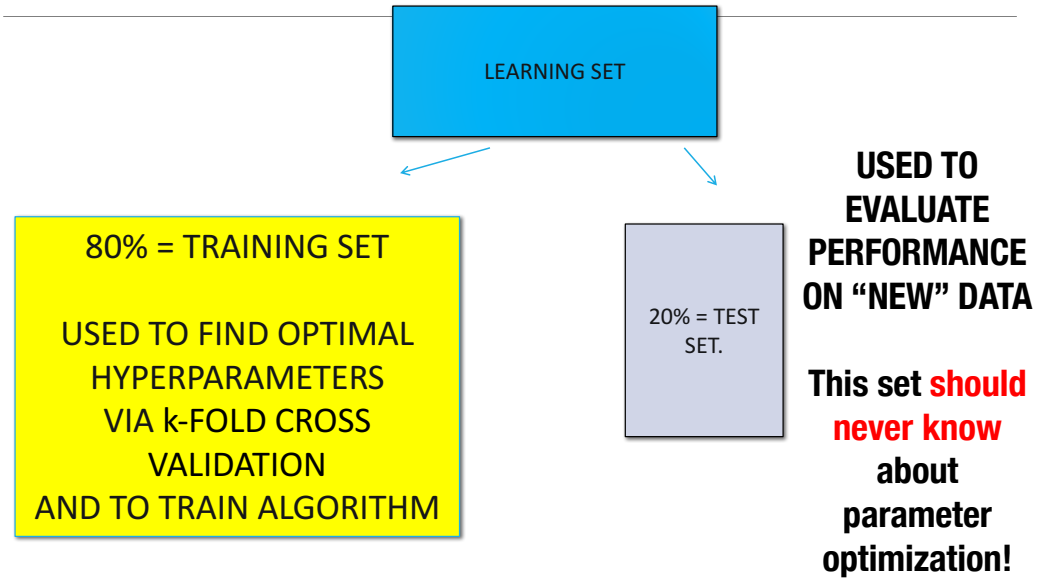
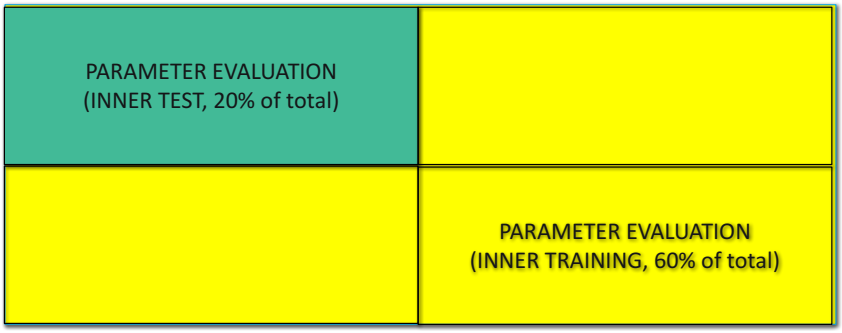
How can we optimize the hyperparameters?



Nested cross validation (nested = two levels, outer/inner)

Outer 5 fold cross validation:
takes 80% for training, 20% for testing
and averages performance 5 times (nothing new so far)

Inner cross validation **(inside yellow outer training set)**
will be used to pick hyper parameters;
we will do this 5 times
(or as many outer CV folds we have).



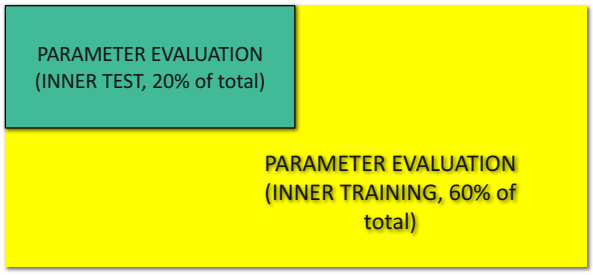
Here we show 4 fold inner cross validation on the outer training set.

Let's see it with one parameter:
C = {1, 10, 100}

1) Set C = 1

2) Do k fold inner cross validation on the outer training test (yellow, 80% of total). Here we divide in 4 folds where 3 are used for inner training, 1 for inner testing.

3) Report average of performance.

