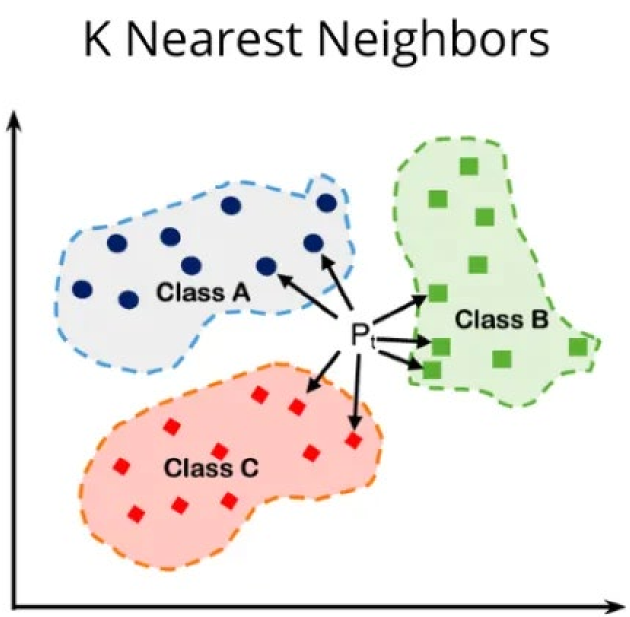
Local Averaging; Model Selection

Announcements

- Exam 1 on Thursday during the lecture period
- Please try to be in your seats by 1:30
- Please bring a phone or other device to scan an upload your exam to gradescope (this will make grading and the process of requesting regrades much easier)
- You are allowed one cheat sheet: One piece of standard printer paper, front and back, handwritten by you (printing what you write on a tablet is fine). **Please put your name on your cheat sheet**.
- Exam duration: 70 minutes
- Practice problems and solutions on Canvas

Nearest Neighbor Classification

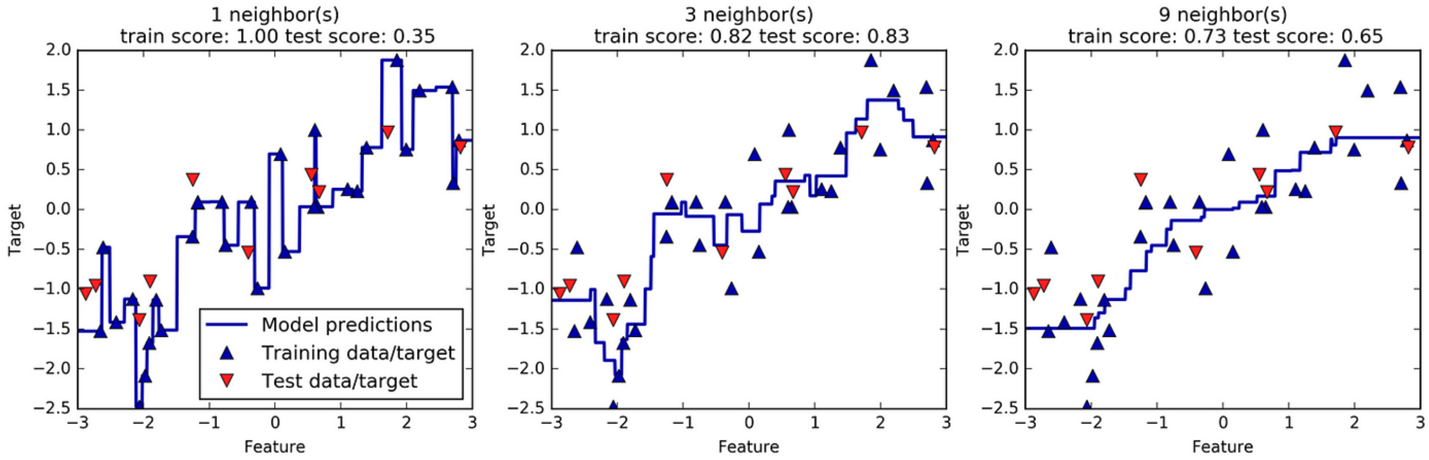
- **Nearest-neighbor classifier:** Assign $\boldsymbol{x}$ the same label as the closest training instance $\boldsymbol{x}_i$
- **k-nearest-neighbors classifier:** Assign a label to $\boldsymbol{x}$ by taking the most common training label among the k closest training instances $\boldsymbol{x}_i$



https://medium.com/@sachinsoni600517/k-nearest-neighbours-introduction-to-machine-learning-algorithms-9dbc9d9fb3b2

