**The Importance of Cross Validation in Machine Learning**

**Explaining why Machine Learning needs Cross Validation and how it is done in Python**

The cross validation method is used to test trained [machine-learning](https://databasecamp.de/en/machine-learning) models and to evaluate their performance independently. For this purpose, the underlying data set is divided into training data and test data. However, the model’s accuracy is then calculated exclusively on the test data set to assess how well the model responds to data that has not yet been seen.

**Why do you need Cross Validation?**

To train a general machine learning model, one needs data sets so that the model can learn. The goal is to recognize and learn certain structures within the data. Therefore, the size of the dataset should not be neglected, because too little information may lead to wrong insights.

The trained models are then used for real applications. That is, they are supposed to make new predictions with data that the AI has not seen before. For example, a [Random Forest](https://databasecamp.de/en/ml/random-forests) is trained to classify production parts as damaged or undamaged based on measurement data. The AI is trained with information about former products that are also uniquely classified as damaged or undamaged. Afterward, however, the fully trained model is to decide for new, unclassified parts from production whether they are flawless.

In order to simulate this scenario already in training, a part of the data set is deliberately not used for the actual training of the AI, but instead retained for testing in order to be able to evaluate how the model reacts to new data.

**What is Overfitting?**

The targeted withholding of data that is not used for training also has another concrete reason. The aim is to avoid so-called overfitting. This means that the model has adapted too much to the training data set and thus delivers good results for this part of the data, but not for new, possibly slightly different data.

Here is an honestly made-up example: Let’s assume we want to train a model that is supposed to deliver the perfect mattress shape as a result. If this AI is trained on the training dataset for too long, it may end up overweighting characteristics from the training set. This happens because the [backpropagation](https://databasecamp.de/en/ml/backpropagation-basics) still tries to minimize the error of the loss function.

In the example, it could lead to the fact that mainly side sleepers are present in the training set and thus the model learns that the mattress shape should be optimized for side sleepers. This can be prevented by not using part of the data set for actual training, i.e. for adjusting the weights, but only for testing the model once against independent data after each training run.

**What does Cross Validation do?**

Generally speaking, cross validation (CV) refers to the possibility of estimating the accuracy or quality of the model with new, unseen data already during the training process. This means that already during the learning process it is possible to estimate how the AI will perform in reality.

In this process, the data set is divided into two parts, namely training data and test data. The training data is used during model training to learn and adjust the weights of the model. The test data, in turn, is used to independently evaluate the accuracy of the model and validate how good the model already is. Depending on this, a new training step is started or the training is stopped.

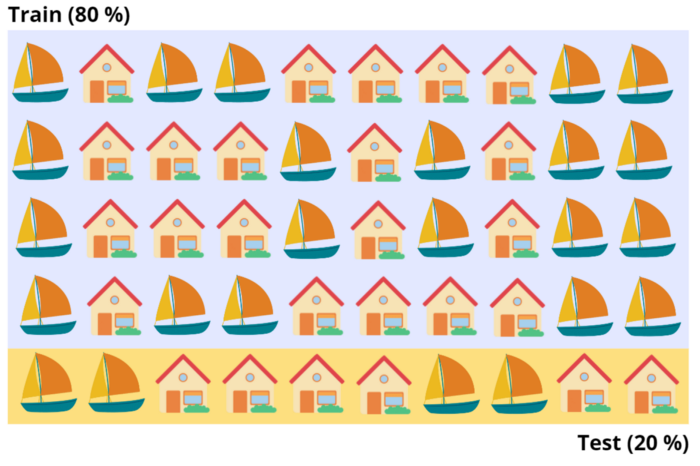
The steps can be summarized as follows:

1. Split the training data set into training data and test data.
2. Train the model using the training data.
3. Validate the performance of the AI using the test data.
4. Repeat steps 1–3 or stop the training.

To divide the data set into two groups, there are different algorithms that are chosen depending on the amount of data. The most famous ones are the Hold-Out and the k-Fold Cross Validation.

