# Cross-validation

Ryan Miller

# Outline

1. Tuning parameters
2. Cross-validation
3. Grid search and tuning parameter optimization

# $k$-Nearest Neighbors

Previously, we learned about *$k$-nearest neighbors*, a supervised learning algorithm that makes predictions using the $k$ closest observed samples to a new data-point. In order to use the algorithm, we must provide the following:

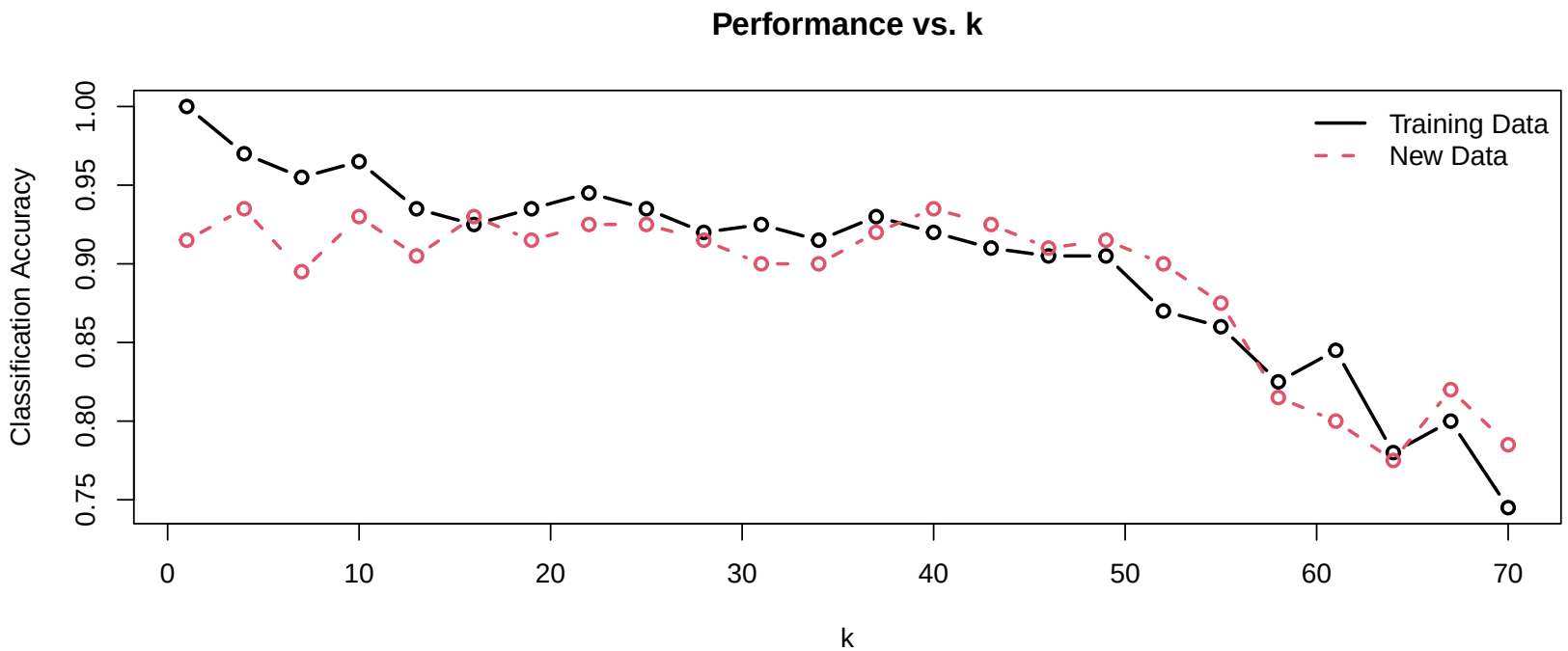
1. A way of measuring distance
2. A value for $k$
3. Whether neighbors should contribute uniformly or be weighted by distance

# Tuning Parameters

- A **parameter** is an unknown value that is required for an algorithm to make predictions
  - Some parameters, such as the coefficients in a regression model, are estimated from the data by the algorithm
- **Tuning parameters** (sometimes called *hyperparameters*) must be specified before an algorithm can run
  - For $k$-nearest neighbors, the value of $k$, the distance metric, and the weighting scheme are tuning parameters

# Tuning Parameters

Shown below is the classification accuracy of various kNN models for our toy data (healthy vs. unhealthy classification):




For small values of $k$, models *overfit the training data*, while for large values of $k$ models are *too biased to be useful*.

# Tuning Parameters

- We don't want to use the test set to determine our tuning parameters
  - This would introduce the very same problems we were trying to avoid by using independent training and testing sets
- So, how might we determine good values for tuning parameters using *only the training data*?

