k-Nearest Neighbors

Ivan Corneillet

Data Scientist

GENERAL ASSEMBLY

Learning Objectives

After this lesson, you should be able to:

- Define class label and classification
- Build a k-Nearest Neighbors using *sklearn*
- Evaluate and tune model by using metrics such as classification accuracy/error

Announcements and Exit Tickets

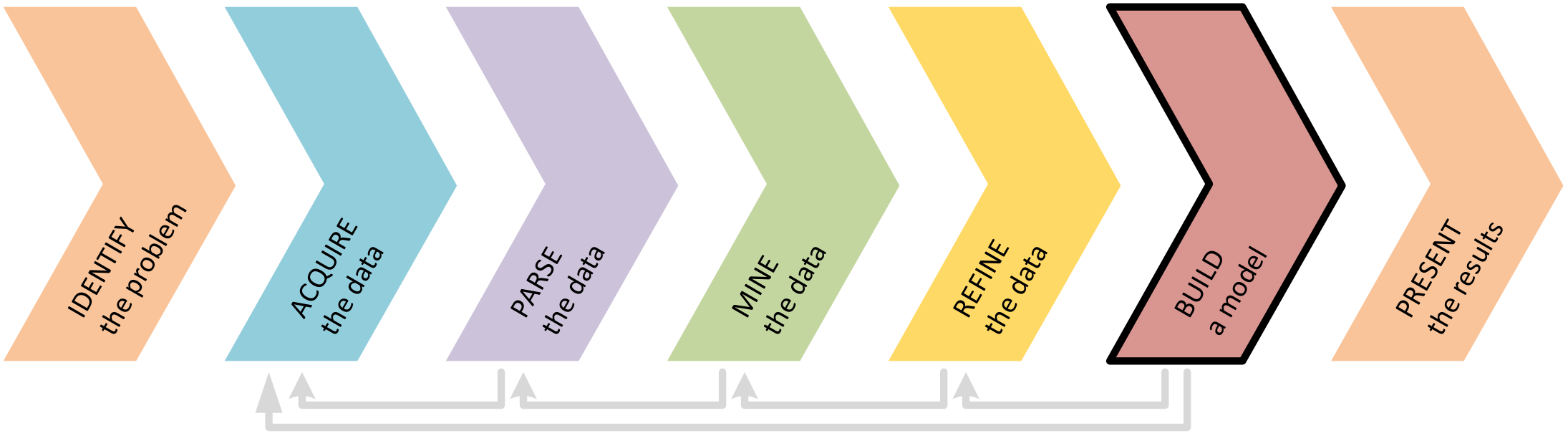
Review

Today

Today, we are introducing what classification is and what classification models are

Research Design and Data Analysis	Research Design	Data Visualization in *pandas*	Statistics	Exploratory Data Analysis in *pandas*
Foundations of Modeling	Linear Regression	Classification Models	Evaluating Model Fit	Presenting Insights from Data Models
Data Science in the Real World	Decision Trees and Random Forests	Time Series Models	Natural Language Processing	Databases

Today, we keep our focus on the ❻ **BUILD** a model step but with a focus on classification



Here's what's happening today:

- Announcements and Exit Tickets
- Review
- ❻ Build a Model | k-Nearest Neighbors
  - Types of machine learning problems
  - What's classification; what's binary classification?
  - Classification vs. regression
  - Iris dataset
    - Exploratory data analysis
    - Hand-coded classifiers
  - Classification metrics
  - k-Nearest Neighbors
    - High dimensionality
    - What's the best value for k?
  - Validation and cross-validation
- Lab – k-Nearest Neighbors
- Review
- Exit Tickets

❻ Build a Model

Types of Machine Learning Problems

Types of Machine Learning Problems

	Continuous	Categorical
Supervised (a.k.a., predictive modeling)	Linear Regression (sessions 6 and 7)	k-Nearest Neighbors (session 8) Logistic Regression (session 9)
Unsupervised	A machine learning model that doesn't use labeled data is called unsupervised. It extracts structure from the data. Goal is "representation"	

❻ Build a Model

What's Classification and what's Binary Classification?

What's classification?

