

Tutorium: Supervised Learning

- everything written in blue is additional information, things we would say during the tutorial
- everything in black is important information, definitions, etc., ...

Dear students!

Welcome to today's tutorial in Informatics 2. Parts of this session would have been held on the blackboard. We will try to give you this "blackboard feeling" at home as good as possible. ☺

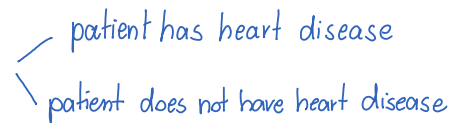
After this session you will know about:

- basic classification terminology
- basic functioning of the kNN classifier

Classification: Classification is a process of categorizing given data into classes.

Classes: Classes are often referred to as targets, labels or categories.

During classification, each data sample is assigned to a class based on its properties.

Example: Heart disease detection has two classes 

note: We talked a lot about classes in OOP with python, but the term "class" in machine learning has nothing to do with that.

Classifier: Is an algorithm that is used to map the input data to a specific category. Classifiers are also referred to as models.

Any classifier needs to learn how given input data is related to the classes.

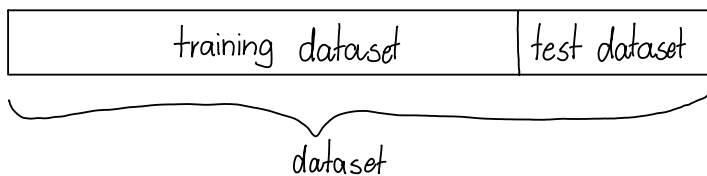
This process is called training or fitting. Afterwards we want to assess how good the trained classifier can predict "unseen" samples, which were not available during the training process. This is called testing.

Again let us think on the heart disease dataset. We could for example train our classifier with data recorded in Spain and test the classifier with data recorded in Switzerland.

Training Dataset: A training dataset is a set of samples used during the training process. It is used to fit the parameters of a classifier. The goal is to create a trained (also called fitted) classifier that will later also work with new, unknown data.

Test Dataset: A test dataset is independent of the training dataset. It is a set of samples used only to evaluate the performance of a classifier. This means that the trained classifier is used to classify samples in the test set. (Of course later on the trained classifier is also used to predict new/unseen data) Those classifications are compared to the ground truth classes of the samples. This can be used to calculate the classification accuracy.

So now a dataset could be split like the drawing below. We see that the training and the test dataset don't share any samples. (This is very important to correctly evaluate the classifier)

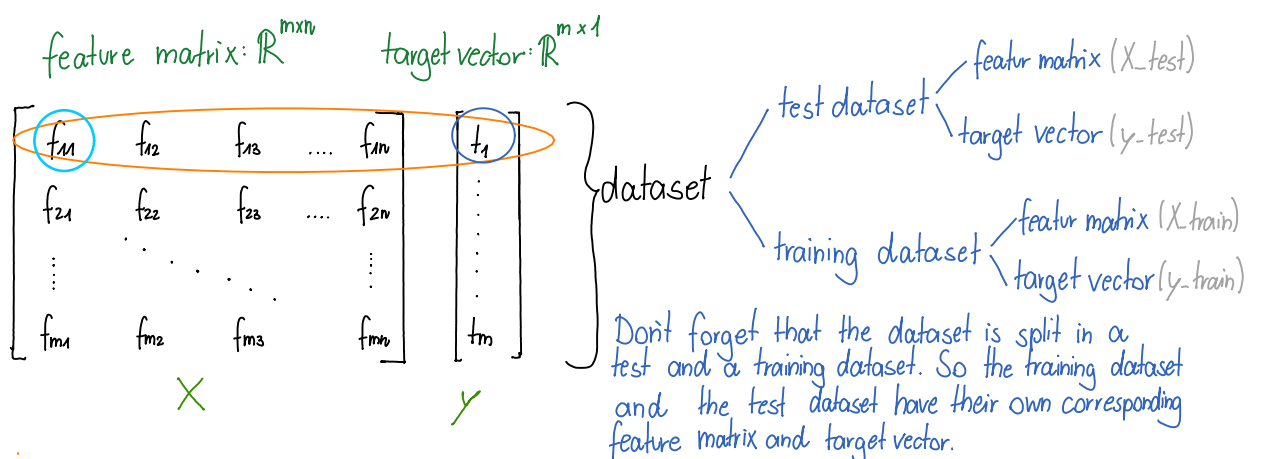


What do you think? How would you distribute the data? Do we need more training or more test data?