Nearest Neighbors Regression

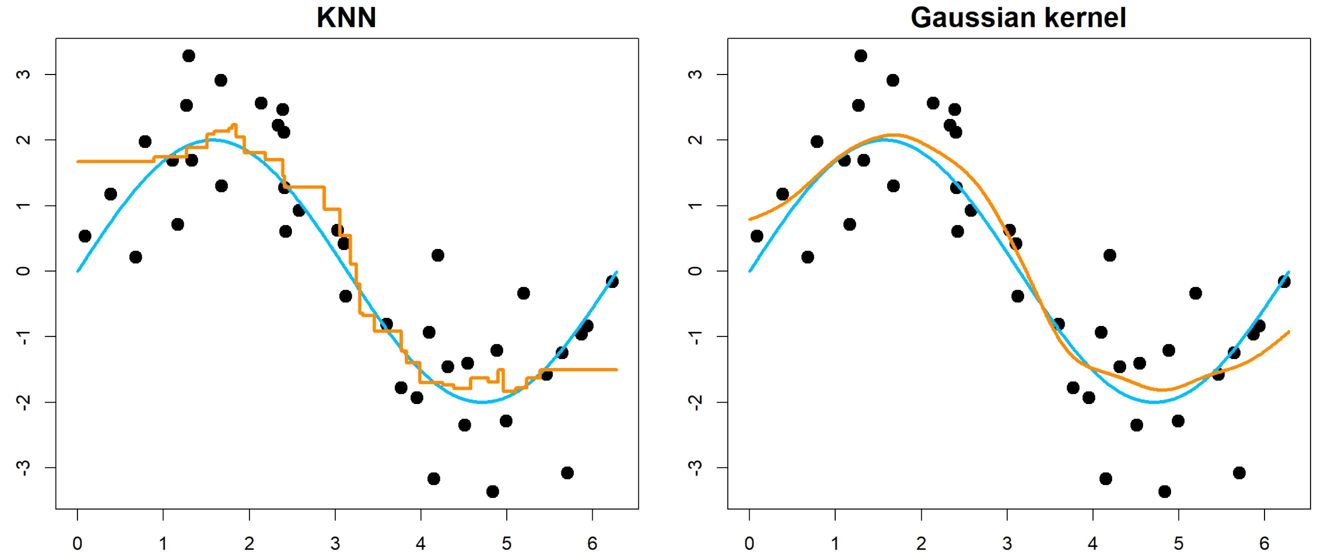
- k **nearest neigbbors regression:** Assign a response to $\boldsymbol{x}$ by taking the average training response over the k closest training instances $\boldsymbol{x}_i$



https://medium.com/analytics-vidhya/k-neighbors-regression-analysis-in-python-61532d56d8e4