- Classification is a machine learning problem for solving a set of categorical values (y; the response vector) given the knowledge we have about these values (X; the feature matrix)
  - E.g., what if you are predicting whether an image is of a *human*, *dog*, or *cat*?
- The possible values of the response variable are called *class labels*
  - E.g., *"human"*, *"dog"*, and *"cat"*

What's binary classification?

- Binary classification is the simplest form of classification
  - I.e., the response is a *boolean* value (true/false)
- Many classification problems are binary in nature
  - E.g., we may be using patient data (medical history) to predict whether a patient smokes or not

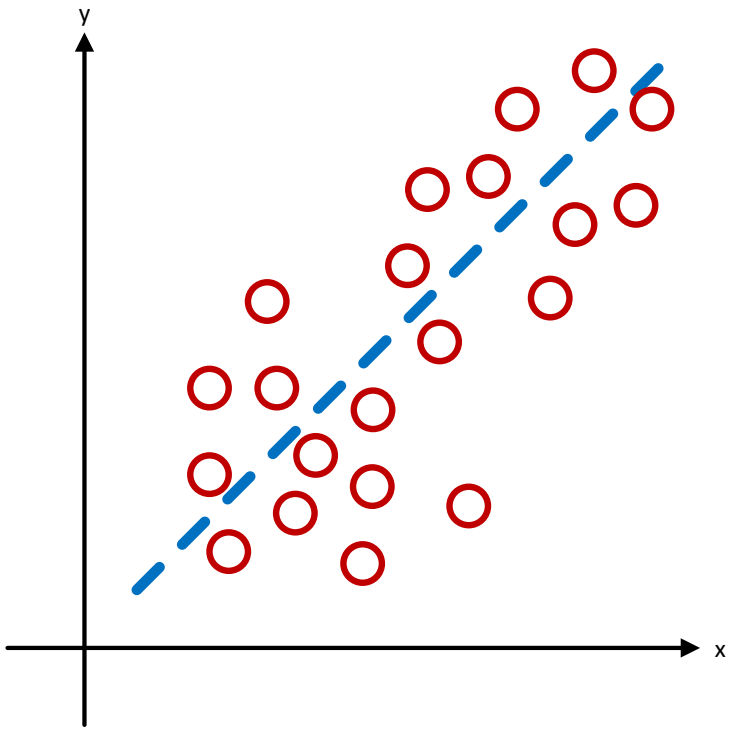
- At first, many problems don't appear to be binary; however, you can usually transform them into binary problems
  - E.g., what if you are predicting whether an image is of a *"human"*, *"dog"*, or *"cat"*?
  - You can transform this non-binary problem into three binary problems
    - 1. Will it be *"human"* or *"not human"*?
    - 2. Will it be *"dog"* or *"not dog"*?
    - 2. Will it be *"cat"* or *"not cat"*?
- This is similar to the concept of binary variables

