Problem 2.1

Softplus function

The function

$$\max(0, t) \tag{1}$$

is approximated by the so called softplus function

$$\log\Big(1 + e^t\Big). \tag{2}$$

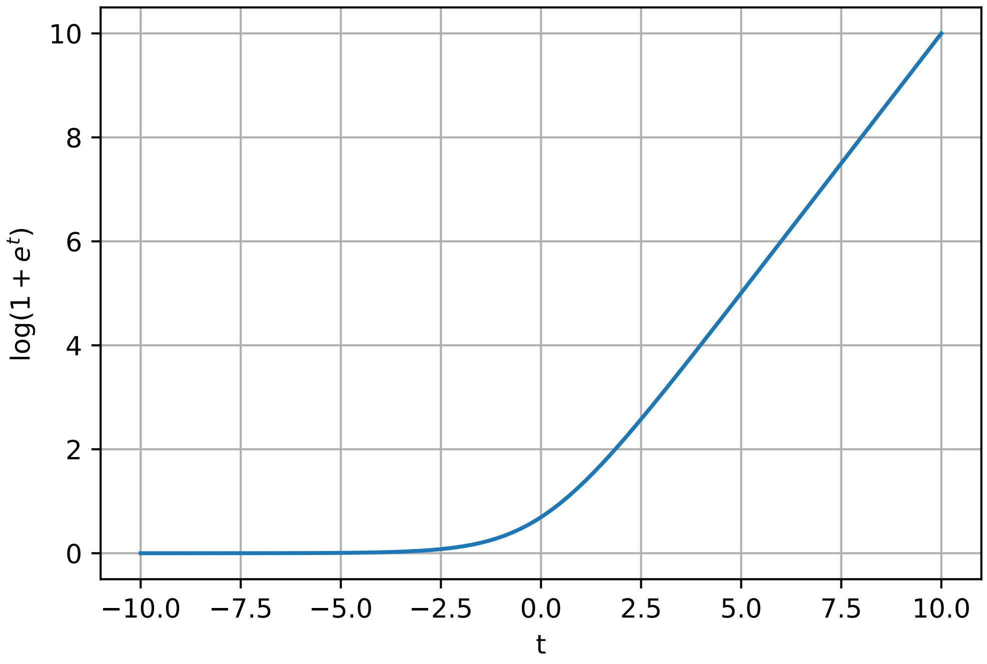

Figure 1: Softplus function equation (2)

For $t > 0$ it approaches t and for $t < 0$ it approaches 0. Therefore the function saturates for t < 0. As $t = -y_{(i)}\tilde{y}_{(i)}$ in the loss function, it saturates for $y_{(i)}\tilde{y}_{(i)} > 0$. Therefore, whenever the predicted soft label and the true label have the same sign $\text{sign}(y_{(i)}) = \text{sign}(\tilde{y}_{(i)})$, the loss function saturates and only has little effect on the overall loss.

Analytic gradient

The loss function is given by

$$J(\mathbf{w}, b) = \frac{1}{N}\sum_{i=1}^{N} \log\left(1 + e^{-y_{(i)}\left(\mathbf{x}_{(i)}^T\mathbf{w} + b\right)}\right). \tag{3}$$

The gradient with respect to the weights is

$$\nabla_{\mathbf{w}} J(\mathbf{w}, b) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{1 + e^{-y_{(i)}\left(\mathbf{x}_{(i)}^T \mathbf{w} + b\right)}} \nabla_{\mathbf{w}} e^{-y_{(i)}\left(\mathbf{x}_{(i)}^T \mathbf{w} + b\right)} = \tag{4}$$

$$\frac{1}{N} \sum_{i=1}^{N} \frac{e^{-y_{(i)}\left(\mathbf{x}_{(i)}^T \mathbf{w} + b\right)}}{1 + e^{-y_{(i)}\left(\mathbf{x}_{(i)}^T \mathbf{w} + b\right)}} \left(-y_{(i)} \mathbf{x}_{(i)}\right). \tag{5}$$

