**Streamling workflows with pipelines:**

1. Loading dataset
2. Combining transformers and estimators in a pipeline

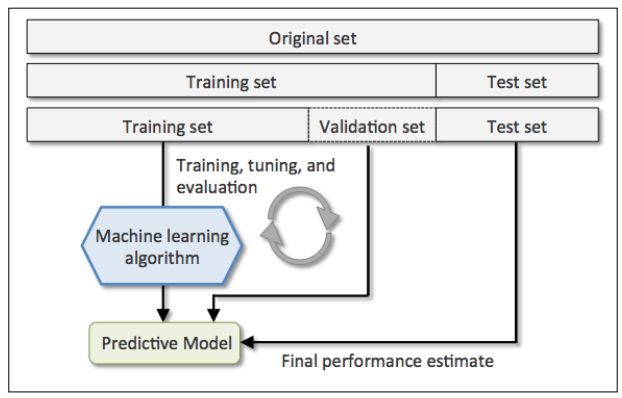
Instead of going through the fitting and transformation steps for the training and test dataset separately, we can chian the StandardScaler,PCA,and LogisticRegression objects in a pipeline:

**Using K-fold cross-validation to assess model performance**

These methods can help us to obtain reliable estimates of the model’s generalization error, that is, how well the model performs on unseen data.

1. **Holdout cross-validation method**

A better way of using the holdout method for model selection is to separate the data into three parts: a training set, a validation set, and a test set. The training set is used to fit the different models, and the performance on the validation set is the used for the model selection. Illustrations as follow:



The disadvantage of the holdout method is that the performance estimate is sensitive to how we partition the training set into the training and validation subsets.

1. **K-fold cross-validation**

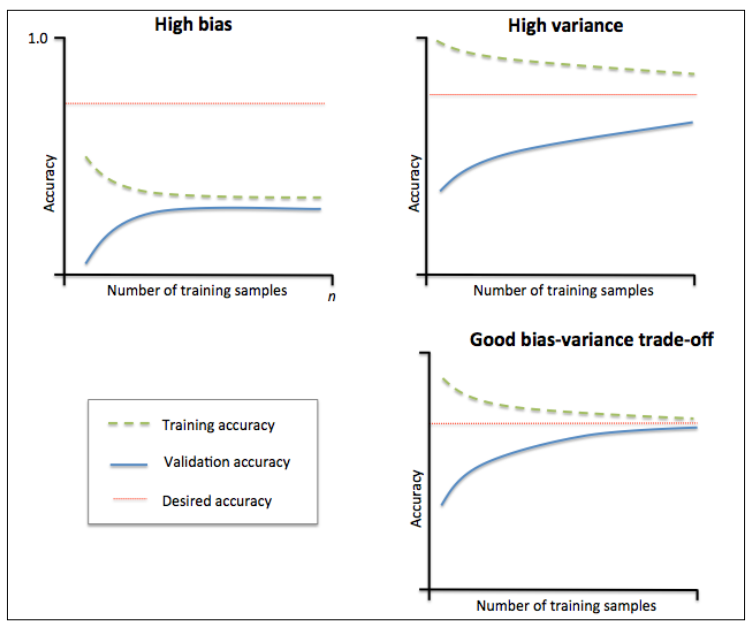
IN k-fold cross-validation, we randomly split the training dataset into k folds without replacement, where k-1 folds are used for the model training and one fold is used for testing.

Typically, we use k-fold cross-validation for model tuning, that is, fnding the optimal hyperparameter values that yield a satisfying generalization performance. Once we have found satisfactory hyperparameter values, we can retrain the model on the complete training set and obtain a fnal performance estimate using the independent test set.

If we are working with relatively small training sets , it can be useful to increase the number of folds. If we increase the value of k, more training data will be used in each iteration, which results in a lower bias towards estimating the generalization performance by averaging the individual model estimates. However , large values of k will also increase the runtime of the cross-validation algorithm and yield estimates with higher variance since the training folds will be more similar to each other. On the other hand, if we are working with large datasets, we can choose a smaller value for k, for example, k = 5 , and still obtain an accurate estimate of the average performance of the model while reducing the computational cost of refitting and evaluating the model on the different folds.

**Debugging algorithms with learning and validation curves**

By plotting the model training and validation accuracies as functions of the training set size, we can easily detect whether the model suffers from high variance or high bias, and whether the collection of more data could help to address this problem



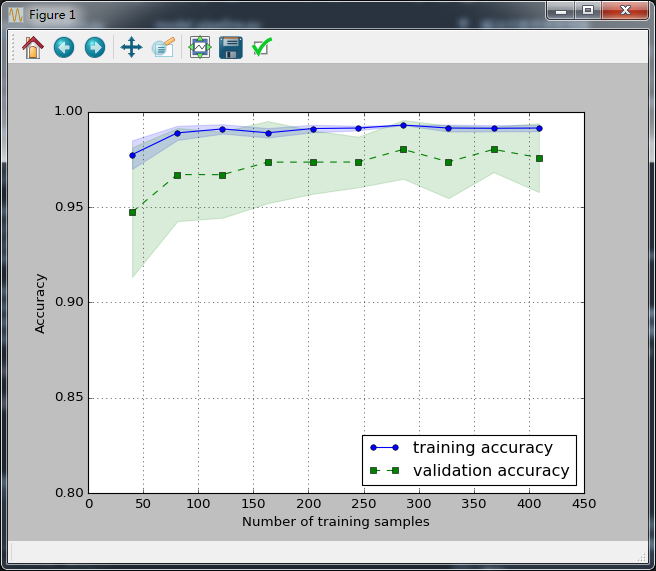
The training accuracy curve is wrong in the figure above.

