Assignment 3

Machine Learning 1, SS23

Team Members		
Last name	First name	Matriculation Number
Grassl	Ifeoma	12011965
Royer	Christoph	12004184

1 k-Nearest Neighbors

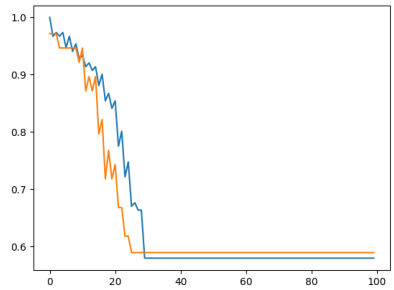
1.1 Implementation

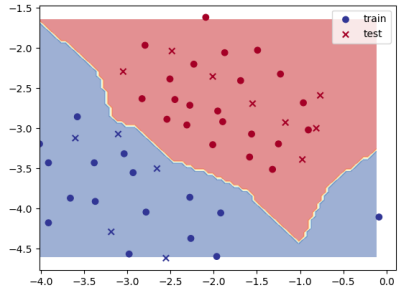
For implementation see `task1_1.py`

1.2 Application

Output for dataset 1:

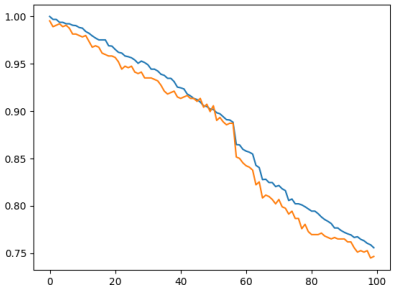
```
Test Score: 1.0
Dataset 1: {'k': 1}
```

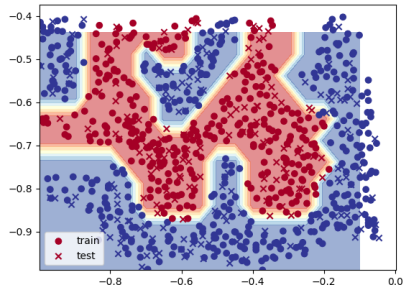




Output for dataset 2:

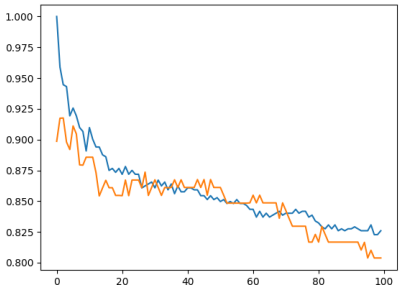
```
Test Score: 0.9953703703703703
Dataset 2: {'k': 1}
```

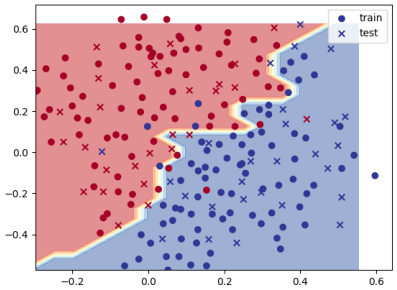




Output for dataset 3:

```
Test Score: 0.8867924528301887
Dataset 3: {'k': 3}
```