# Cross-validation

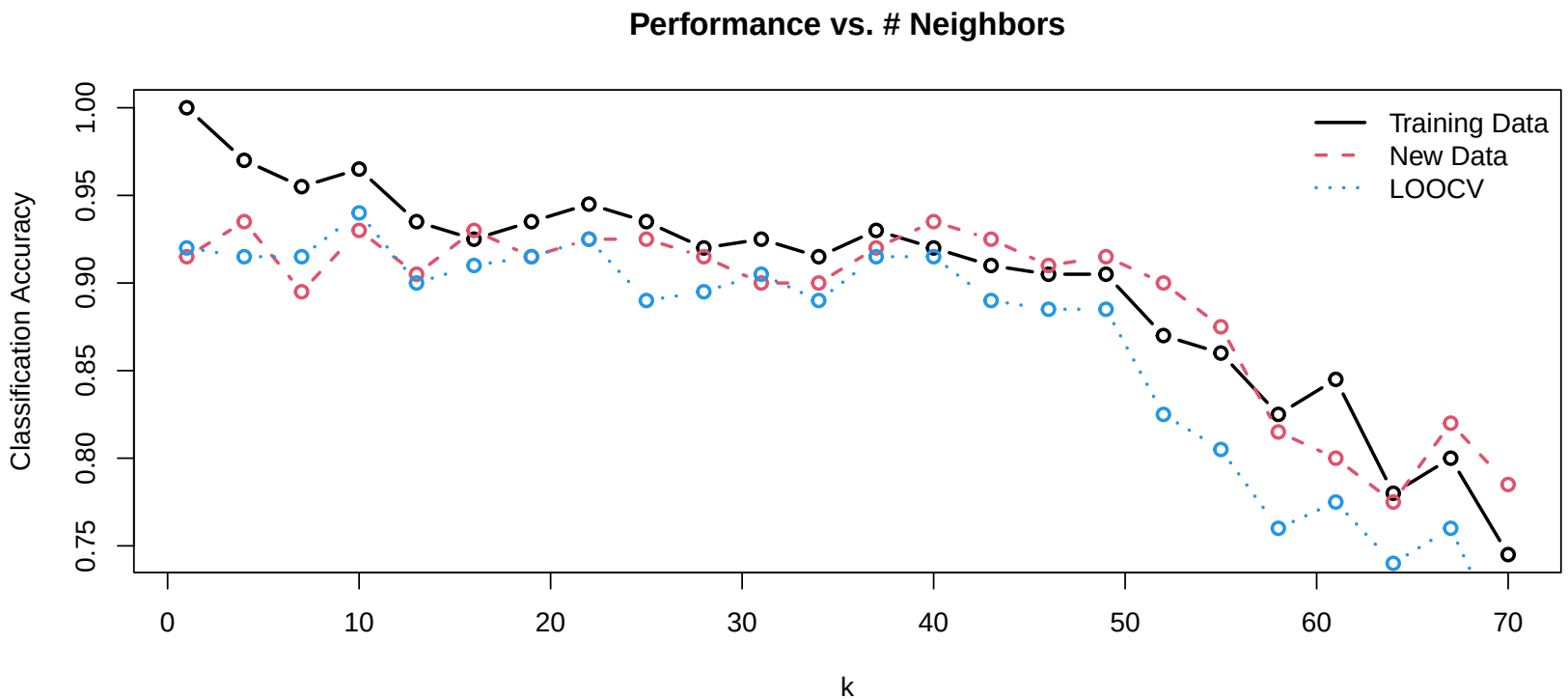
Cross-validation (CV) is a general term describing methods of repeated data-splitting used to evaluate the performance of a model on data that was not used in the training process:

1. Non-exhaustive cross-validation, notably $k$-fold cross-validation
2. Exhaustive cross-validation, notably leave-one-out cross-validation (LOOCV)