Now we can introduce a final definition of classification:

Def: Classification is the task of approximating a mapping function from input variables (X) to discrete output variables (y).

The standard representation of a dataset is by a feature matrix (X) and a corresponding target vector (y). A dataset consisting of c classes, m samples and each sample of n features and one target would look like this:



- one sample (out of m samples)
- one feature (out of n features)
- one class (out of c classes)

We already defined classes. But what about features and samples?

Samples: Samples refer to the individual objects described by the dataset. Each row in the matrix X contains all features (i.e. datapoints) for a single sample. For the heart disease dataset each row of the feature matrix X refers to a single observed person. In other datasets samples could be flowers (Iris dataset), sound files or anything else you can describe with a set of quantitative measurements.

Features: A feature is an individual measurable property of each sample that is observed. Each column of the matrix X contains the values of a single feature for all samples. Possible features of the heart disease dataset are: age, sex, smoker, resting heart rate, ...

With this knowledge the feature matrix and the target vector can be described as:

Feature matrix: Two-dimensional grid of data, where rows represent individual elements of the dataset, and columns represent properties known about each of these elements.

Target vector: One-dimensional array which contains classes/labels (the quantity we want to predict).

Again, this would be $\begin{cases} \text{no heart disease} \\ \text{heart disease} \end{cases}$ for the heart disease dataset \Rightarrow 2 classes

For flowers it could for example be the species $\begin{cases} \text{Iris Versicolor} \\ \text{Iris Setosa} \\ \text{Iris Virginica} \end{cases} \Rightarrow$ 3 classes

With this introduction we are now ready to get to know our first Classifier: kNN! First some general information:

In general there exist a lot of classification algorithms and it is not possible to say which one is better than others. It depends on the dataset and the desired application.

We distinguish between two types of learners in classification: Lazy Learners Vs. Eager Learners

Lazy Learners: They store the training data and wait until classification needs to be performed. This is done based on the most similar data in the stored training dataset. There is no need for learning or training the model.

kNN, Case-based reasoning

\hookrightarrow less training time, but more predicting time

Think of them as lazy, they literally wait until the last minute before classifying any data sample! ☺

Eager Learners: They built a generalized classification model based on the given training dataset

Decision Tree, Naive Bayes, Artificial Neural Networks

before even receiving data for classification.

Due to the model construction, they take a longer time to train, less time to predict.

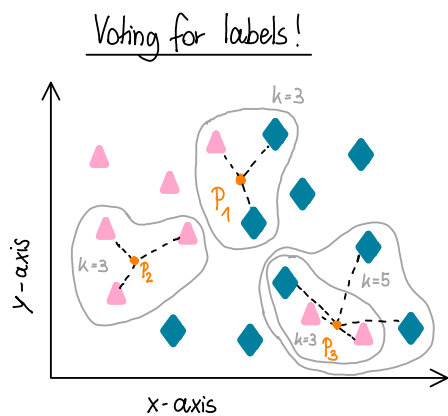
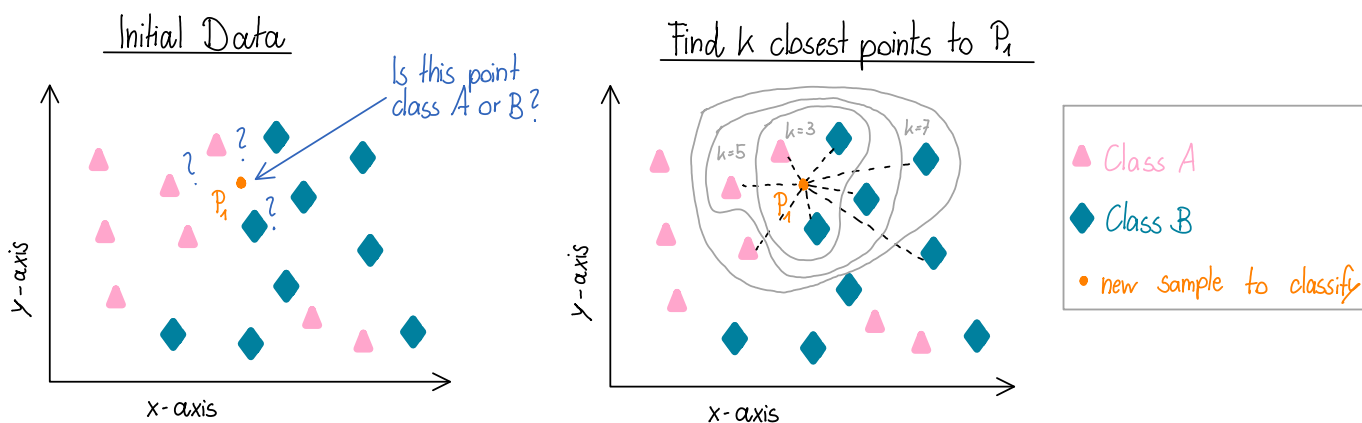
Think of them as active, being ready and eager to classify new data samples!