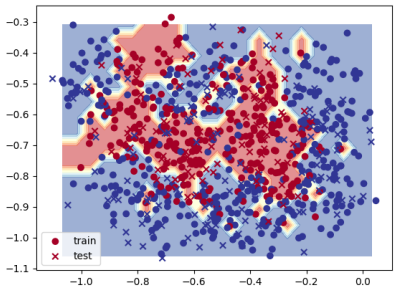


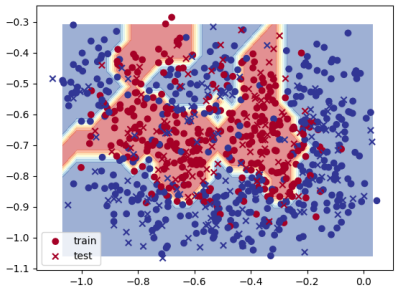
1.3 Choice of k

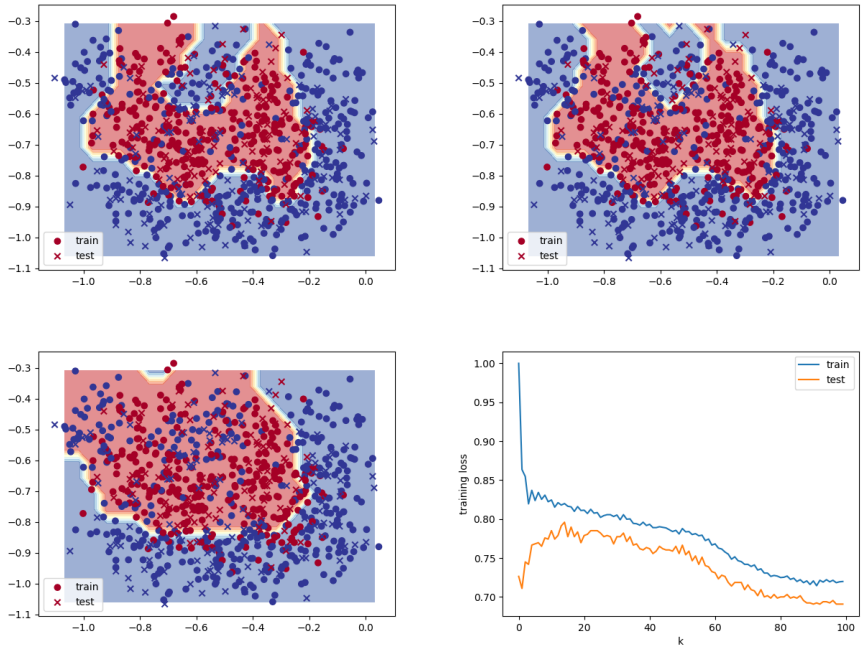
When $k = 1$, the training accuracy is always 100%, since every point is its own nearest neighbor. This will often not generalize well though, as it takes every noisy sample as a new area for the given class. A high k on the other hand will be very robust against noise, but will have problems with complex decision boundaries, especially when there are relatively few training samples present near a complex boundary.

```
Test Score for k=1: 0.7407407407407407
Cross-validated score for k=1: 0.7264162194394752
Test Score for k=5: 0.8009259259259259
Cross-validated score for k=5: 0.7666070363744782
Test Score for k=20: 0.7870370370370371
Cross-validated score for k=20: 0.7696362552176506
Test Score for k=50: 0.7824074074074074
Cross-validated score for k=50: 0.7557185450208707
Test Score for k=100: 0.7546296296296297
Cross-validated score for k=100: 0.6908288610614192

Best k: 15
Score: 0.7916666666666666
```







2 Support Vector Machines

Ass3 Task 2.1

01 June 2023 17:33

1. Consider a training data set with N input vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ with corresponding target labels $y_1, \ldots, y_N$, where $y_i \in \{-1, +1\}$. The maximum margin solution for linear support vector classifiers without slack variables is found by solving

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1} \max\left(0, 1 - y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b)\right) \tag{1}$$

where $\phi(\boldsymbol{x}) = \boldsymbol{x}$ is the identity function. This is equivalent to the formulation we have seen in the lecture

$$J(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left[y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) - 1\right],$$

where $\alpha_i \geq 0$ are Lagrange multipliers. Although $f(x) = \max(0, x)$ is a convex function, it is not differentiable. However, it is possible to express the gradient in a piece-wise fashion:

$$\frac{\partial f}{\partial x} = \begin{cases} \frac{\partial x}{\partial x}, & \text{if } x \geq 0 \\ \frac{\partial 0}{\partial x} & \text{else} \end{cases} = \begin{cases} 1, & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases}$$

Derive the gradients for $\boldsymbol{w}$ and b and fill in the missing code of the gradient descent routine. *Hint:* Don't forget to apply the chain rule of calculus!

