi_B(y) \end{bmatrix} = \Phi_3(x)^T \Phi_3(y)$$

4. $\boldsymbol{k_4(x, y) = x^T Q y}$

Using the characteristics of a valid kernel and having in mind that the $k_A(x, y)$ is a valid kernel and the Q a symmetric (transpose corresponds to the matrix) positive definite matrix (taking the square root results in a positive full ranked matrix), the right side expression gets evaluated as follows:

$$x^T Q y = x^T \cdot \sqrt{Q} \cdot \sqrt{Q} \cdot y = x^T \cdot \sqrt{Q}^T \cdot \sqrt{Q} \cdot y = (\sqrt{Q} x)^T \cdot (\sqrt{Q} y) = \Phi_4(x)^T \Phi_4(y)$$