Upper-left: high-bias (underfit), has both low training and cross-validation accuracy

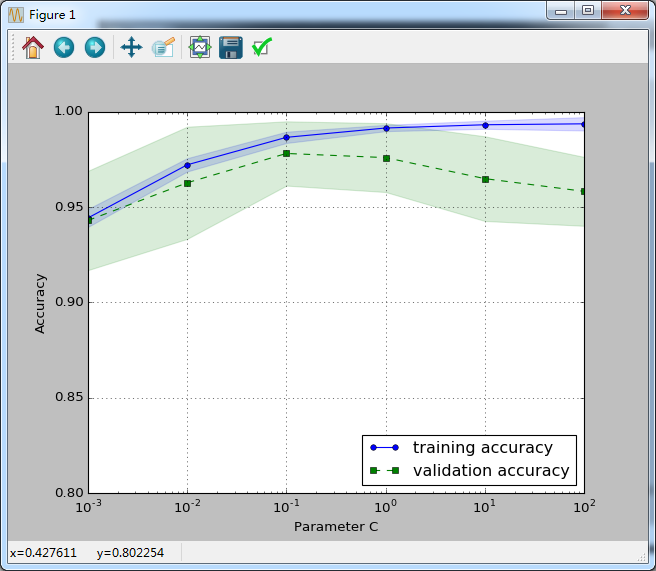
Solutions: 1. Increase the number of parameters of the model , collecting or constructing additional feature;2. Decreasing the degree of regularization, (SVM or logistic regression)

Upper-right: high-variance, a large gap between the training and cross-validation accuracy

Solutions: 1. Collect more training data or reduce the complexity of the model, eg. Increasing the regularization parameter;2. Decrease the number of features via feature selection or feature extraction.



**Addressing overfitting and underfitting with validation curves:**



Although the differences in the accuracy for varying values of C are subtle, we can see that the model slightly underfits the data when we increase the regularization strength (small values of C). However ,for large values, it means lowering the strength of regularization ,so the model tends to slightly overfit the data. In this case, the sweet spot appears to be around C= 0.1

**Fine-tuning machine learning models via grid search**

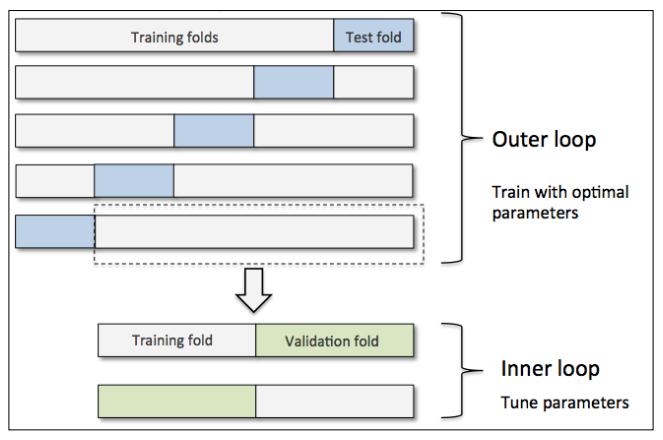
Two types of parameters: 1. Those learned from the training data, eg. Weights in logistic regression, 2. Parameters of a learning algorithm (tuning parameters, also called hyperparamters, regularation parameter in logistic regression or depth parameter of a decision tree)

It’s a bruth-force exhaustive search paradigm where we specify a list of values for different hyperparamters, and the computer evaluates the model performance for each combination of those to obtain the optimal set:

**Algorithm selection with nested cross-validation**

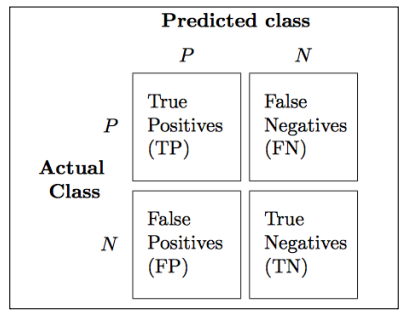
If we want to select among different machine learning algorithms though, another recommended approach is **nested cross-validation**.

In nested cross-validation , we have an outer k-fold cross-validation loop to split the data into training and test folds, and an inner loop is used to select the model using k-fold cross-validation on the training fold. After model selection , the test fold is then used to evaluate the model performance. Following figure explains the concept



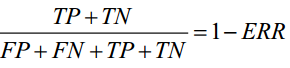
**Looking at different performance evaluation metrics**

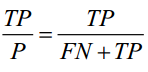
Confusion matrix: a matrix that lays out the performance of a learning algorithm. The confusion matrix is simply a square matrix that reports the counts of the true positive, true negative, false positive, and false negative predictions of a classifier

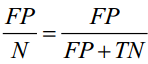


Different:

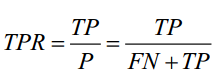
Error (ERR) = 

ACC(accuracy) = 

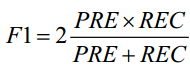
TPR (true positive rate) = 

FPR (false positive rate) = 

PRE (Precison) = 

REC (recall) = 

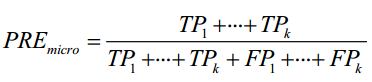
In Practice, combination of precision and recall is used, the so-called F1-score:



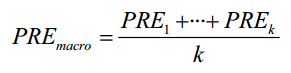
**Plotting a receiver operating characteristic**

**The scoring metrics for multiclass classifcation**

The micro-average of the precision score in a k-class system can be calculated as follows:



The macro-average is simply calculated as the average scores of the different systems:



Reference:

1. Python machine learning
2. <https://en.wikipedia.org/wiki/Receiver_operating_characteristic>
3. *The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms*