Machine learning

*ePortfolio*



**Group 40**

Name: Stefan Winter

Student number: 2067606

Name: Joost Oudesluijs

Student number: 2056392

Name: Joost Schutte

Student number: 2069625

[880083-M-6](https://tilburguniversity.instructure.com/courses/5509)

Dr. Güven Ç and Dr. Önal, I.

Department of Cogntive science and Artificial intelligence

School of Humanities and Ditital sciences

Tilburg University

June 2021

|  |
| --- |
|  |

# Task 1

Description of your experiments and results including:

* features used
* learning model and algorithm used
* parameter tuning

**Principal component analysis**

Principal component analysis (PCA) is a preprocessing step before classification models are applied (Husson et al., 2010). PCA is often used to reduce the dimensionality and could therefore be seen as a denoising method. The first dimensions extract the most essential information, while the last dimensions extracts mainly noise.

**Support vector machines**

Support vector machines are algorithms that find a hyperplane that classifies all the datapoints to categories in an N-dimensional space (Ghandi, 2018). This hyperplane maximizes the distance or margin between different classes. The support vectors are the closest points to the classifier and determine the position of the hyperplane.

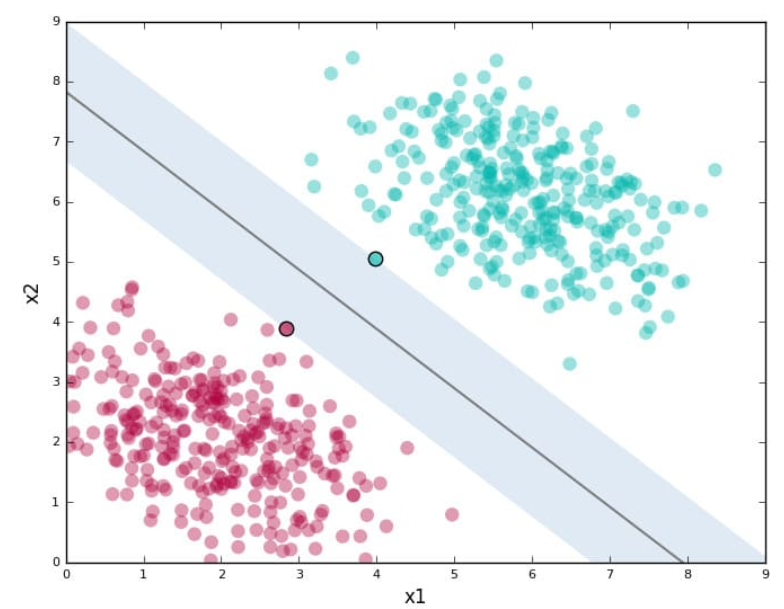


Figure 1: Visualized support vector machine. The black line represents the hyperplane with the maximized distance between the classes. The margin is the distance from the hyperplane to the first observations of both classes and is visualized as a light-blue area. The first observations to the classifier are the support vectors.

To correctly classify the datapoints, a kernel function projects low-dimensional data into a higher-dimensional space. Enhancing the dimensions makes it possible to better divide the observations into classes. Therefore support vector machines works more effective in high dimensional spaces, or datasets with a high number of input features.

To optimize support vector machines the cost parameter (C) could be tuned. With this cost parameter, one can determine the weight of x many samples inside the margin to contribute to the overall error. Most often the cost parameter is found by trial and error compared to a maximization or minimization of an evaluation metric.

**Our tuning…**

**Random forest**

Before random forest will be described, decision trees will be introduced. Decision trees is supervised learning model and is often used for classification tasks (James et al., 2013). The architecture of the decision tree is comparable with a regular tree. Therefore decision trees are built with roots, nodes, branches and leaves (Sato & Tsukimoto, 2001). The root node is the starting point of the tree and splits every node with recursive partitioning based on the largest information gain. At each coming leaf node, the data splits into partitions based on the feature with the second largest information gain. This process is done until a base case is reached or when the label is small enough.

Although decision trees are transparent and easily interpretable, this model overfits the data quite often. To overcome overfitting, random forests are introduced. Random forests differentiate on decision trees by generating multiple trees. The process of generating multiple trees is done by random sampling with replacement of training data points and random subsets of features when splitting nodes (Koehrsen, 2018). The random selection of features in each node allows variables to enter the model that otherwise would be overlooked (Strobl et al., 2008). The model is then trained with these random samples and subsets to decrease the total variance. When the predictions are made the bagging approach is applied. This means that the predictions of each tree are averaged. Therefore different features combinations could generate different predictions, the majority vote will result to the final prediction.