Kernel Smoothing

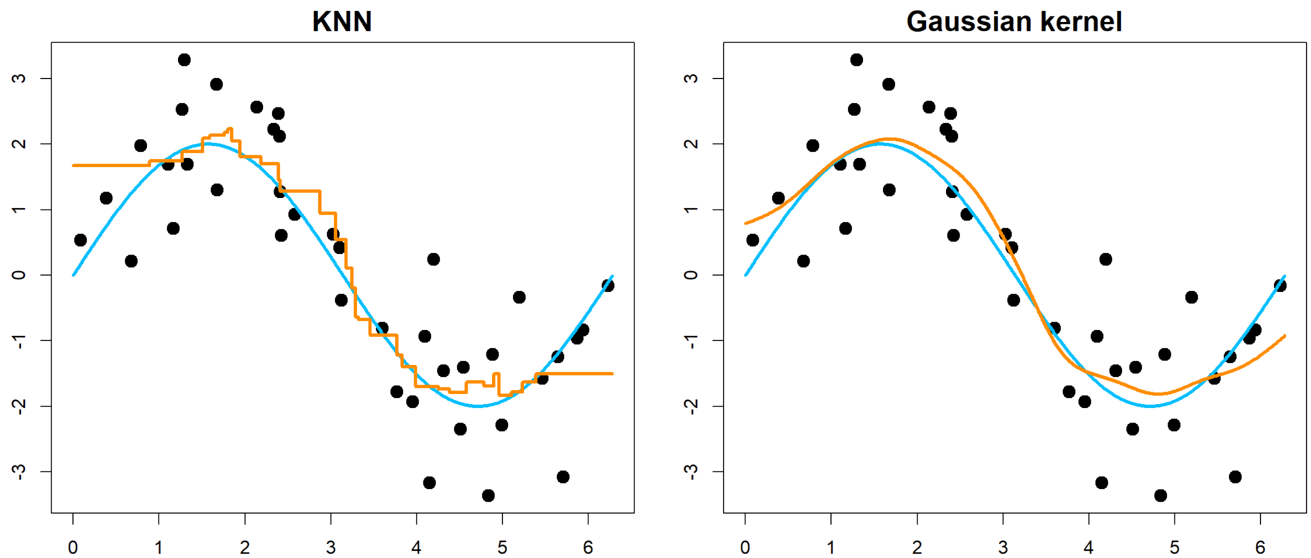
- Regression setting with training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$
- A *smoothing kernel* is a function $k(\boldsymbol{x}, \boldsymbol{x}')$ that assigns a larger value the more similar $\boldsymbol{x}$ and $\boldsymbol{x}'$ are
- Example: Gaussian smoothing kernel:
- Kernel smoothing estimate:



https://teazrq.github.io/SMLR/kernel-smoothing.html

Kernel Smoothing

- Regression setting with training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$
- A *smoothing kernel* is a function $k(\boldsymbol{x}, \boldsymbol{x}')$ that assigns a larger value the more similar $\boldsymbol{x}$ and $\boldsymbol{x}'$ are
- Example: Gaussian smoothing kernel:
- Kernel smoothing estimate:



https://teazrq.github.io/SMLR/kernel-smoothing.html

Partition-Based Prediction

- Classification setting with training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$
- Let $A_1, A_2, \ldots, A_M$ be a partition of the feature space
- Define

$$A(x) :=$$

- Classification: Given a test point $\boldsymbol{x}$ predict
- Regression: Given a test point $\boldsymbol{x}$ predict
- How to determine partition?

Local Smoothing Interpretation

- All three approaches have a common form. Consider regression. Given a test point $\boldsymbol{x}$ the prediction is

$$\sum_{i=1}^{n} w_i(\boldsymbol{x}) y_i$$

where, for each $\boldsymbol{x}$, the weights $w_i(\boldsymbol{x})$ are nonnegative and sum to 1.

Plug-In Interpretation

- All three approaches can also be viewed as plug-in estimates.
- For example, consider the partition-based approach to classification
- Recall the Bayes classifier formula

$$f^*(\boldsymbol{x}) = \arg\max_k \ \pi_k g_k(\boldsymbol{x})$$

where $g_k(\boldsymbol{x})$ is the class-conditional density of $\boldsymbol{x}$ given $y = k$.

- We can estimate g_k using the
- The corresponding plug-in method is precisely the partition-based classifier described earlier (exercise)
- Similar arguments can be made for the other local averaging methods

Model Selection

The Word "Model" in ML

- The word "model" is used in many different ways in ML. It can refer to
  - An assumption about the joint distribution of data (e.g., logistic regression model)
  - An assumption about the form of a function (e.g., a linear model)
  - The output of an ML algorithm (e.g., "let's apply the learned model to test data")
  - A machine learning algorithm with a certain choice of hyperparameters (as in "model selection", today's topic)
- The word "model" is never necessary, but sometimes convenient. The intended usage can be inferred from context with enough experience.

Poll

- Many ML algorithms have *hyperparameters*. These are parameters that are not determined by the learning algorithm.