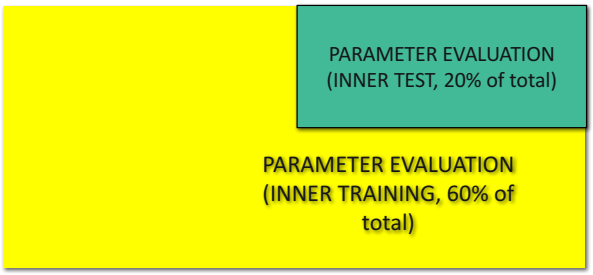


Let's see it with one parameter: C = {1, 10, 100}

1) Set C = 1

2) Do k fold inner cross validation on the outer training test (yellow, 80% of total). Here we divide in 4 folds where 3 are used for inner training, 1 for inner testing.

3) Report average of performance

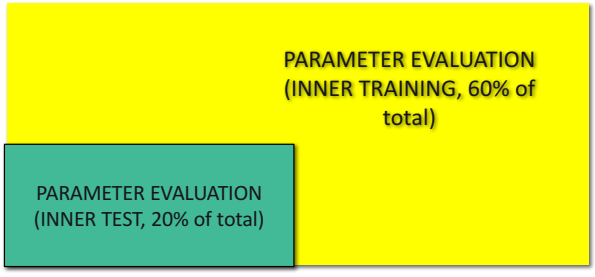


Let's see it with one parameter: C = {1, 10, 100}

1) Set C = 1

2) Do k fold inner cross validation on the outer training test (yellow, 80% of total). Here we divide in 4 folds where 3 are used for inner training, 1 for inner testing.

3) Report average of performance

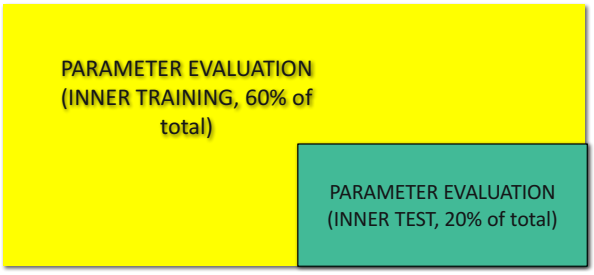


Let's see it with one parameter: C = {1, 10, 100}

1) Set C = 1

2) Do k fold inner cross validation on the outer training test (yellow, 80% of total). Here we divide in 4 folds where 3 are used for inner training, 1 for inner testing.

3) Report average of performance (inner test scores) on the four folds.

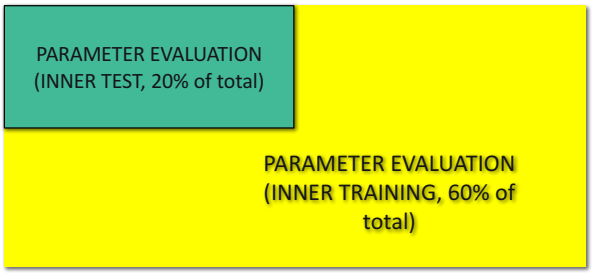


Let's see it with one parameter: C = {1, 10, 100}

1) Set C = 10 and repeat:

2) Do k fold cross validation on the outer training test (yellow, 80% of total). Here we divide in 4 folds where 3 are used for inner training, 1 for inner testing.

3) Report average of performance

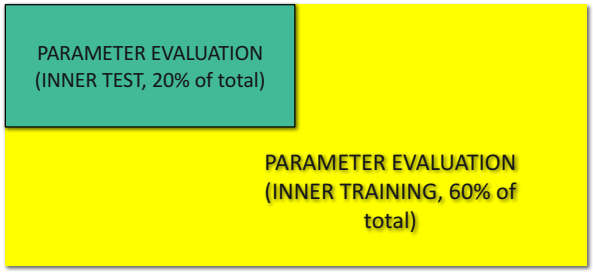


Let's see it with one parameter: C = {1, 10, 100}

1) Set C = 100 and repeat:

2) Do k fold cross validation on the outer training test (yellow, 80% of total). Here we divide in 4 folds where 3 are used for inner training, 1 for inner testing.