❻ Build a Model

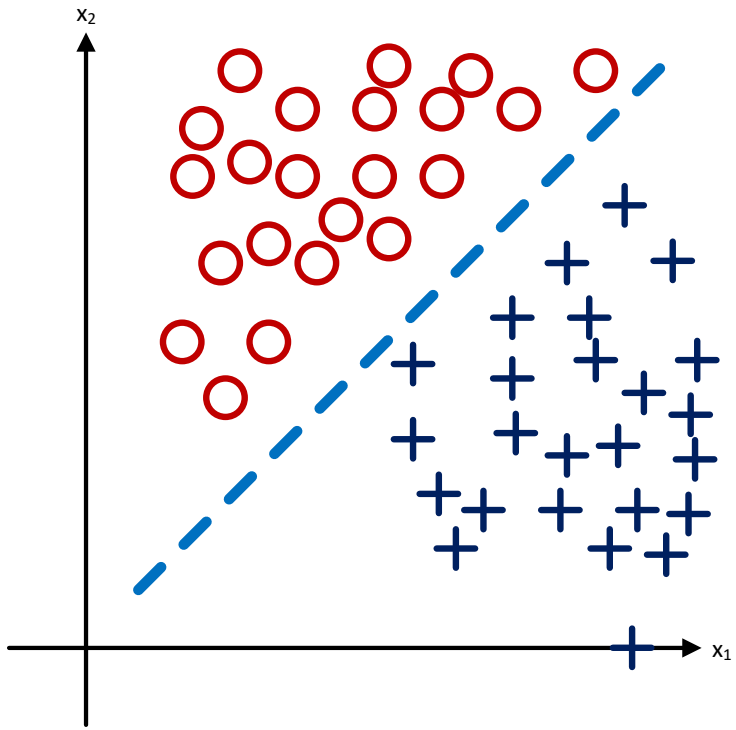
Classification vs. Regression

Classification and regression differ in what they are trying to predict

Regression



Classification



Iris Dataset

The *Iris* dataset contains 3 classes of 50 instances each, each class referencing a type of iris plant (*Setosa*, *Versicolor*, and *Virginica*)

Iris Setosa

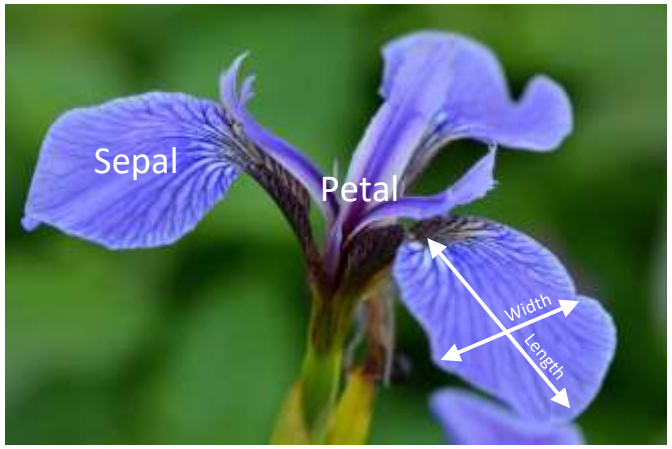
Iris Versicolor

Iris Virginica

Source: Flickr

Iris dataset (cont.)

- Can we teach a machine to identify the type of iris based on the following four attributes?
  - Sepal length and width
  - Petal length and width



Source: Flickr

Iris Dataset

Activity & Codealong – Part A

Exploratory Data Analysis

Activity | Iris Dataset | Exploratory Data Analysis

EXERCISE

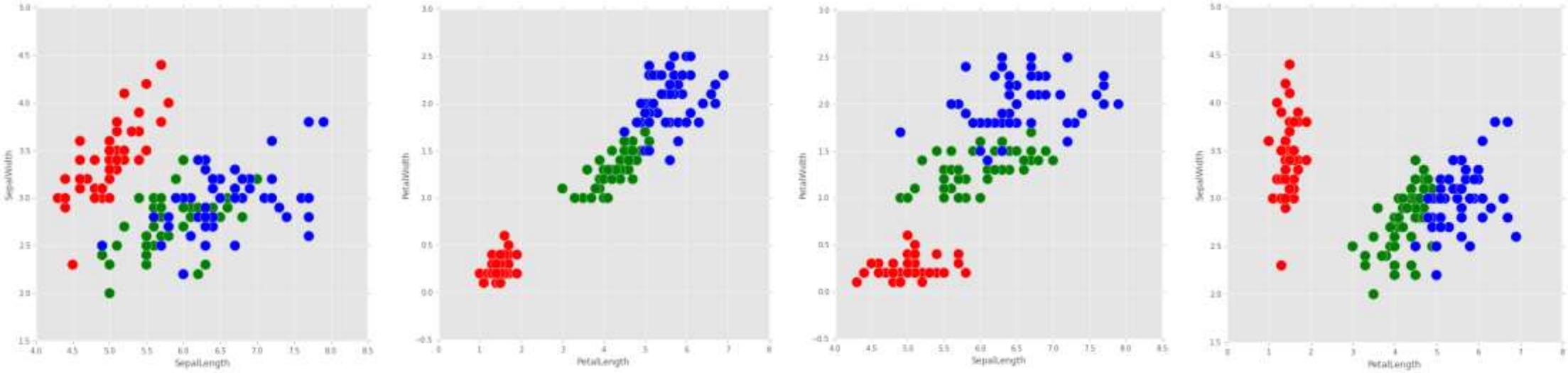
DIRECTIONS (10 minutes)

