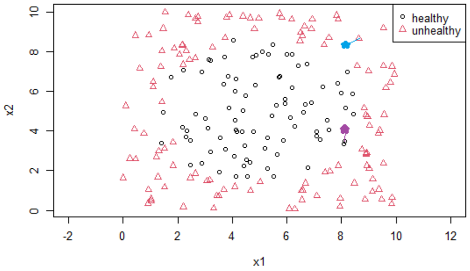
# Cross-validation

Ryan Miller

Grinnell College
Statistics

## Introduction

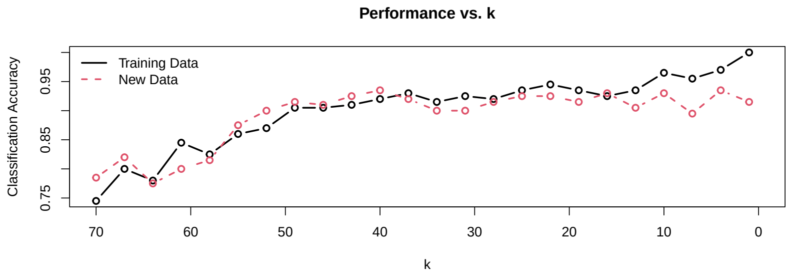
So far, we've been working with toy data involving the classification of healthy vs. unhealthy samples:



In our first lecture we established that we are less interested in classification performance on the *training data* and instead care more about performance on unseen *test data*

## Introduction (cont.)

Below is the classification accuracy of various KNN models for this toy data set:



Small values of $k$ will *overfit the training data*, while for large values $k$ introduce too much bias.

## Test Data vs. Model Selection

- After the test data has been used once it will no longer allow for an unbiased evaluation of model performance
    - That is, we could choose the value of $k$ that performs best on our test data, but this value would be "cherry-picked" and we'd expect the corresponding error rate to be too optimistic
- What might we do to avoid over reliance on the test data?

## Single Validation

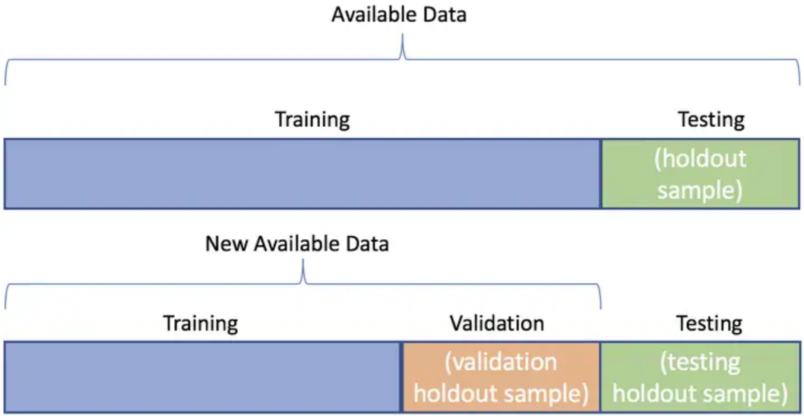


image credit: https://algotrading101.com/learn/train-test-split/

## Single Validation (cont.)

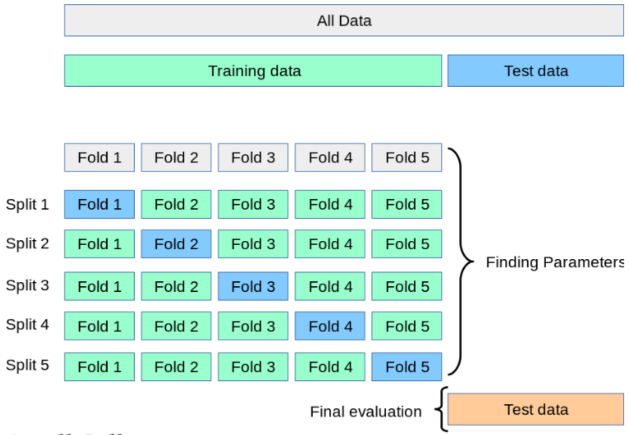
Carving off a separate validation set from the training data has a few distinct disadvantages:

1. Our performance estimate is highly dependent upon the samples assigned to the validation set and might not generalize to other collections of new data (high variance estimate of performance)
2. We might make decisions that overfit to this single validation set, undermining some of its value

Grinnell College
Statistics

## Cross-validation

Cross-validation provides more robust estimates of model performance by repeating the training-validation process on different "folds" of data. Shown below is a diagram of 5-fold cross-validation:



Grinnell College Statistics

## Cross-validation (cont.)

Pseudocode of 5-fold cross-validation:

```
## Assign n obs into k folds
fold_id = sample(1:k, size = n)

## Loop through each fold:
for i in 1:k
   train_X = samples[fold_id != i]
   train_y = labels[fold_id != i]
   eval_X = samples[fold_id == i]
   eval_y = labels[fold_id == i]

   model.fit(train_X, train_y)
   pred[fold_id == i] = model.predict(eval_X, eval_y)

## Calculate performance
score(pred)
```

Grinnell College
Statistics

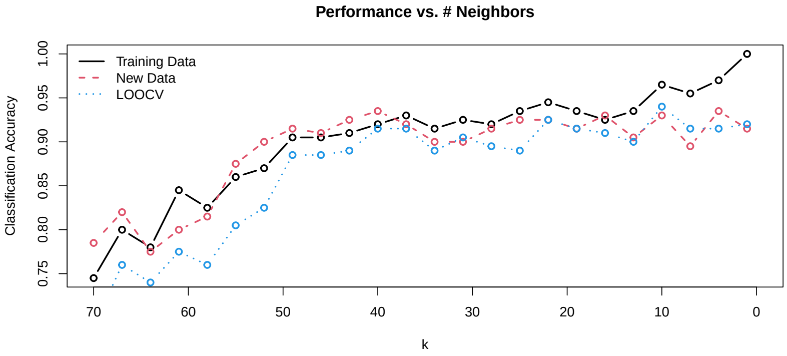
## Cross-validation (cont.)