3) Report average of performance



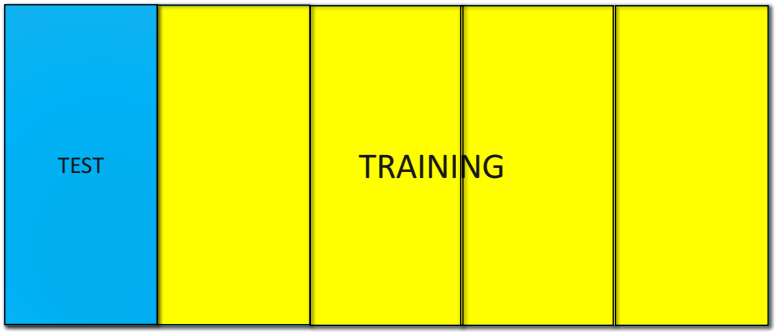
Summary of inner CV results: picking parameters Metric: accuracy

	Fold 1 = test Folds 2, 3, 4 = training	Fold 2 = test Folds 1, 3, 4 = training	Fold 3 = test Folds 1, 2, 4 = training	Fold 4 = test Folds 1, 2, 3 = training	Average accuracy
C = 1	80%	85%	79%	84%	82%
C = 10	83%	87%	78%	77%	81%
C = 100	81%	83%	80%	76%	80%

Summary of inner CV results: picking parameters Metric: accuracy

	Fold 1 = test Folds 2, 3, 4 = training	Fold 2 = test Folds 1, 3, 4 = training	Fold 3 = test Folds 1, 2, 4 = training	Fold 4 = test Folds 1, 2, 3 = training	Average accuracy
C = 1	**80%**	**85%**	**79%**	**84%**	**82%**
C = 10	83%	87%	78%	77%	81%
C = 100	81%	83%	80%	76%	80%

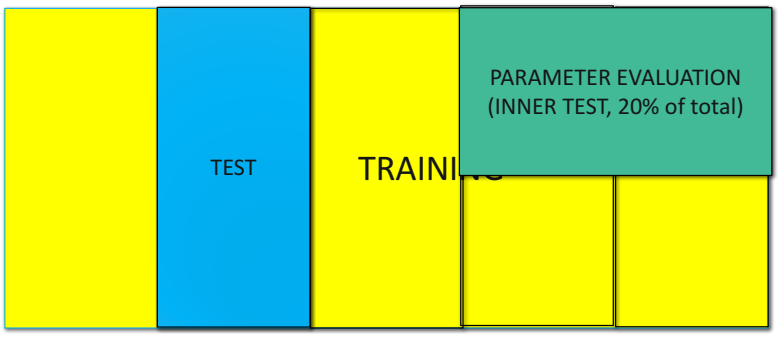
Now I apply the winning model (with C = 1) to the test set of the first outer fold.



This way the "test" set has never seen the training data and has never participated in the hyper-parameter optimization.

I'll save the accuracy score: e.g., 82%.

Now we go back to outer cross validation, pick the second fold as test and repeat the process.



Summary of inner CV results on this second outer fold: picking parameters
Metric: accuracy

	Fold 1 = test Folds 2, 3, 4 = training	Fold 2 = test Folds 1, 3, 4 = training	Fold 3 = test Folds 1, 2, 4 = training	Fold 4 = test Folds 1, 2, 3 = training	Average
C = 1	80%	83%	78%	83%	81%
C = 10	84%	87%	81%	80%	83%
C = 100	81%	83%	80%	76%	80%

Summary of inner CV results on this second outer fold: picking parameters
Metric: accuracy

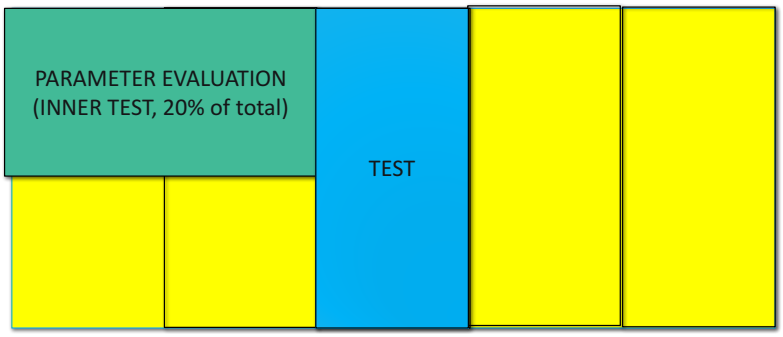
	Fold 1 = test Folds 2, 3, 4 = training	Fold 2 = test Folds 1, 3, 4 = training	Fold 3 = test Folds 1, 2, 4 = training	Fold 4 = test Folds 1, 2, 3 = training	Average
C = 1	80%	83%	78%	83%	81%
C = 10	**84%**	**87%**	**81%**	**80%**	**83%**
C = 100	81%	83%	80%	76%	80%

Now I apply the winning model (with C = 10) to the test set of the second outer fold.