1. Using the Iris dataset (`iris.csv` in the `datasets` folder), perform exploratory analysis between *SepalLength*, *SepalWidth*, *PetalLength*, and *PetalWidth* (the *feature* variables) and *Species* (the *class* variable). How can you use these features to separate one species from the other two?
2. When finished, share your answers with your table

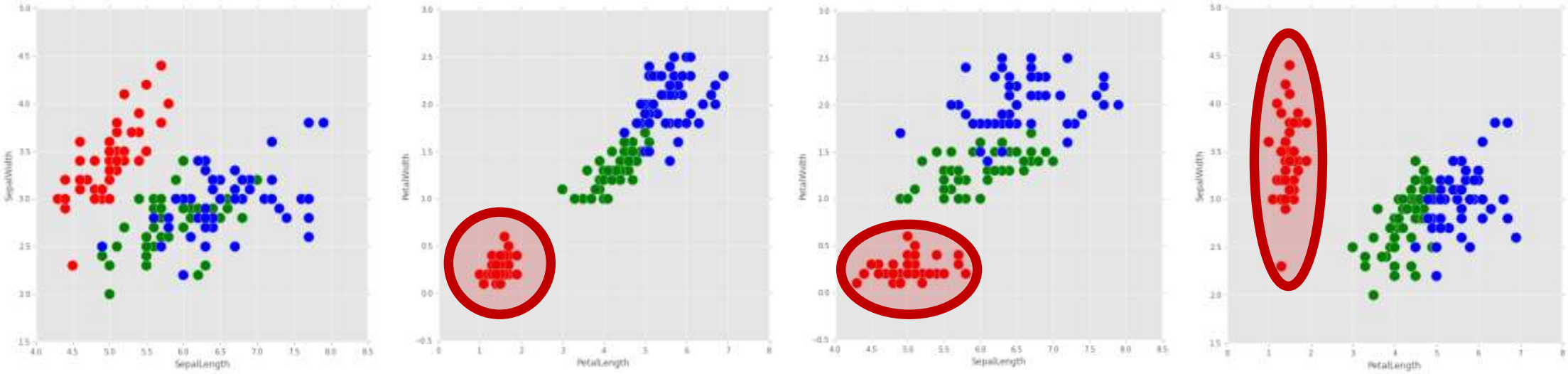
DELIVERABLE

Answers to the above questions

Activity | Iris Dataset | Exploratory Data Analysis (cont.)



Iris Dataset | The *Setosa* class (in red) is linearly separable from the other two (*Versicolor* in green and *Virginica* in blue)



Iris Dataset

Activity & Codealong – Part B

First Hand-Coded Classifier

Activity | Iris Dataset | First hand-coded classifier

EXERCISE

DIRECTIONS (10 minutes)

1. Using the Exploratory Data Analysis, write a first hand-coded classifier to separate Setosa (return '`Setosa`') from Virginica and Versicolor (always return one or the other). How would you measure how good is your classifier?
2. When finished, share your answers with your table

DELIVERABLE

Answers to the above questions

❻ Build a Model

Classification Metrics

The metrics we've used for regressions do not apply for classification

- We could measure distance between the probability of a given class and an item being in the class. E.g., guessing .6 for a 1 is a .4 error, while guessing .99 for 1 is .01 error…
- but this overly complicates our current goal: understanding binary classifications, like whether something is right or wrong

Instead, let's start with two new metrics, which are inverses of each other: accuracy and misclassification rate

- Since they are opposite of each other, you can pick one or the other; effectively they will be the same. But when coding, do make sure that you are using a classification metric when solving a classification problem!
- *sklearn* will not intuitively understand if you are doing classification or regression, and accidentally using mean squared error for classification, or accuracy for regression, is a common programming pitfall

- Accuracy
  - How many observations that we predicted were correct? This is a value we'd want to increase (like R^2)

- Misclassification rate
  - Directly opposite of accuracy
  - Of all the observations we predicted, how many were incorrect? This is a value we'd want to decrease (like the mean squared error)

Iris Dataset

Codealong – Part C

Classification Metrics

Iris Dataset

Activity & Codealong – Part D

Second Hand-Coded Classifier

Activity | Iris Dataset | Second hand-coded classifier

EXERCISE

DIRECTIONS (10 minutes)