**How does Hold-Out Cross Validation work?**

The Hold-Out method is the simplest method to obtain training data and test data. Many people are not familiar with that name, but most will have used it before. This method simply holds out 80% of the data set as training data and 20% of the data set as test data. The split can be varied depending on the data set.



Train Test Split Example | Source: Author

Although this is a very simple and fast method, which is also frequently used, it also has some problems. For one thing, it can happen that the distribution of elements in the training data set and test data set are very different. For example, it could happen that boats are much more common in the training data than in the test data. As a result, the trained model would be very good at being able to detect a boat but would be evaluated on how well it detects houses. This would lead to very poor results.

In Scikit-Learn there are already defined functions with which the Hold-Out method can be implemented in [Python](https://databasecamp.de/en/python-coding) (example of [Scikit-Learn](https://scikit-learn.org/stable/modules/cross_validation.html)).

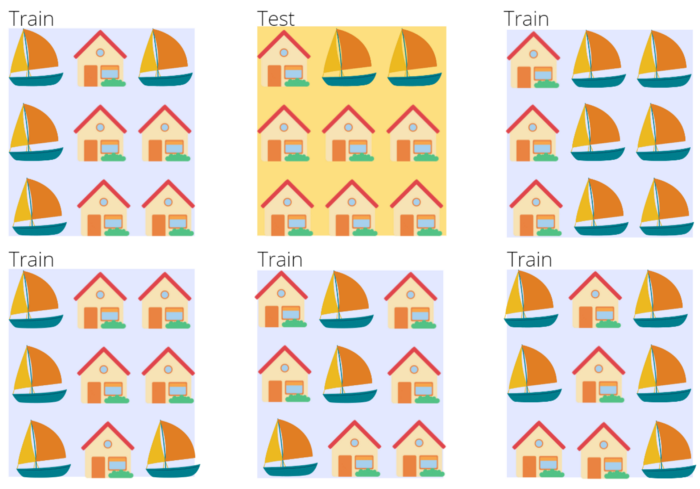
# Import the Modules  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn import datasets  
from sklearn import svm# Load the Iris Dataset  
X, y = datasets.load\_iris(return\_X\_y=True)Get the Dataset shape  
X.shape, y.shapeOut:  
((150, 4), (150,))# Split into train and test set with split 60 % to 40 %  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
X, y, test\_size=0.4, random\_state=0)print(X\_train.shape, y\_train.shape)  
print(X\_test.shape, y\_test.shape)Out:  
((90, 4), (90,))  
((60, 4), (60,))

Another problem with hold-out cross validation is that it should only be used with large data sets. Otherwise, there may not be enough training data left to find statistically relevant correlations.

**How does the k-Fold Cross Validation work?**

The k-Fold Cross Validation remedies these two disadvantages by allowing data sets from the training data to also appear in test data and vice versa. This means that the method can also be used for smaller data sets and it also prevents an unequal distribution of properties between training and test data.

The data set is divided into k blocks of equal size. One of the blocks is chosen randomly and serves as the test data set and the other blocks are the training data. Up to this point, it is very similar to the hold-out method. However, in the second training step, another block is defined as the test data, and the process repeats.



Cross Validation Example | Source: Author

The number of blocks k can be chosen arbitrarily and in most cases, a value between 5 and 10 is chosen. A too-large value leads to a less biased model, but the risk of overfitting increases. A too-small k value leads to a more biased model, as it then actually corresponds to the hold-out method.

Scikit-Learn also provides ready-made functions to implement k-fold cross validation:

# Import Modules  
import numpy as np  
from sklearn.model\_selection import KFold# Define the Data  
X = ["a", "b", "c", "d"]Define a KFold with 2 splits  
kf = KFold(n\_splits=2)# Print the Folds  
for train, test in kf.split(X):  
 print("%s %s" % (train, test))Out:   
[2 3] [0 1]  
[0 1] [2 3]

**This is what you should take with you**

* Cross validation is used to test trained machine learning models and independently evaluate their performance.
* It can be used to test how well the AI reacts to new, unseen data. This feature is also called generalization.
* Without cross validation, so-called overfitting can occur, in which the model over-learns the training data.
* The most commonly used cross validation methods are hold-out and k-folds.