The gradient with respect to the bias term can be computed completely analogous, except that in the inner derivate there is no $\mathbf{x}_{(i)}$ term

$$\nabla_b J(\mathbf{w}, b) = \frac{1}{N} \sum_{i=1}^{N} \frac{e^{-y_{(i)}\left(\mathbf{x}_{(i)}^T \mathbf{w} + b\right)}}{1 + e^{-y_{(i)}\left(\mathbf{x}_{(i)}^T \mathbf{w} + b\right)}} \left(-y_{(i)}\right). \tag{6}$$

Gradient descent

A gradient descent algorithm has been implemented. After $K = 600$ iterations the parameters $\hat{\mathbf{w}}^{(K)} = [-2.13, 0.04]^T$ and $\hat{b}^{(K)} = 1.25$ have been found. The classifier achieves an accuracy of 100 %, i.e. it correctly classifies all samples from the training set. In Figure 2 the loss and parameters are shown during the gradient descent iteration. The loss decreases monotonically and almost approaches zero. The parameters do not reach a steady state during 600 iterations. They still change slowly towards the end.

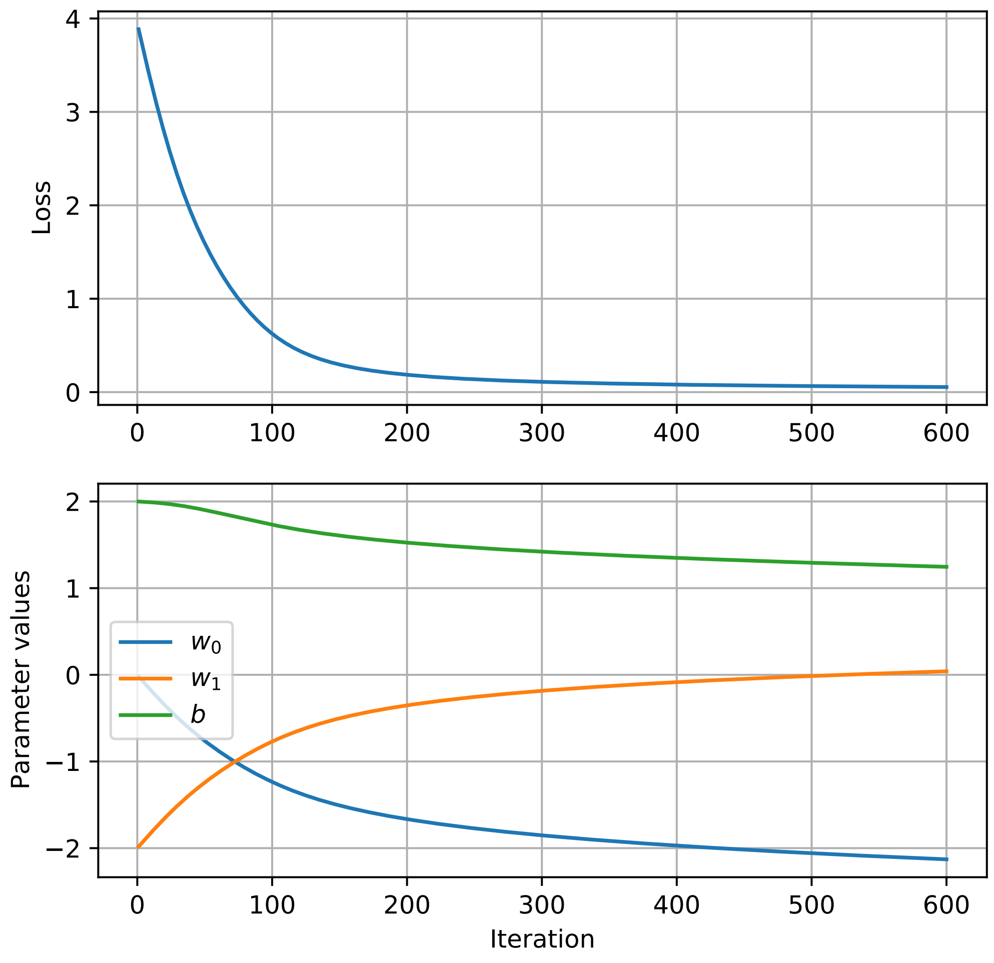

Figure 2: Loss and parameters during gradient descent

Decision surface

The decision surface is shown in Figure 3. The data can be separated by the classifier and from this plot it too can be seen that the classifier achieves an accuracy of 100 %. The decision surface is very close to the red data. As was shown in Figure 1, the loss function approaches zero for $t < 0$ and the gradient becomes very small. Therefore, as the classifier already perfectly classifies all the samples in the test set, the gradient is very small. That is why in Figure 2 the parameters change slowly towards the end.