1. Improve the first hand-coded classifier to further separate the remaining classes of iris. How much better is this new classifier?
2. When finished, share your answers with your table

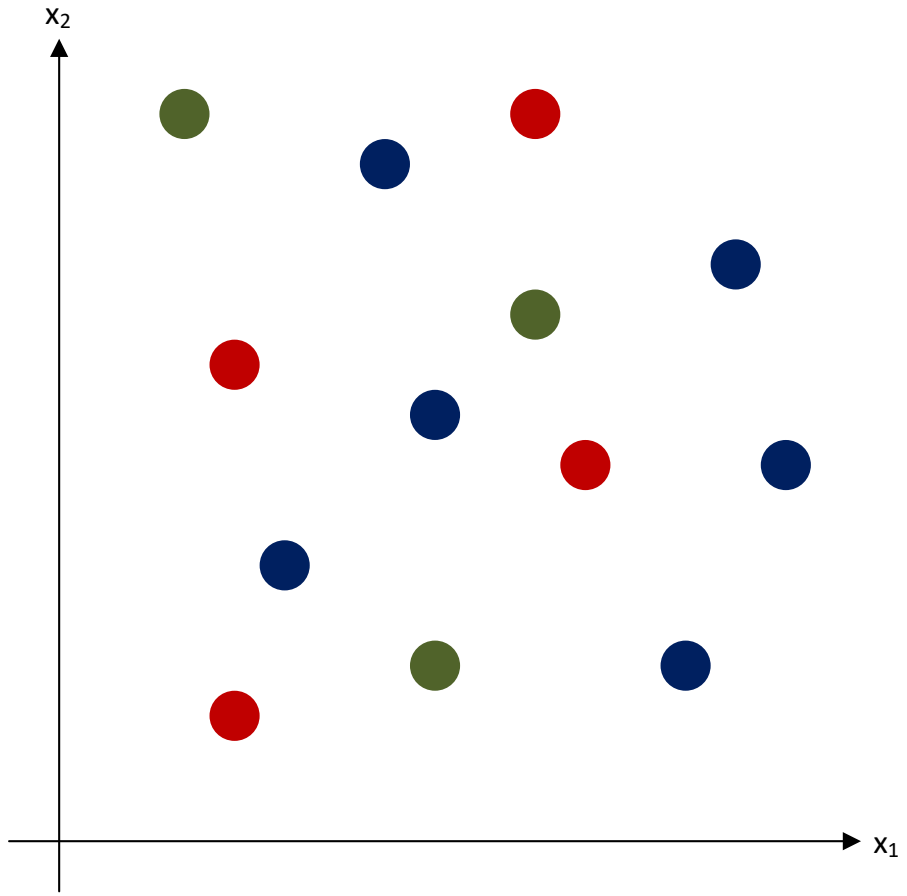
DELIVERABLE