- Which of the following is NOT a hyperparameter

  $k(x,x') = \exp\left(-\frac{1}{2\sigma^2}\|x-x'\|^2\right)$

  (A) k in k-nearest neighbors classification

  (B) the regularization parameter λ in ridge regression

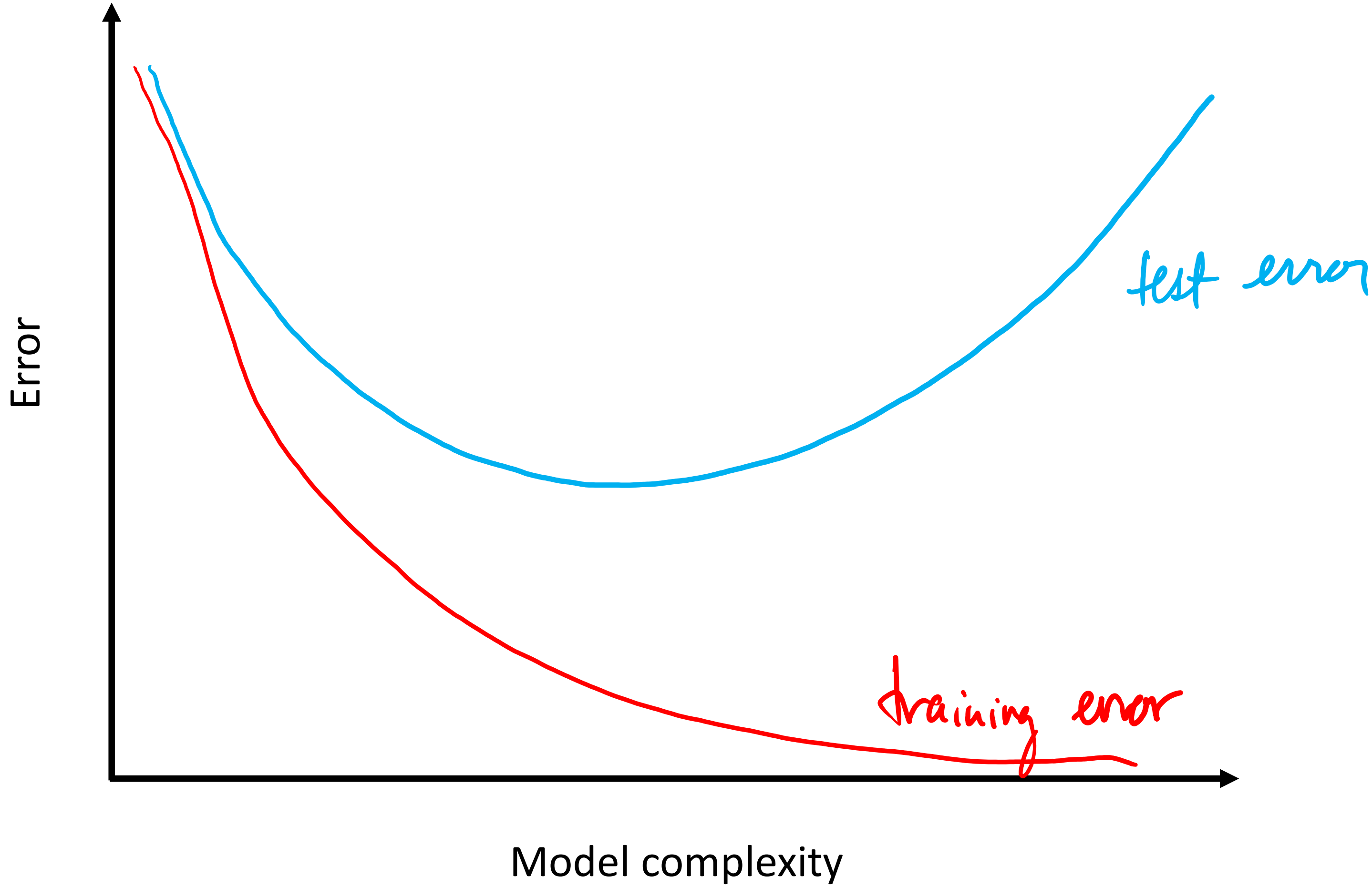
  (C) the bandwidth σ of a Gaussian kernel