To increase performance of random forest hyperparameter can be tuned in multiple ways. The number of decision trees that comprise the random forest. The number of features available for splitting each tree node. The maximal size of terminal nodes.

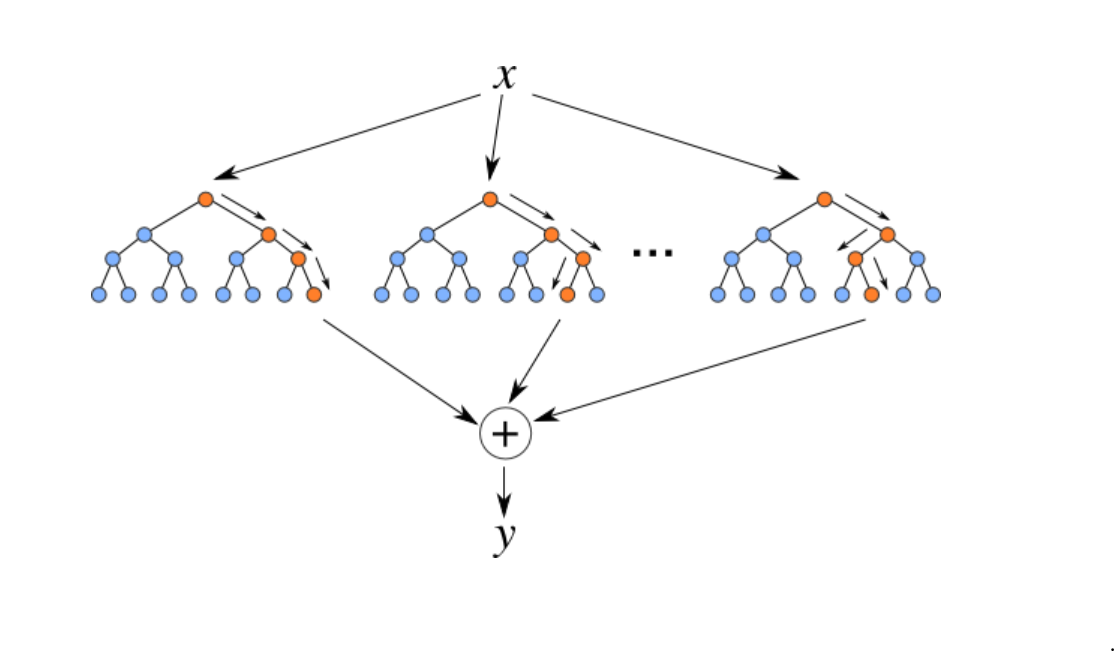


Figure 2: Visualized random forest classifier.

**Our tuning…**

* description of method or system built to perform classification
* discussion of experiments run and performance of your solution (this may include analysis such as confusion matrix, accuracy per class, etc.)

# Task 2

Description of your experiments and results including:

* features used
* learning model and algorithm used
* parameter tuning
* description of method or system built to perform classification

discussion of experiments run and performance of your solution (this may include analysis such as confusion matrix, accuracy per class, etc.)

# Reference

Breiman, L. (2001). Random forests. Machine learning, 45, 5-32.

Gandhi, R. (2018b, July 5). Support Vector Machine — Introduction to Machine Learning Algorithms. Retrieved from https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Koehrsen, W. (2018, December 1). An Implementation and Explanation of the Random Forest in Python. Retrieved from <https://towardsdatascience.com/an-implementation-and-> explanation-of-the-random-forest-in-python-77bf308a9b76

Sato, M. and H. Tsukimoto. 2001. Rule extraction from neural networks via decision tree induction. In Proceedings of the International Joint Conference on Neural Networks.

Strobl, C., Boulesteix, A. L., Kneib, T., Augustin, T., & Zeileis, A. (2008). Conditional variable importance for random forests. BMC bioinformatics, 9, 307.

# Appendices

## Group work

### Task 1

Stefan Winter

* Explaining Sourcetree

Joost Oudesluijs

Joost Schutte

### Task 2

Stefan Winter

Joost Oudesluijs

Joost Schutte