Answers to the above questions

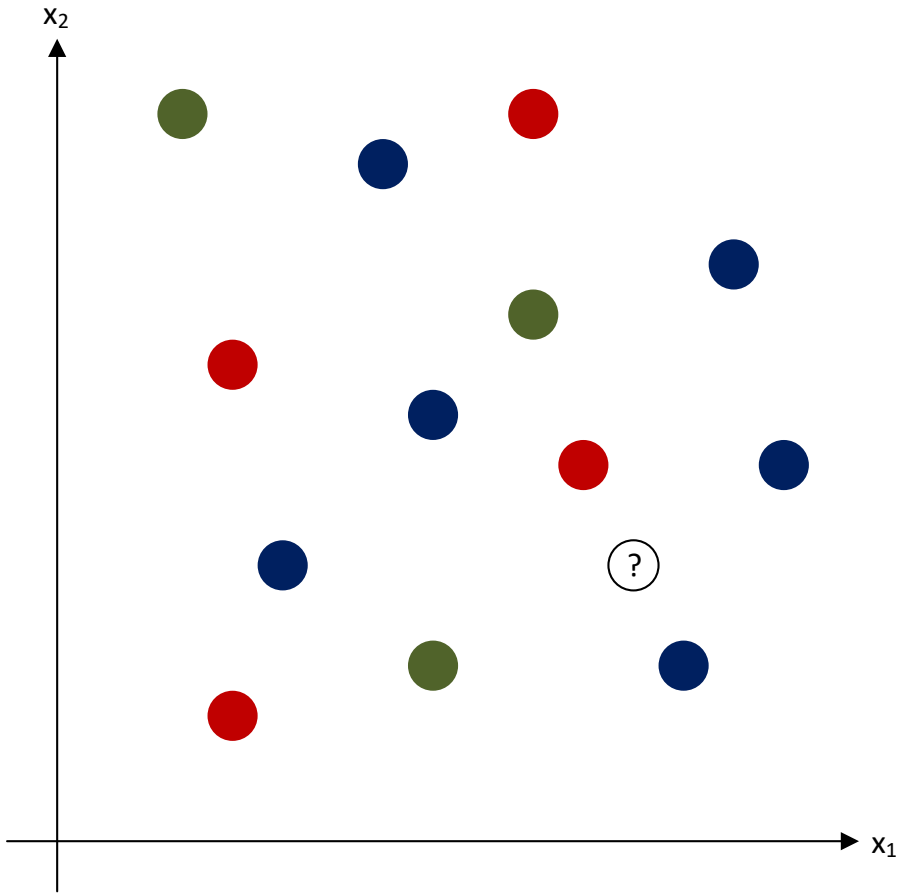
k-Nearest Neighbors

k-Nearest Neighbors

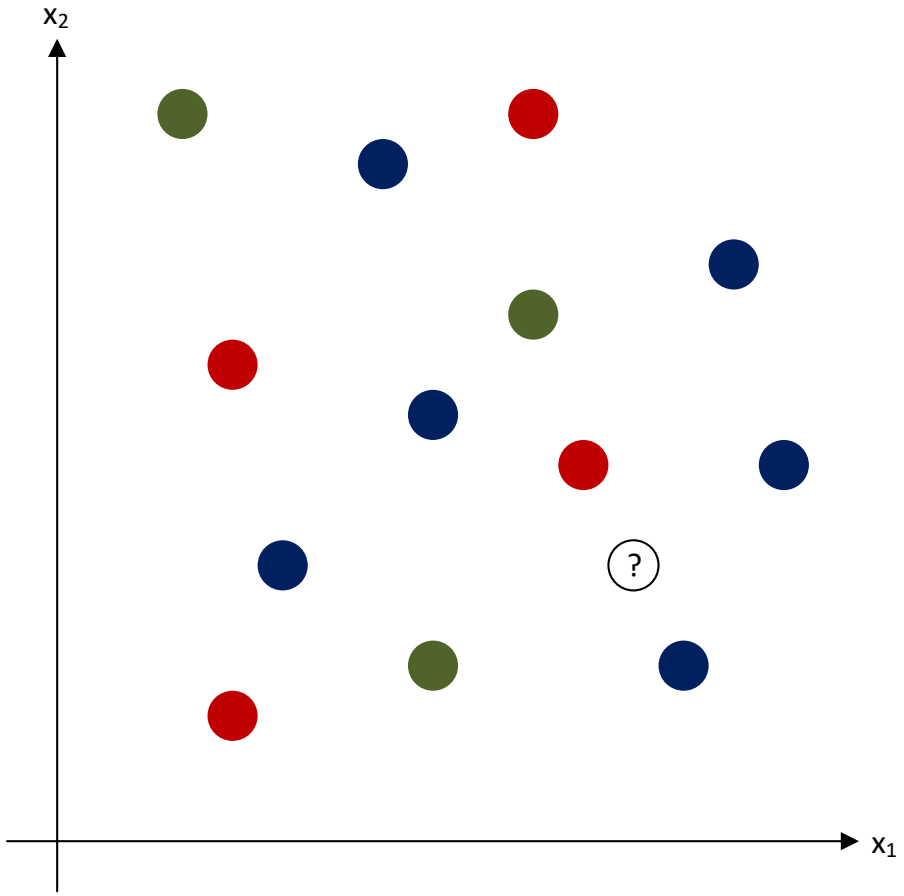
- k-Nearest Neighbors is a classification algorithm that makes a prediction based upon the closest data points