  (D) the number of support vectors of an SVM

- In most cases, hyperparameters affect model complexity

- Therefore, we must take care to avoid overfitting

Error vs. Model Complexity: The Conventional Wisdom



(there can be exceptions to this behavior)

Train/Test Split

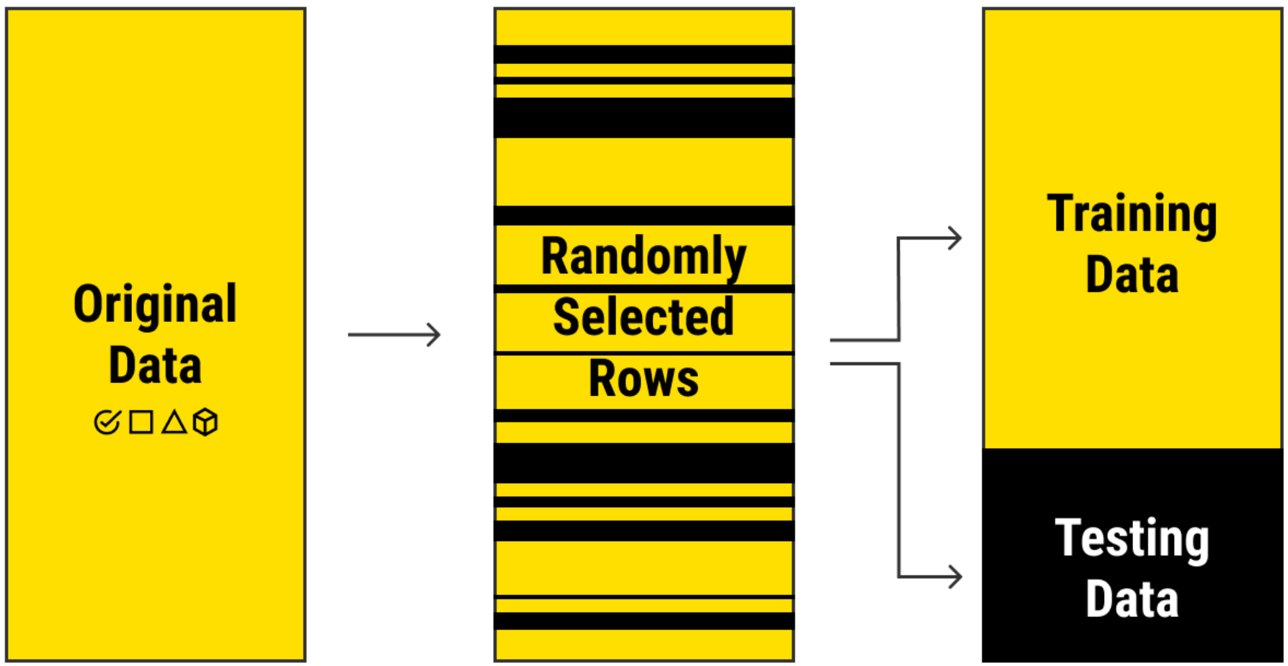
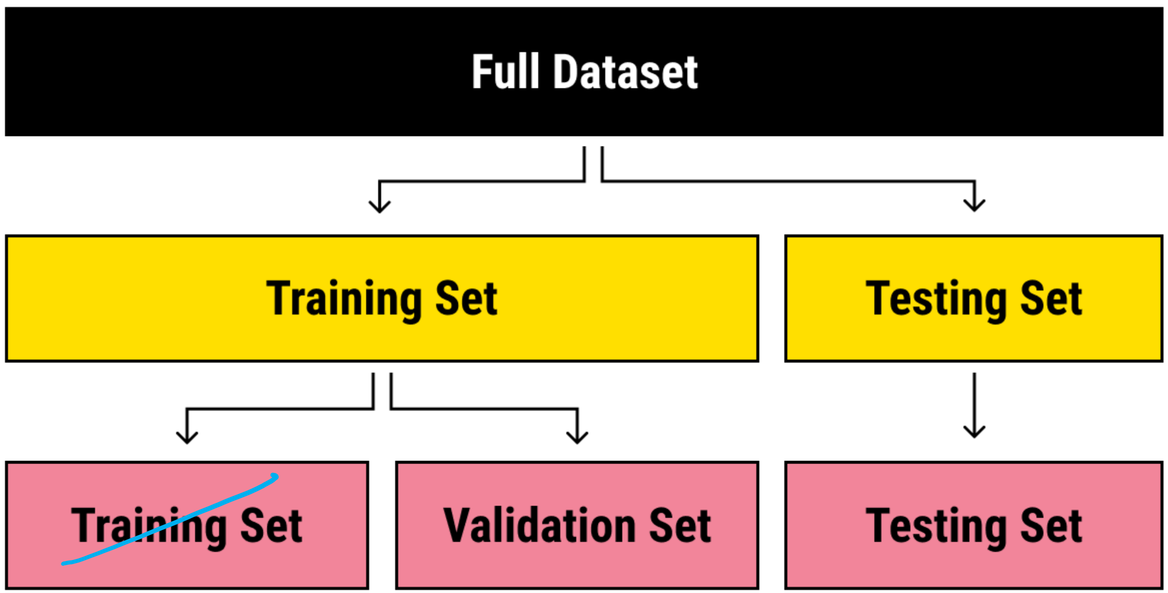


