# Model selection

Model selection in machine learning is the process of selecting the best algorithm and model architecture for a specific job or dataset. It entails assessing and contrasting various models to identify the one that best fits the data & produces the best results.

Model selection is a process that can be applied both across different types of models (e.g. logistic regression, SVM, KNN, etc.) and across models of the same type configured with different model hyperparameters (e.g. different kernels in an SVM).

# k-Fold Cross-Validation

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

If you have a machine learning model and some data, you want to tell if your model can fit. You can split your data into training and test set. Train your model with the training set and evaluate the result with test set. But you evaluated the model only once and you are not sure your good result is by luck or not. You want to evaluate the model multiple times so you can be more confident about the model design.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

Note that k-fold cross-validation is to evaluate the model design, not a particular training. Because you re-trained the model of the same design with different training sets.

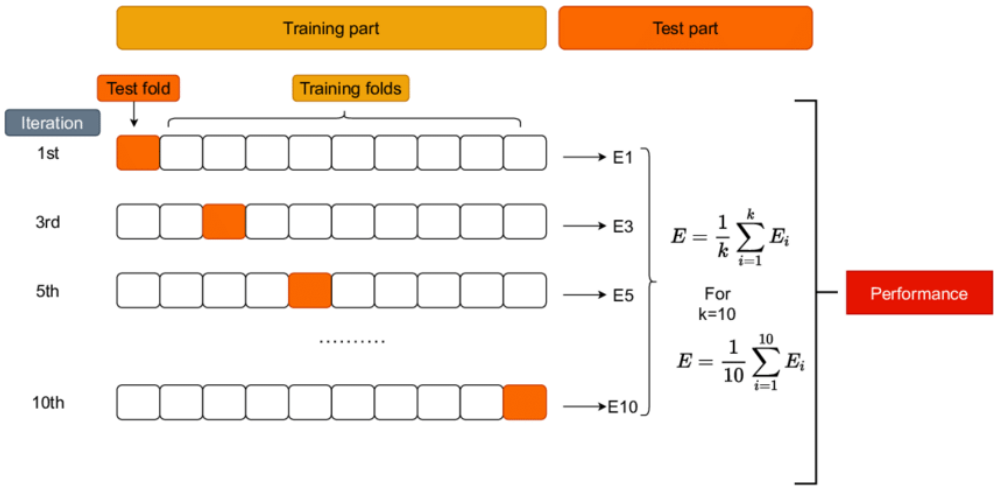
The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set

4. Retain the evaluation score and discard the model

4. Summarize the skill of the model using the sample of model evaluation scores

Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.




**Worked Example**

To make the cross-validation procedure concrete, let's look at a worked example.

Imagine we have a data sample with 6 observations:

1[0.1, 0.2, 0.3, 0.4, 0.5, 0.6]

The first step is to pick a value for k in order to determine the number of folds used to split the data. Here, we will use a value of k=3. That means we will shuffle the data and then split the data into 3 groups. Because we have 6 observations, each group will have an equal number of 2 observations.

For example:

1Fold1: [0.5, 0.2]

2Fold2: [0.1, 0.3]

3Fold3: [0.4, 0.6]

We can then make use of the sample, such as to evaluate the skill of a machine learning algorithm.

Three models are trained and evaluated with each fold given a chance to be the held out test set.

For example:

- **Model1**: Trained on Fold1 + Fold2, Tested on Fold3

- **Model2**: Trained on Fold2 + Fold3, Tested on Fold1
- **Model3**: Trained on Fold1 + Fold3, Tested on Fold2

The models are then discarded after they are evaluated as they have served their purpose.

The skill scores are collected for each model and summarized for use.

# Bias–variance tradeoff

In statistics and machine learning, the bias–variance tradeoff describes the relationship between a model's complexity, the accuracy of its predictions, and how well it can make predictions on previously unseen data that were not used to train the model. In general, as we increase the number of tuneable parameters in a model, it becomes more flexible, and can better fit a training data set. It is said to have lower error, or bias. However, for more flexible models, there will tend to be greater variance to the model fit each time we take a set of samples to create a new training data set. It is said that there is greater variance in the model's estimated parameters.