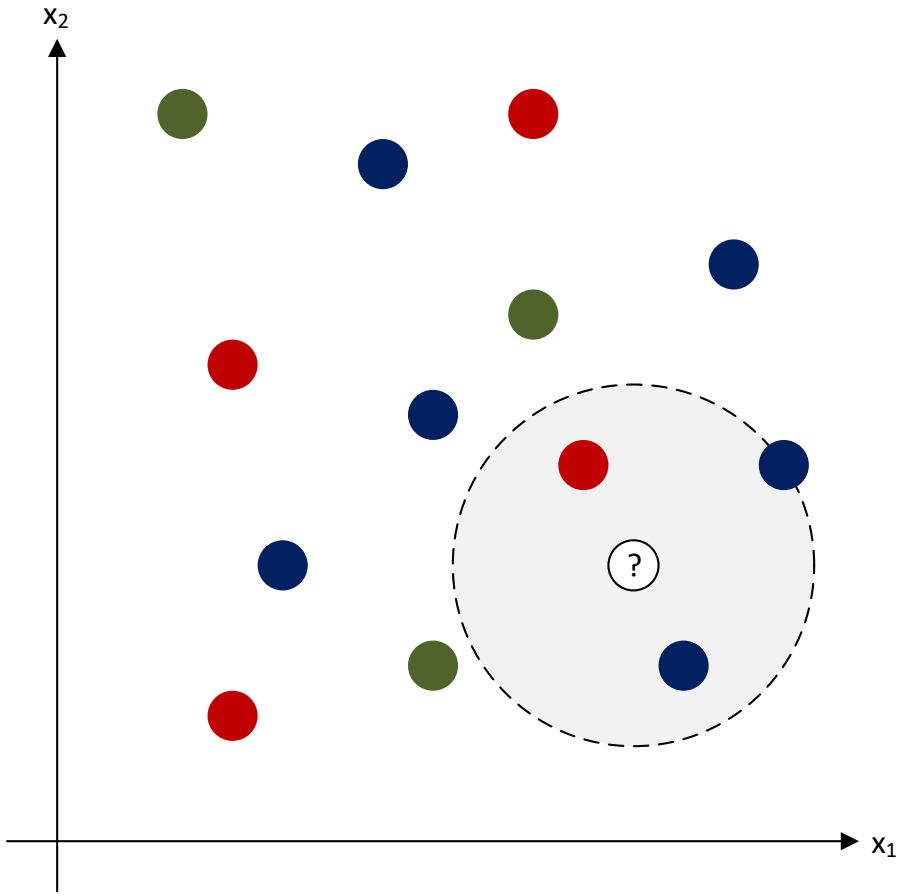
k-Nearest Neighbors | How would you predict the color of the “question mark” point?



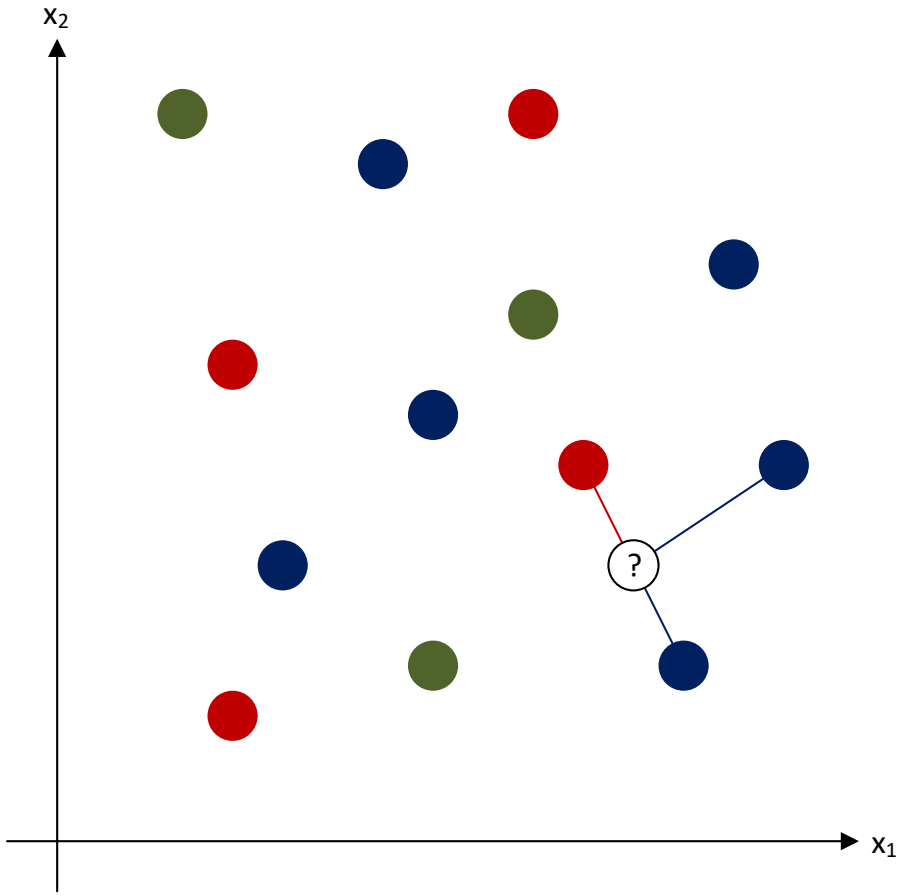
k-Nearest Neighbors | ❶ Pick a value for k, e.g., $k = 3$