A decision surface which maximizes the margin might be preferable in some cases due too higher robustness. This can be done by penalizing $||\mathbf{w}||$ in the loss function and therefore maximizing the margin around the decision surface (Support Vector Classification).

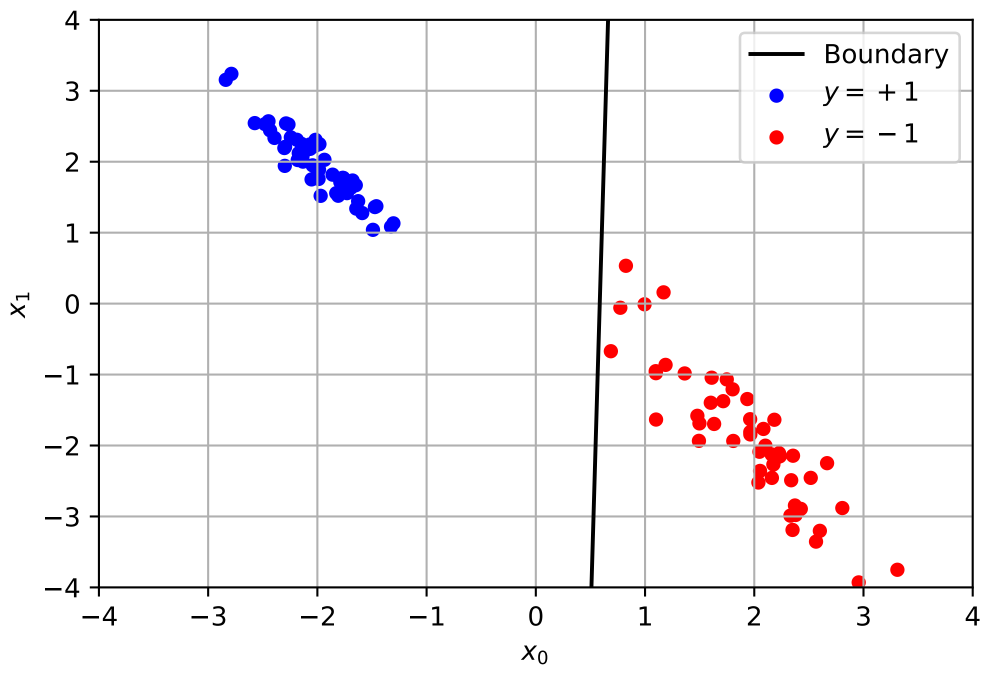

Figure 3: Decision surface and data points of the training set

Loss comparison

For the two samples $s_1 = \{[-100, 0]^T, 1\}; s_2 = \{[-1, 0]^T, -1\}$ the loss is shown in Table 1.

The sample s_1 is correctly classified, the loss of that samples is zero, this corresponds to the $t < 0$ region of the softplus function. In this region the function is approximately zero. The sample s_2 is wrongly classified, the loss is 3.41, this corresponds to the $t > 0$ region of the softplus function. In this region the loss is approximately proportional to the distance of the sample from the decision surface.

Compared to the least squares classifier, this cost function does not penalize correctly classified samples and wrongly classified samples are penalized proportionally to the distance form the decision surface. Therefore, this loss function is superior.

	Loss
s_1	0.0
s_2	3.41

Table 1: Loss of the two samples

Problem 2.2

SVC blobs dataset

In Figure 4 the dataset is shown with the decision surface. The decision surface separates the two labels and is exactly in between the two regions. The inclusion of the $||\mathbf{w}||$ term in the loss leads to a decision boundary that is maximizing the margin. The two translated version of the hyperplane pass through some of the datapoints, those are the so called support vectors. The margin is the distance between the two translated lines and can be calculated by $\frac{2}{||\mathbf{w}||}$. In this case the margin is 1.47.

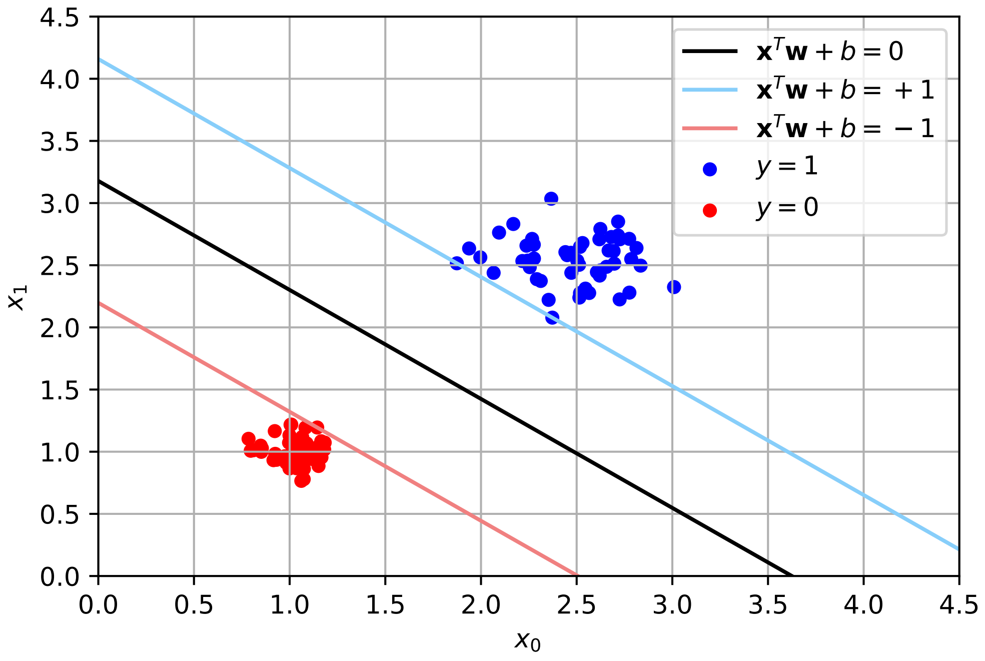

Figure 4: Blobs dataset with decision surface of SVC. The two symmetric translations are shown too

The weights and bias are shown in Table 2.

w_0	0.89
w_1	1.02
b	-3.24

Table 2: Weights and bias of the linear SVC

Linear SVC circles dataset

The circles dataset with the decision surface of the linear SVC is shown in Figure 5. The data is not linearly separable, therefore a linear SVC makes no sense and the produced result is poor.

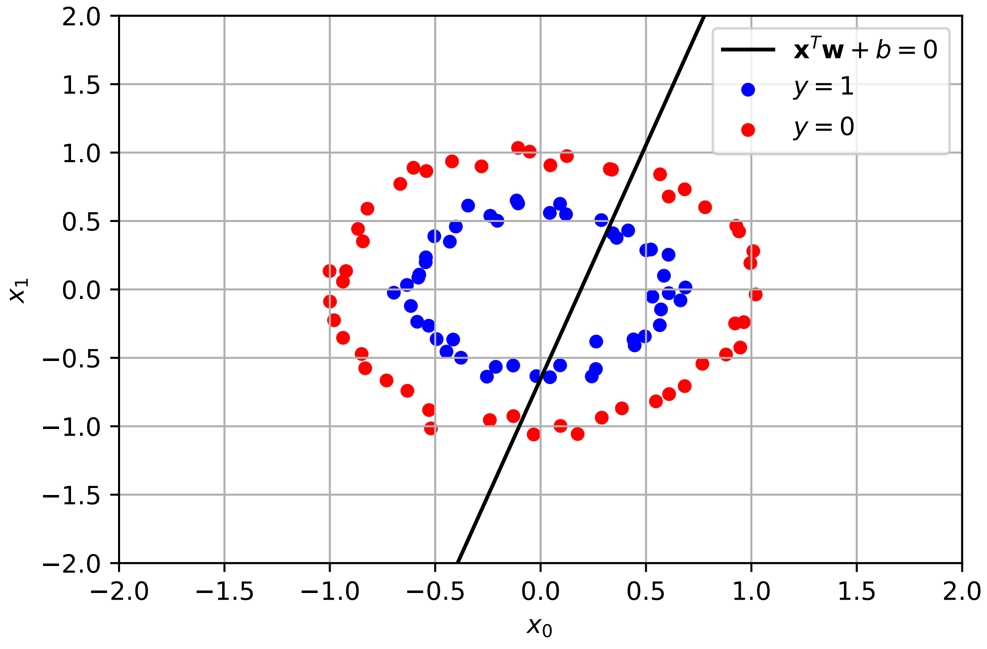

Figure 5: Circles dataset with decision surface of linear SVC. The data is not linearly separable, therefore a linear SVC makes no sense

SVC circles dataset with transformation

As can be seen in the previous Figure, the separating feature is the radius. Therefore introducing the transformation

$$\phi(x) = [x_0, x_1, x_2]^T = [x_0, x_1, x_0^2 + x_1^2]^T \tag{7}$$

makes the data linearly separable. In Figure 6 the transformed data is shown in the transformed space x_0, x_2, with $x_1 = 0$. The decision surface as well as the two translations are also shown. With this transformation the data is linearly separable, and a linear SVC can be used. The data points are projected to the plane with $x_1 = 0$ that is why some points appear to be inside the margin.

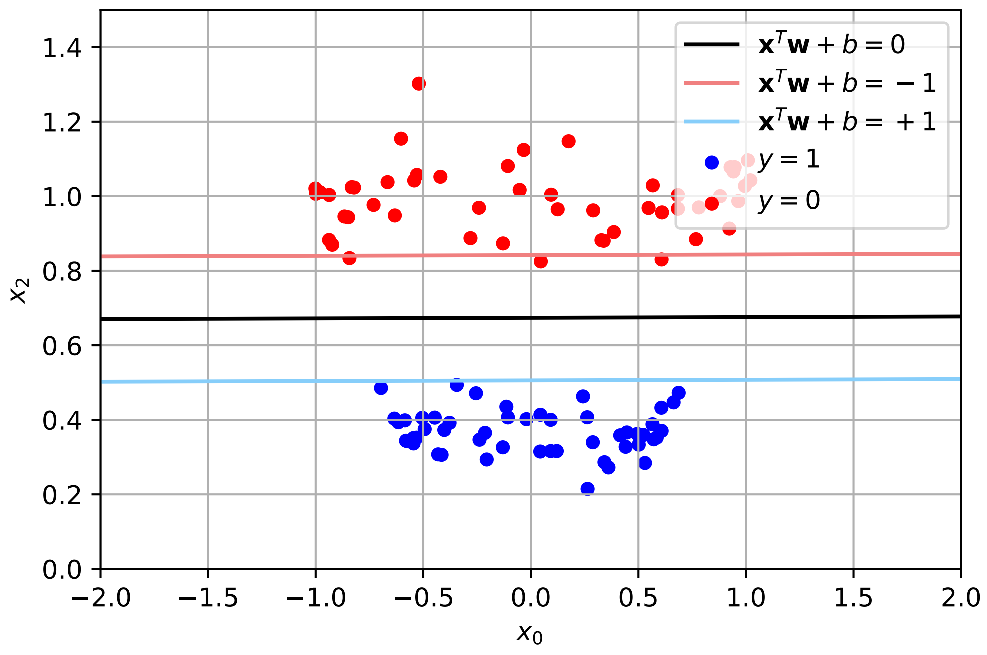

Figure 6: Circles dataset in the transformed space. In this space the data is linearly separable.

SVC with polynomial kernel on circles dataset

A polynomial kernel with degree two was used to obtain the heatmap in Figure 7. The predicted output is 1 inside a circle and 0 outside. Like the transformation used in the previous section, the polynomial kernel is also able to capture the nonlinear shape of the decision surface.