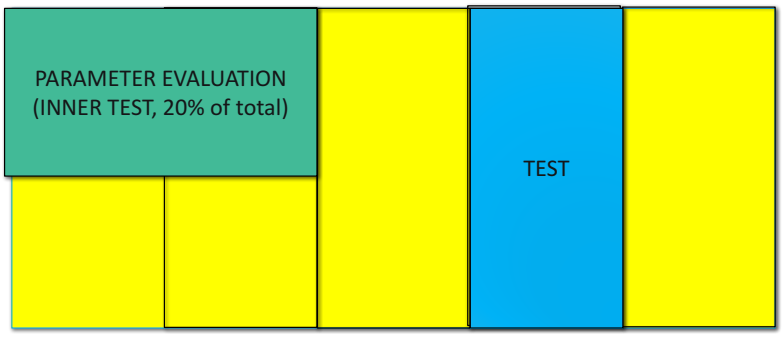
This way the "test" set has never seen the training data and has never participated in the hyper-parameter optimization.

I might get a different accuracy, e.g. 80%. I save this.

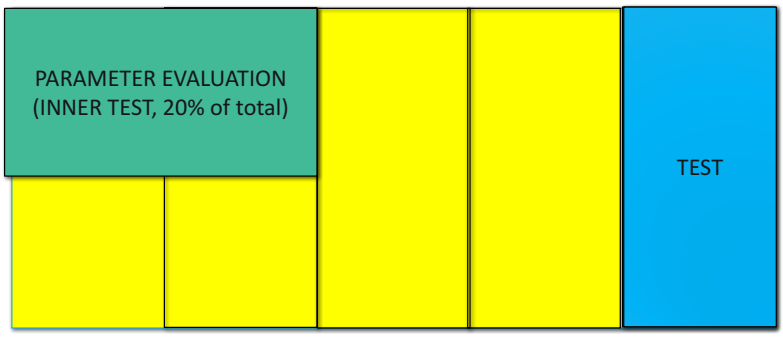
I repeat this process of inner CV for the other three folds, so I will have 5 winning models and 5 scores.



I repeat this process of inner CV for the other three folds, so I will have 5 winning models and 5 scores.



I repeat this process of inner CV for the other three folds, so I will have 5 winning models and 5 scores.



At the end I will have the five performances (outer test scores) of the five winning models.

The average of these gives me the average performance of my optimized SVM.

For example, 82, 80, 83, 84, 81 %. This gives me a "quotable generalization performance" (average) of 82% with a standard deviation of 1.6%.

Note that the five winning models might have different hyperparameters. That's ok as long as they are not CRAZY different. They are only used to evaluate the generalization error.

Final bit: If I am building an algorithm for use on new data, I now will use all the training data to build a model and only perform one level of cross validation to find ideal hyperparameters. They might be different from what I found in the previous process as well. I will quote as accuracy (or any other metric) the performance found above.

Updated summary of how to build a ML model

1. **Choose a class of model (aka a machine learning algorithm) by importing the appropriate estimator class from Scikit-Learn. Today: SVM.**
2. **Choose model hyperparameters by instantiating this class with desired values. Alternatively: optimize hyperparameters.**
3. **Arrange data into a features matrix and target vector, if necessary.**
4. **Split the learning set into training/test using k fold cross validation (outer CV), let's say 5. Any pre-processing, e.g. standardization/feature selection needs to happen on the train set only.**
5. **Split the training set into inner training/inner validation set using k fold cross validation (inner CV), let's say 4.**
6. **Fit the model in the inner loop by calling the ``fit()`` method for every combination of parameters you want to try out.**
7. **Find combination of parameters with the best average "inner test" scores (averaged over the inner 4 folds).**
8. **Repeat for the outer 5 folds. Report mean scores and standard deviation of winning models. That's the generalization error.**
9. **If performance still not satisfactory, diagnose bias/variance and proceed.**
10. **Build final model using all training data, using k-fold CV to find ideal parameters, and reporting generalization error found above.**