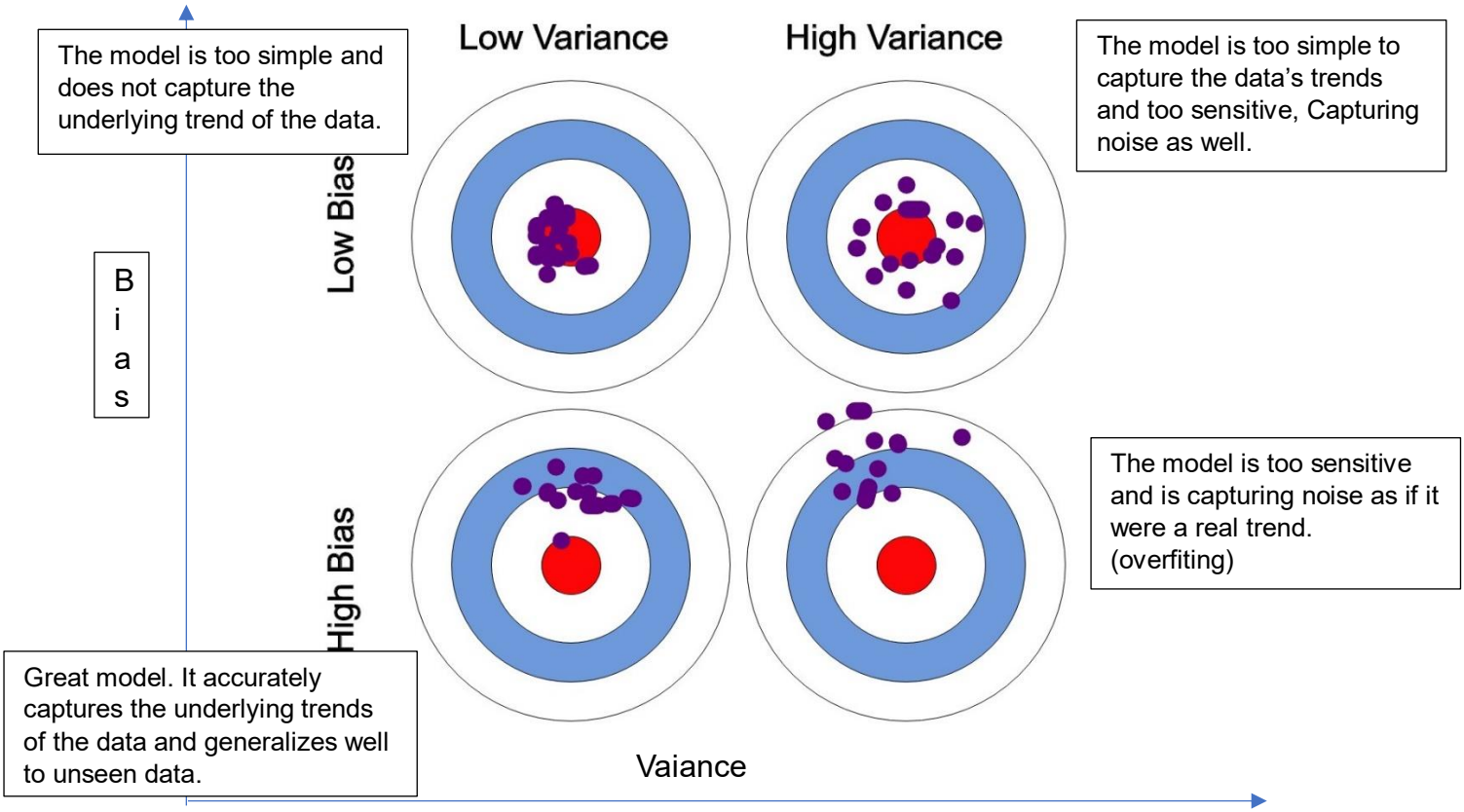
The bias–variance dilemma or bias–variance problem is the conflict in trying to simultaneously minimize these two sources of error that prevent supervised learning algorithms from generalizing beyond their training set:

- **Bias**: The *bias* error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

  A systematic error that occurs in the machine learning model itself due to incorrect assumptions in the ML process. Technically, we can define bias as the error between average model prediction and the ground truth.

- **Variance:** The *variance* is an error from sensitivity to small fluctuations in the training set. High variance may result from an algorithm modeling the random noise in the training data (overfitting).

  How much the model can adjust depending on the given data set. Variance refers to the changes in the model when using different portions of the training data set.




The bias–variance tradeoff is a central problem in supervised learning. Ideally, one wants to choose a model that both accurately captures the regularities in its training data, but also generalizes well to unseen data. Unfortunately, it is typically impossible to do both simultaneously. High-variance learning methods may be able to represent their training set well but are at risk of overfitting to noisy or unrepresentative training data. In contrast, algorithms with high bias typically produce simpler models that may fail to capture important regularities (i.e. underfit) in the data.