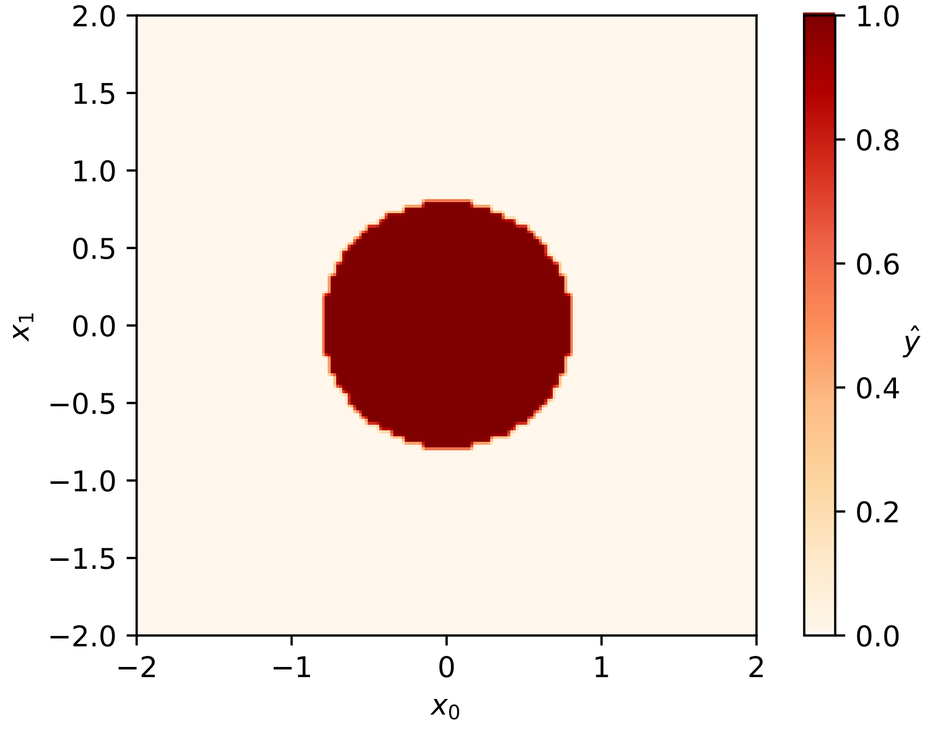

Figure 7: Heatmap of predicted labels for the SVC with polynomial kernel of degree 2

Kernel function

The nonlinear transformation is implicitly defined via ther kernel function

$$k(\mathbf{x}_{(i)}, \mathbf{x}_{(j)}) = \langle\phi\left(\mathbf{x}_{(i)}\right)\ \phi\left(\mathbf{x}_{(j)}\right)\rangle. \tag{8}$$

The polynomial kernel of degree 2 is defined as

$$k(\mathbf{x}_{(i)}, \mathbf{x}_{(j)}) = \left(\mathbf{x}_{(i)}^T\mathbf{x}_{(j)} + 1\right)^2. \tag{9}$$

For $\mathbf{x} = [x_0, x_1]^T$ the transformation an be found by expanding

$$\left(\mathbf{x}_{(i)}^T\mathbf{x}_{(j)} + 1\right)^2 = \left(\mathbf{x}_{(i)}^T\mathbf{x}_{(j)}\right)^2 + 2\mathbf{x}_{(i)}^T\mathbf{x}_{(j)} + 1 = \tag{10}$$

$$\left(x_{0,(i)}x_{0,(j)} + x_{1,(i)}x_{1,(j)}\right)^2 + 2x_{0,(i)}x_{0,(j)} + 2x_{1,(i)}x_{1,(j)} + 1 = \tag{11}$$

$$\left(x_{0,(i)}x_{0,(j)}\right)^2 + 2x_{0,(i)}x_{0,(j)}x_{1,(i)}x_{1,(j)} + \left(x_{1,(i)}x_{1,(j)}\right)^2 + 2x_{0,(i)}x_{0,(j)} + 2x_{1,(i)}x_{1,(j)} + 1. \tag{12}$$

Therefore the transformation defined by the kernel is

$$\phi(\mathbf{x}) = \left[x_0^2, \sqrt{2}x_0x_1, x_1^2, \sqrt{2}x_0, \sqrt{2}x_1, 1\right]^T. \tag{13}$$

The 2 dimensional vector is transformed to a 6-d vector ($d_2 = 6$).

Problem 1.3

Dataset histogram

A histogram of the train and test set is shown in Figure 8. For both dataset there are far more samples with $y = 0$ compared to $y = 1$. For both only 1 % of the samples have the label 1. The dataset is imbalanced.

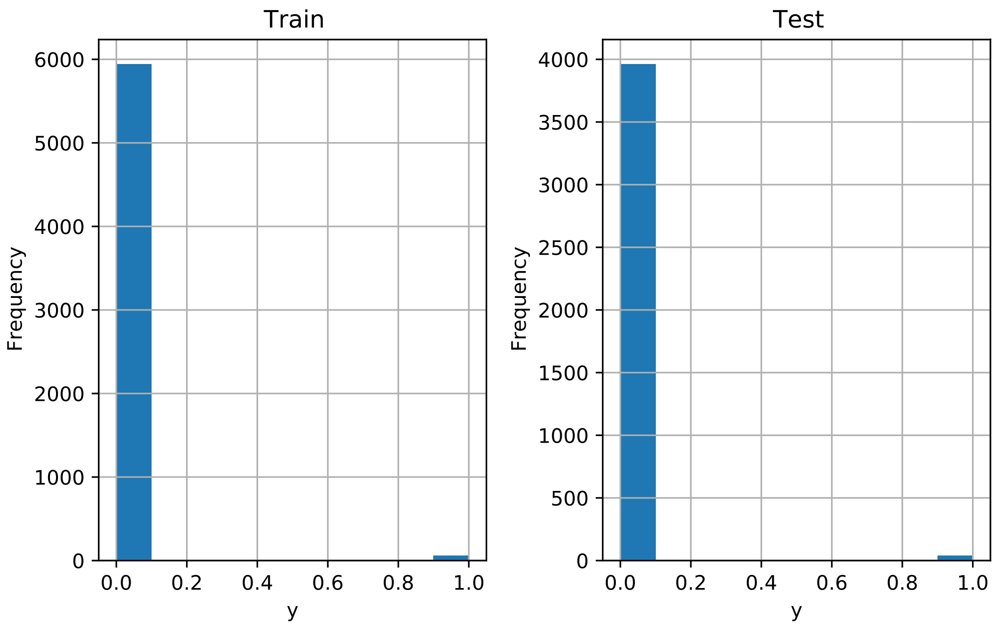

Figure 8: Histogram of train and test dataset

Original dataset

A SVC with RBF kernel was used as a classifier and was trained on the original dataset. In Table 3 the confusion matrix is shown for this classifier evaluated on the test set. In Table 4 some performance metrics are noted.

While just looking at the accuracy the performance of the classifier seems to be good with 99 %. This is just due to the fact that only 1 % of the samples have the label $y = 1$. In fact by looking at the confusion matrix it can be seen that the classifier predicts $\hat{y} = 0$ for every sample. The classifier has just learned that $y = 1$ is much more likely independent of the features. The other performance metrics cannot be evaluated.

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	3960	0
$y = 1$	40	0

Table 3: Confusion Matrix for SVC trained on original dataset

Accuracy	0.99
Precision	NaN
Recall	0.00
F1-Score	NaN

Table 4: Performance-Metrics for SVC trained on original dataset

Undersampling

Undersampling was implemented on the train dataset to train the classifier. The same SVC as in the previous task has been used. Again in Table 5 and Table 6 the results are reported.

The accuracy actually got a bit worse, but the other performance metrics improved. Precision is the fraction of true positives divided by total predicted positive. This value is low, meaning that the classifier predicts 1 too often for data that is actually negative $y = 0$. Recall is the fraction of true positives divided by total positive. This value is relatively high with 0.73 meaning that of the samples that are actually 1 most of the are correctly classified to be 1. The F1-Score is the geometric mean of both measures.

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	3491	469
$y = 1$	11	29

Table 5: Confusion Matrix for SVC trained with undersampling

Accuracy	0.88
Precision	0.06
Recall	0.73
F1-Score	0.11

Table 6: Performance-Metrics for SVC trained with with undersampling

Oversampling

The results for oversampling are reported in Table 7 and Table 8.

With this method both false positives and false negatives decreased, thus increasing precision and recall compared to undersampling.

Subsequently also the F1-Score is higher, meaning that for this dataset the highest F1-Score can be achieved with oversampling.

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	3759	201
$y = 1$	9	31

Table 7: Confusion Matrix for SVC trained with oversampling

Accuracy	0.95
Precision	0.13
Recall	0.78
F1-Score	0.23

Table 8: Performance-Metrics for SVC trained with with oversampling