kNN... k - Nearest Neighbors

In kNN, "k" is the number of nearest neighbors. k is generally an odd number if the number of classes is two. The algorithm can be described in three simple steps:

- ① Calculate distances
- ② Find k closest neighbors
- ③ Vote for labels

Now we will look at an example. Let's assume P_1 is the point, which we want to classify, i.e. for which we want to predict a label/class.



We found three closest neighbors for P_1 . Now we check the classes of these closest points. The class with the most votes is then taken as the prediction.

$P_1 \dots [\triangle, \diamond, \diamond] \Rightarrow$ Prediction: \diamond Class B (one A, two B)

Let's try some more points with different k:

(k=3) $P_2 \dots [\triangle, \triangle, \triangle] \Rightarrow$ Prediction: \triangle Class A (all three are class A)

(k=3) $P_3 \dots [\triangle, \triangle, \diamond] \Rightarrow$ Prediction: \triangle Class A (two A, one B)

(k=5) $P_3 \dots [\triangle, \triangle, \diamond, \diamond, \diamond] \Rightarrow$ Prediction: \diamond Class B (two A, three B)

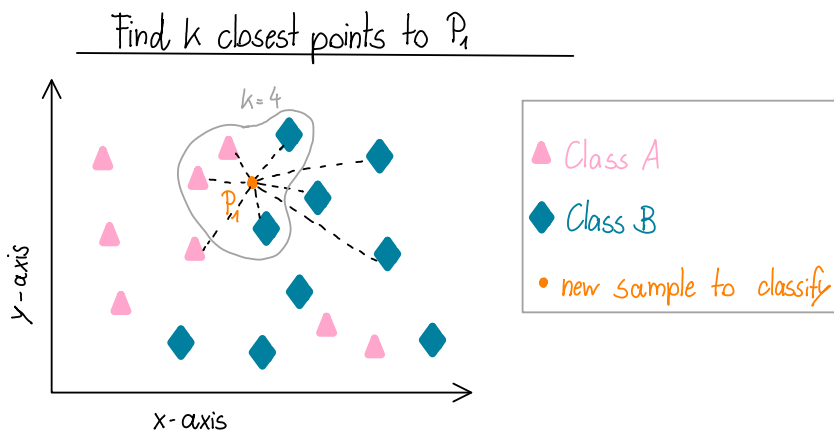
The example of P_3 shows, that the size of k can change the outcome of the prediction.

You can think of k as a controlling variable for the predicting model. There is no "optimal" number of neighbors that suits all kind of datasets. Also here each dataset has its own requirements. In general k should not be too high (decision is not local anymore) and not too low (noisy, often inaccurate).

Practice has shown, that $k = \sqrt{m}$, where m stands for number of samples in your training dataset works for most datasets.

But there is one thing we should not do when selecting the size of k .

As already mentioned k should not be an even number, if we have two classes.



$P_i \dots [\triangle, \triangle, \diamond, \diamond] \Rightarrow$ Prediction: ? (two A, two B)

In this scenario we can no longer predict point P_i with the method explained earlier. Now it depends on how the algorithm is implemented. Options i.e. are "random" which selects randomly one of the classes or "nearest" which calculates whether in sum the distances of one or the other class are shorter/nearer to P_i . The nearer one is then chosen as class/label.

Lastly there is a question for you! What do you think, is kNN a lazy or an eager learner?