Task: Find gradient for w and b of our SVM objective function

$$F = \underbrace{\frac{1}{2} w^T w}_{\text{regularization term}} + \underbrace{C \sum_{i=1} \max(0, 1 - y_i (w^T \phi(x_i) + b))}_{\text{hinge loss term}}$$

Derive gradient for w:

$$\frac{\partial F}{\partial w} = \frac{1}{2} \frac{\partial}{\partial w}(w^T w) + \frac{\partial}{\partial w}\left(C \sum_{i=1} \max(0, 1 - y_i (w^T \phi(x_i) + b))\right)$$

$$w^T w = w_1 \cdot w_1 + w_2 \cdot w_2 + \ldots + w_n \cdot w_n$$
$$= w_1^2 + w_2^2 + \ldots + w_n^2$$
$$\frac{\partial}{\partial w} w^T w = \left[\frac{\partial}{\partial w}(w_1^2), \ldots, \frac{\partial}{\partial w}(w_n^2)\right] = [2w_1, 2w_2, \ldots, 2w_n] = 2w$$

$$= \frac{1}{2} \cdot 2w + C \sum_{i=1} \max(0, \frac{\partial}{\partial w}(1 - y_i (w^T \underbrace{\phi(x_i)}_{= x_i} + b))$$

$$= \frac{\partial}{\partial w} 1 - \frac{\partial}{\partial w} y_i (w^T x_i) - \frac{\partial}{\partial w} y_i \cdot b$$

$$= -y_i \cdot x_i$$

$$= w + C \sum_{i=1} \max(0, -y_i \cdot x_i)$$

$$\frac{\partial F}{\partial w} = w + C \sum_{i=1} \begin{cases} -y_i \cdot x_i, & \text{if } (1 - y_i (w^T \phi(x_i) + b)) > 0 \\ 0, & \text{else} \end{cases}$$

Derive gradient for b

$$\frac{\partial F}{\partial b} = \frac{1}{2} \frac{\partial F}{\partial b}(w^T w) + \frac{\partial F}{\partial b}\left(C \sum_{i=1} \max(0, 1 - y_i (w^T \phi(x_i) + b)\right)$$

$$= C \sum_{i=1} \max(0, \frac{\partial F}{\partial b} 1 - \frac{\partial F}{\partial b} y_i (w^T \phi(x_i)) - \frac{\partial F}{\partial b} y_i \cdot b$$

$$= C \sum_{i=1} \max(0, -y_i)$$

$$\frac{\partial F}{\partial b} = C \sum_{i=1} \begin{cases} -y_i, & \text{if } (1 - y_i (w^T \phi(x_i) + b)) > 0 \\ 0, & \text{else} \end{cases}$$

2.1 Try different values for C and η

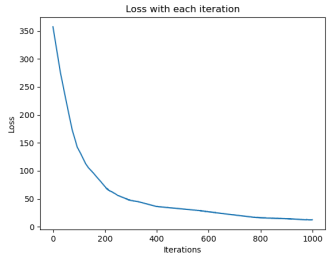
First we used grid search to get an idea of what would be the most suitable parameters for C and η.

```
parameters = {
    "max_iter": [1, 10, 100, 1000],
    "eta": [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0],
    "C": [0.001, 0.01, 0.1, 1.0, 10.0, 100.0]
}
```

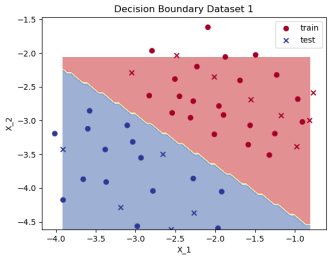
According to grid search the following parameters were the best:

```
{'C': 10.0, 'eta': 0.0001, 'max_iter': 1000}
```

The scores produce a test score of 1.0 - these are the corresponding plots.