Cross-validation (CV) is a general term describing methods of repeated data-splitting used to evaluate the performance of a model on data that was not used in the training process:

1. $k$-**fold cross-validation** - partitions the training data into $k$ equally sized folds. This approach is *non-exhaustive*, meaning it doesn't consider every possible arrangement and thus performance estimates will vary from one set of fold assignments to another
2. **Leave-one-out cross-validation (LOOCV)** - training and validation are repeated $n$ times while holding out each single sample as the validation set. This approach is *exhaustive*, but often more computationally expensive.

## Example - LOOCV

Cross-validation helps us *avoid overfitting* by producing reasonable estimates of performance without using the test set:



Grinnell College Statistics

## *k*-fold or LOOCV?

- LOOCV is a special case of $k$-fold CV using $k = n$ (ie: each fold contains just a single observation)
- Compared to $k$-fold CV, LOOCV is a *higher variance procedure*
  - Repeating LOOCV on different samples from the same underlying data generation process will show a greater range of performance estimates
  - Errors in LOOCV are highly correlated because each training set overlaps almost entirely

## $k$-fold or LOOCV?

- LOOCV is a special case of $k$-fold CV using $k = n$ (ie: each fold contains just a single observation)
- Compared to $k$-fold CV, LOOCV is a *higher variance procedure*
    - Repeating LOOCV on different samples from the same underlying data generation process will show a greater range of performance estimates
    - Errors in LOOCV are highly correlated because each training set overlaps almost entirely
- $k$-fold cross-validation offers *better generalization*, but can be unfeasible for small samples
    - Smaller values of $k$ are also more computationally efficient (especially if parallelization can be used)

## Grid Search

- Cross-validation provides a framework for unbiased performance evaluation *using only the training data*
    - However, it must be combined with other methods in order to determine the best values for a method's hyperparameters (ie: finding the best number of neighbors)

## Grid Search

- Cross-validation provides a framework for unbiased performance evaluation *using only the training data*
  - However, it must be combined with other methods in order to determine the best values for a method's hyperparameters (ie: finding the best number of neighbors)
- **Grid search** is a simple (and widely used) approach for finding effective combinations of tuning parameters using cross-validation
  - The idea is to systematically and exhaustively search a grid of candidate values that span interesting areas of the parameter space

## Grid Search (example)

Here's an example parameter grid for KNN that explores $k \in \{3, 4, 5\}$, Euclidean or Manhattan distance, and uniform or distance weighting:

| k | Distance | Weight |
|---|---|---|
| 3 | euclidean | uniform |
| 4 | euclidean | uniform |
| 5 | euclidean | uniform |
| 3 | manhattan | uniform |
| 4 | manhattan | uniform |
| 5 | manhattan | uniform |
| 3 | euclidean | distance |
| 4 | euclidean | distance |
| 5 | euclidean | distance |
| 3 | manhattan | distance |
| 4 | manhattan | distance |
| 5 | manhattan | distance |

Python will allow us to evaluate each row of this grid via cross-validation *using the same fold assignments*.

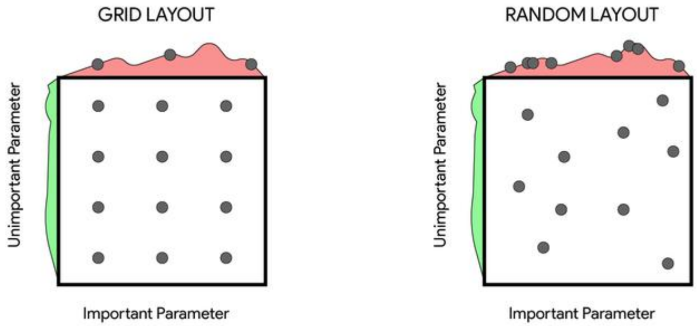
## Randomized Search

- Grid search can be computationally expensive, especially when you'd like to explore a broad range of values for several different hyperparameters
- **Randomized search** is an alternative method that allows you specify distributions to be randomly sampled from for each hyperparameter
  - For KNN, you might sample $k$ from a Poisson distribution with $\mu = \sqrt{n}$

Grinnell College
Statistics

## Grid Search vs. Randomized Search

An underappreciated fact is that the uniform spacing in grid searches can sometimes prevent us from finding optimal values for important parameters:



## Other Approaches

- Randomized search and grid search approaches can be combined to explore large parameter spaces with greater efficiency
    - For example, you might perform several iterations of random search to eliminate parameter values that lead to poor performance, then you might conduct a grid search over the remaining possibilities
- **Successive halving searches** are also supported in Python's `sklearn` library
    - The main idea is to only allow top scoring parameter values to "survive" into later rounds of the search
    - Although we won't cover this method in detail, you're welcome to explore it for use on your final project

## What to Know for Our Next Quiz

- Understand the basic steps behind k-fold cross-validation, including:
  - Dividing the data into k equally sized parts
  - Training a model using the samples in k-1 of these parts and using it to make predictions on the left out fold
  - Repeating until all folds have been left out exactly once
- Understand how k-fold cross-validation can be combined with approaches like grid search and randomized search to make informed choices of hyperparameters and data pre-processing steps