Figure from: https://labelyourdata.com/articles/machine-learning-and-training-data

Holdout Error Estimation



Model fitting (run the algorithm)

Model Selection

Ridge regression: $\lambda \in \{10^{-2}, 10^{-1}, 1, 10, 10^{2}\}$

Figure from: https://labelyourdata.com/articles/machine-learning-and-training-data

K - fold

K=5

Cross Validation

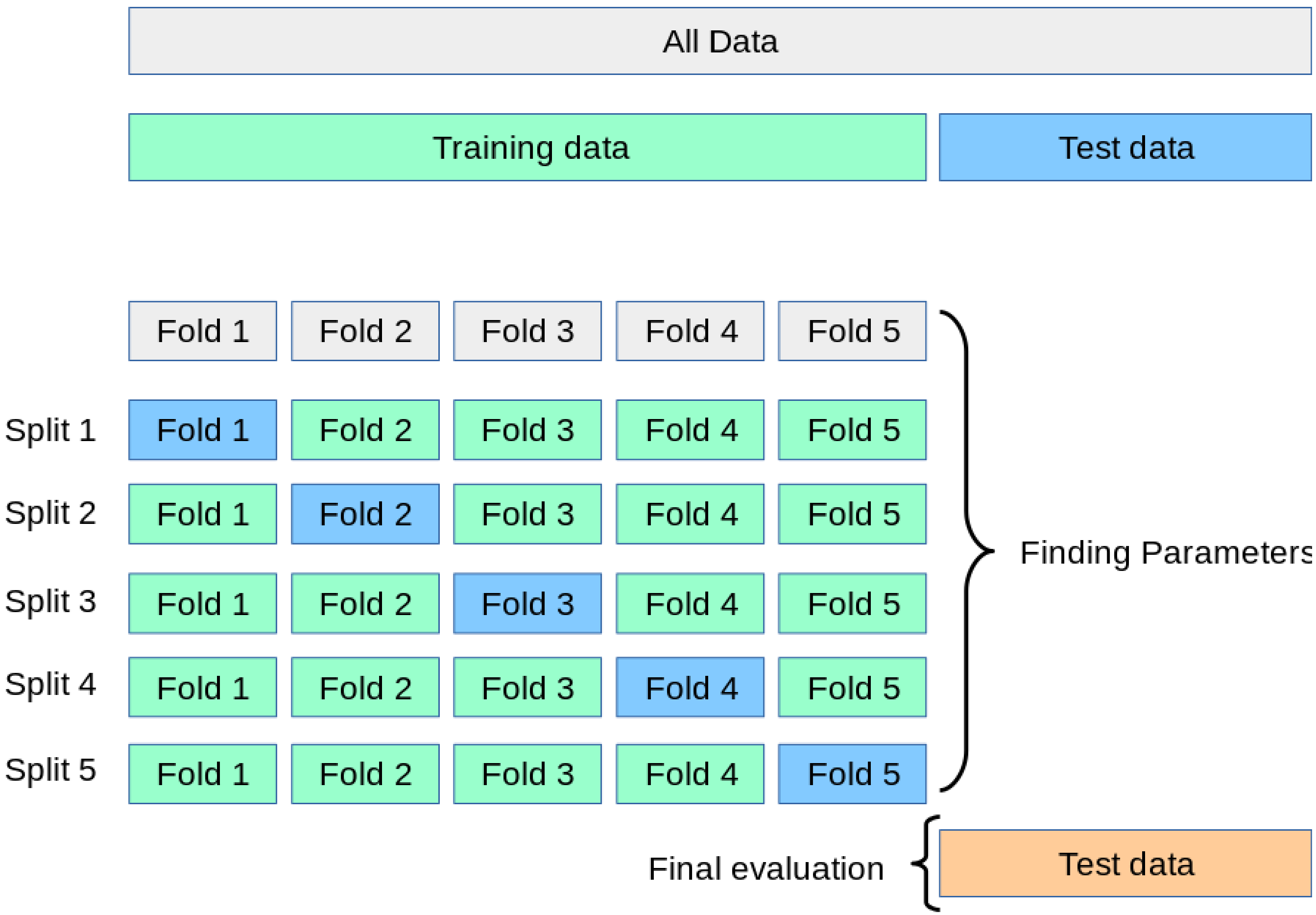
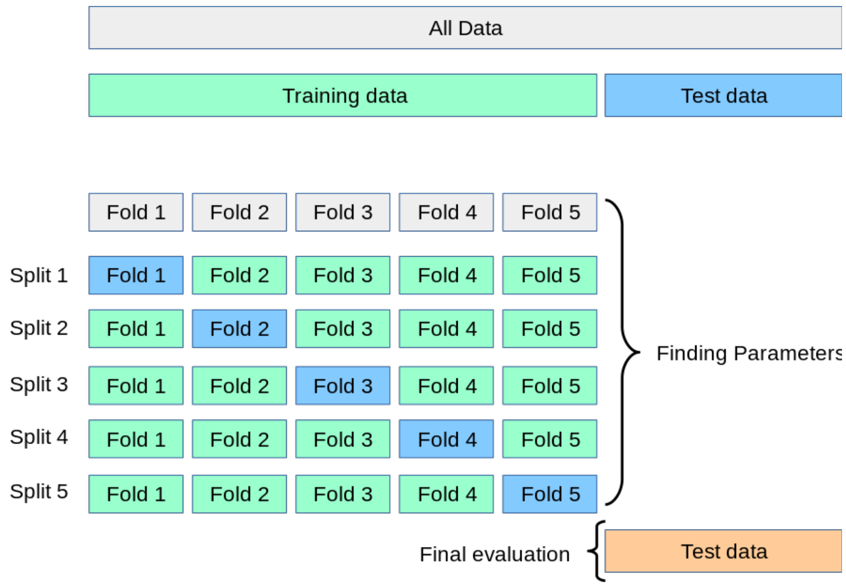


Figure from: https://scikit-learn.org/stable/_images/grid_search_cross_validation.png

Cross Validation

- Determine a finite set of hyperparameters a priori
- For each split, and for each hyperparameter under consideration, fit the model on the data that is not held out
- Average the holdout error estimates to get the cross-validation error estimate for each hyperparameter
- Select the hyperparameter with smallest CV error estimate



Remarks

- Common choices of K: $5, 10$, and n → leave-one-out CV
- In CV, after selecting tuning parameters, re-run the algorithm with the selected parameters on the full training data to get the final model
- In classification, the folds should be chosen so that the proportions of different classes in each fold are the same as in the full sample. This is known as *stratified* cross-validation.
- Mathematical formulation in my notes
- Alternative to holdout and CV: Bootstrap error estimation (also in my notes)

Poll

True or false: Ideally, the test data should never be used to tune parameters

(A) True

(B) False