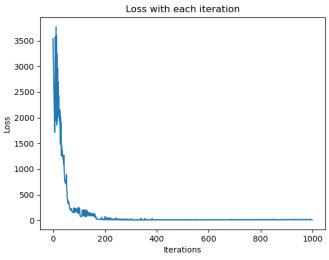
(a) Loss plot: $C = 10, \eta = 0.0001$



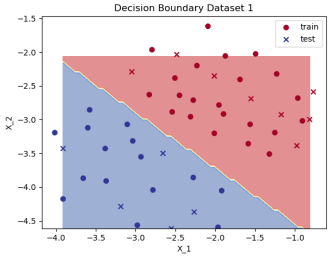
(b) Decision boundary plot

We can see here that the classifier has converged on a solution by looking at the loss plot and seeing that it gets flatter and flatter at higher max iterations, a sign that there won't be any more drastic improvements.

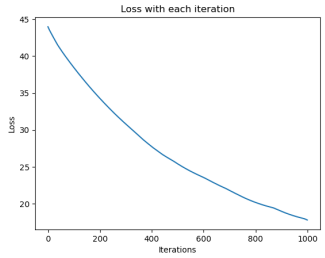
Now we can try some different values for our hyperparameter C



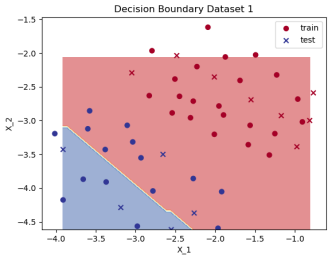
(a) Loss plot: $C = 100, \eta = 0.0001$



(b) Decision boundary plot



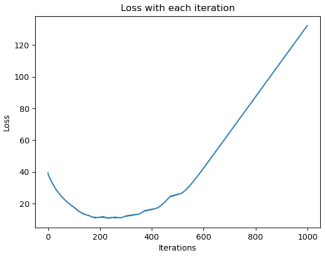
(a) Loss plot: $C = 1, \eta = 0.0001$



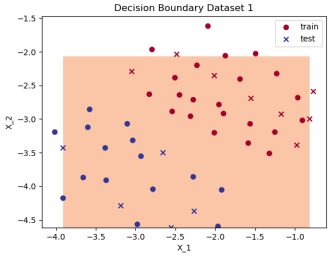
(b) Decision boundary plot

We can see increasing C leads to a steeper loss and decreasing C leads to a less steep loss plot. Here we can see the decision boundary shift. $C$is a hyperparameter that controls the trade-off between maximizing the margin and minimizing the training error. It is known as the regularization parameter (it is part of the regularization term, excuse me for grouping it together with the hinge loss term on page 4). When C is large, the impact of the regularization term becomes stronger relative to the hinge loss term. This means that the model will prioritize finding a small-margin hyperplane that correctly classifies most training examples. When C is small, the regularization term has less influence, and the model will aim to maximize the margin even if it means misclassifying a few training examples. We can see with $C = 1$ a few of the blue dots are misclassified - so you would probably have to adapt the learning rate here.

Now we can try some different values for our hyperparameter η and set $\eta = 0.01$



(a) Loss plot: $C = 1, \eta = 0.01$



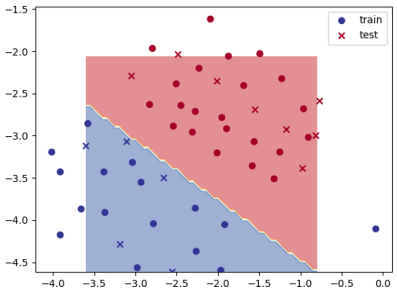
(b) Decision boundary plot

We can see this learning rate is too strong and increases the loss instead of minimizing it. The best approach would now be to utilize grid search again to find the best learning rate for $C = 1$