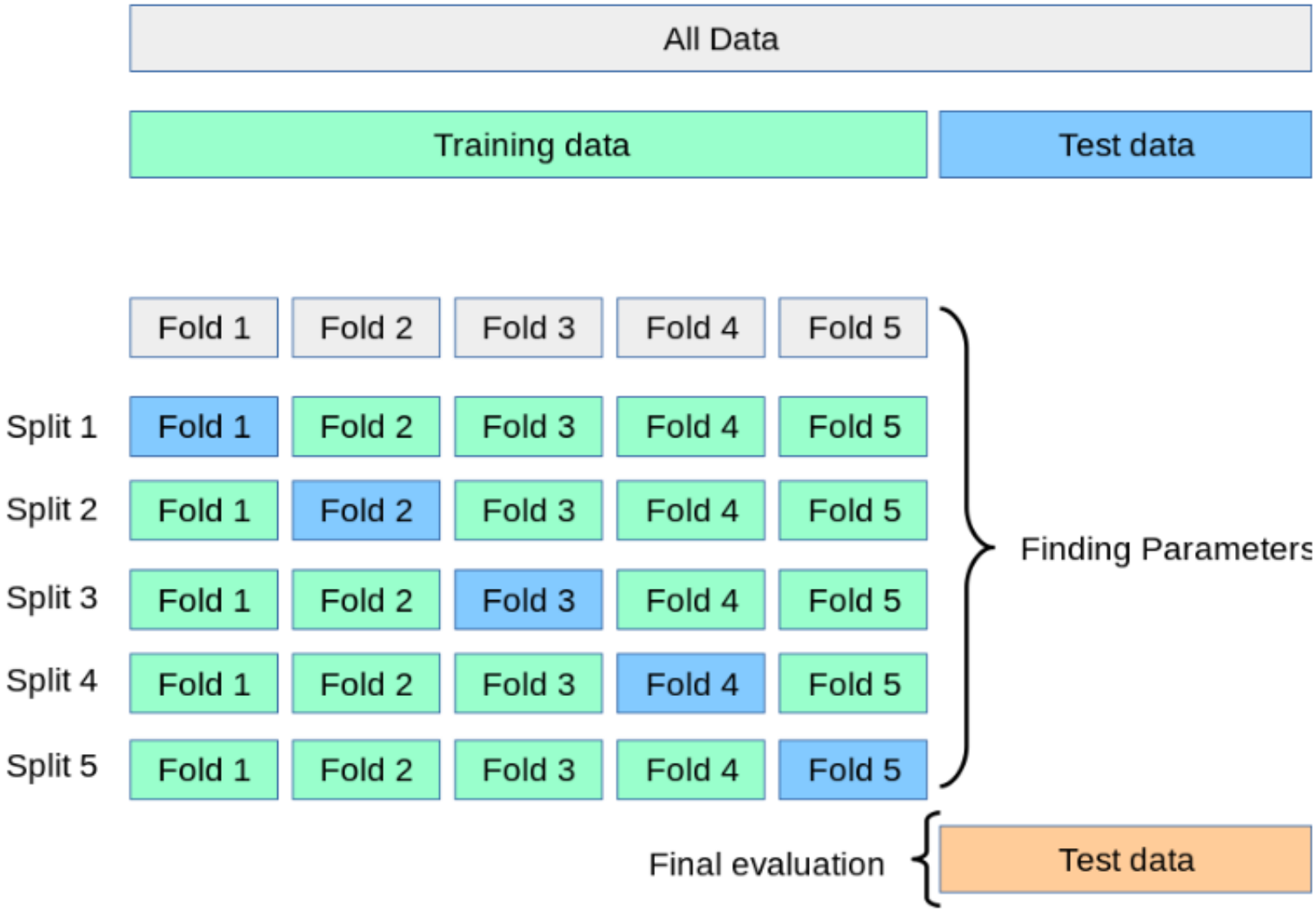
Non-exhaustive CV approaches involve randomness, so they are sometimes applied repeated (we'll generally focus on a single use).

# Example - LOOCV

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



# 5-fold Cross-validation



# $k$-fold Cross-validation

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)
   perform[fold_id == k] = model.predict(eval_X, eval_y)

## Calculate performance
score(perform)
```

# $k$-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 source will show a greater range of performance estimates
  - Errors are highly correlated in LOOCV due to the high degree of overlap in each training segment

# $k$-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 source will show a greater range of performance estimates
  - Errors are highly correlated in LOOCV due to the high degree of overlap in each training segment
- $k$-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)

# How many folds?

The bias-variance trade-off governs how the number of folds impacts the reliability of performance estimates:

- More folds allows larger amounts of data to be used during model training, which should reduce bias
  - However, too many folds will lead to highly correlated training errors, which will increase variance
- Conversely, too few folds can also lead to high variance estimates if model training is to the fold assignments and repeated CV is not used

Most practitioners favor 10-fold or 5-fold CV, with deep learning applications sometimes using smaller numbers of folds, such as $k = 3$, for computational reasons.

# 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 tuning parameters (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 tuning parameters (ie: finding the best number of neighbors)
- **Grid search** is a simple (and widely used) approach for finding optimal combinations of tuning parameters using via cross-validation
  - The idea is to systematically and exhaustively search a grid spanning the parameter space

# Grid Search (example)

Here's an example parameter grid for kNN that considers $k \in \{3, 4, 5\}$, 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 |

# Randomized Search

- Grid search can be computationally expensive, especially when you'd like to tune over a broad range of values for several different tuning parameters
- **Randomized search** is an alternative method that allows you specify distributions to be randomly sampled from for each tuning parameter
  - For kNN, you might specify $k$ as being sampled from a Poisson distribution with $\mu = \sqrt{n}$ (which is generally a decent starting point for $k$)

# 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

# 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
  - We will not cover this method, but you may consider using it on your final project