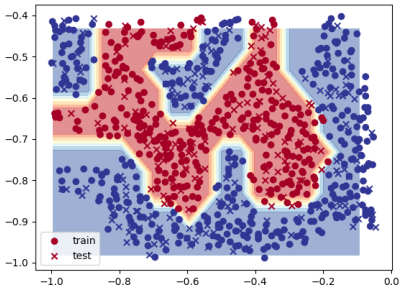
2.2 Skikit-learn built in support vector machine

These are our grid search parameters:
param_grid = ['kernel': ['linear'], 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]] else:
param_grid = ['kernel': ['linear', 'rbf'], 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000], 'gamma': [0.001, 0.01, 0.1, 1, 10, 100, 1000]]
And these are the best parameters for the three datasets

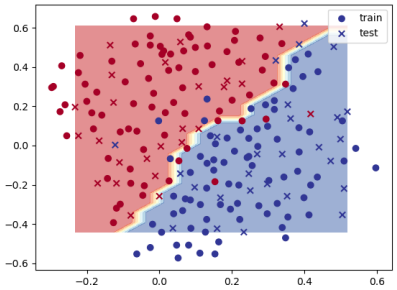
```
Test Score: 1.0
Dataset 1: {'C': 0.1, 'kernel': 'linear'}
Test Score: 1.0
Dataset 1: {'C': 0.1, 'kernel': 'linear'}
Test Score: 0.9953703703703703
Dataset 2: {'C': 1, 'gamma': 1000, 'kernel': 'rbf'}
Test Score: 0.9953703703703703
Dataset 2: {'C': 1, 'gamma': 1000, 'kernel': 'rbf'}
Test Score: 0.8867924528301887
Dataset 3: {'C': 1, 'gamma': 100, 'kernel': 'rbf'}
Test Score: 0.8867924528301887
Dataset 3: {'C': 1, 'gamma': 100, 'kernel': 'rbf'}
```



Dataset 1:



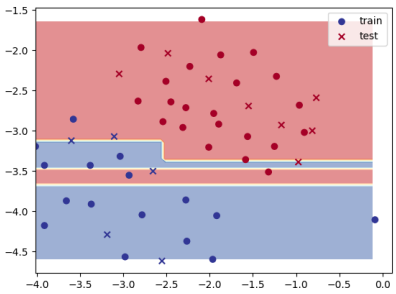
Dataset 2:

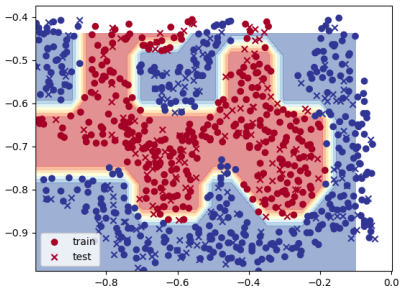


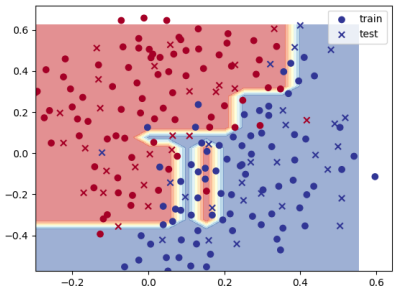
Dataset 3:

3 Decision Trees & Ensemble Methods

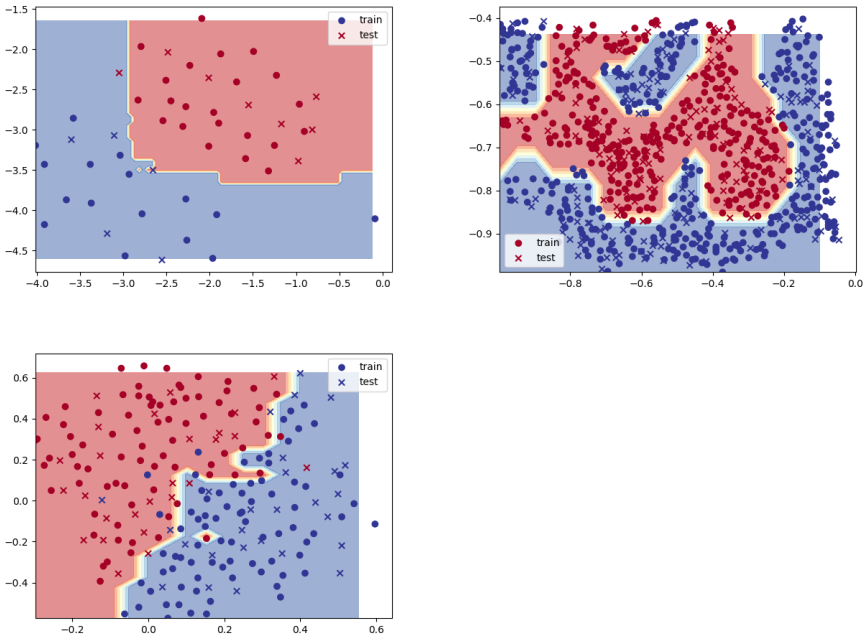
```
n_estimators = 1
Dataset 1: {'max_depth': 6}
Test Score: 0.8461538461538461
Dataset 2: {'max_depth': 19}
Test Score: 0.9444444444444444
Dataset 3: {'max_depth': 9}
Test Score: 0.8679245283018868
```







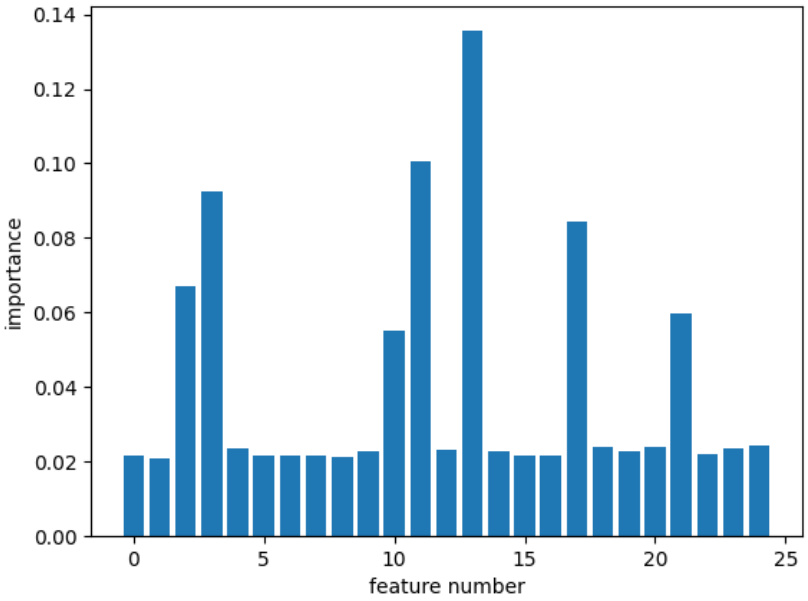
```
n_estimators = 100
Dataset 1: {'max_depth': 5}
Test Score: 0.8461538461538461
Dataset 2: {'max_depth': 10}
Test Score: 0.9861111111111112
Dataset 3: {'max_depth': 18}
Test Score: 0.8867924528301887
```



The classifier is less affected by outliers as the number of trees increases. This is because many of the trees (mis)interpret the outliers differently, and thus the error averages out with an averaged classification over all trees.

3.1 RFECV

The following bar chart shows the most important features in the data set. We can plainly see for example, that feature number 13 is the most important.



The following is the output of the unpruned part of the script:

```
Score of Random Forest: 0.684
Score of SVC: 0.708
```

The following is the output of the pruned part of the script:

```
Score of SVC on pruned data: 0.744
```

We can see that we got a better score on the test data compared to taking all of the features. This tells us that many of the features in the dataset were redundant or irrelevent to the classification, which is why the classifier performed better without considering them.