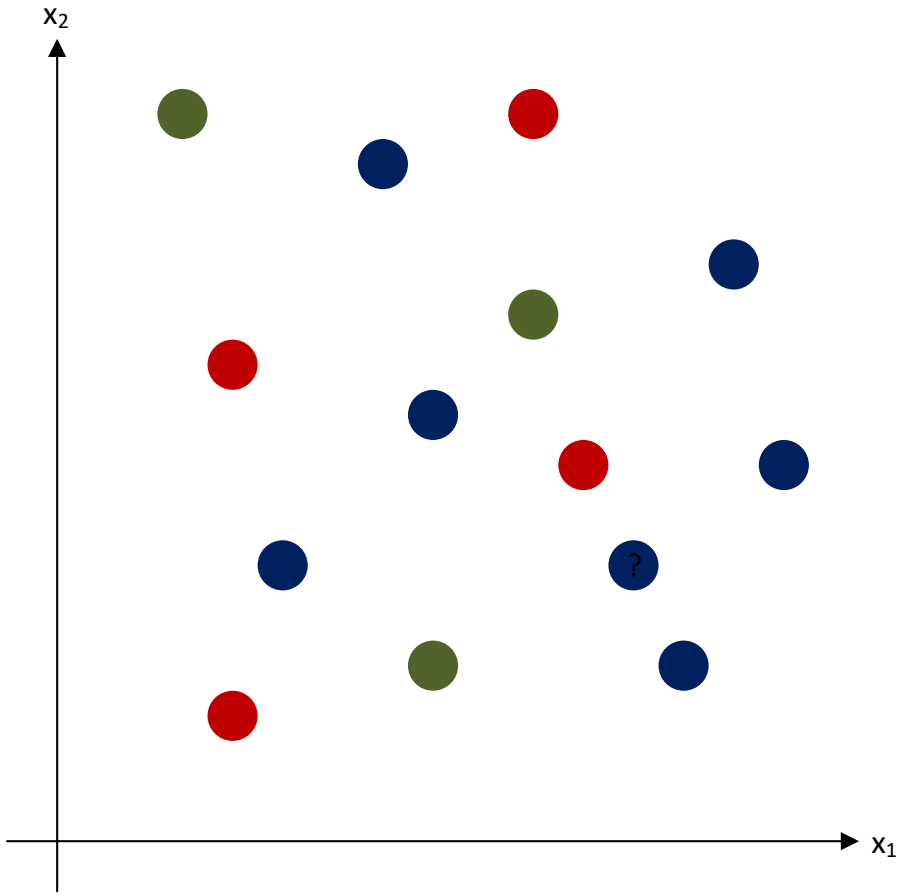
k-Nearest Neighbors | ❷ Calculate the distance to all other points; given those distances, pick the k closest points



k-Nearest Neighbors | ❸ Calculate the probabilities of each class label given those points: $\frac{1}{3}$ “red”, $\frac{2}{3}$ “blue”

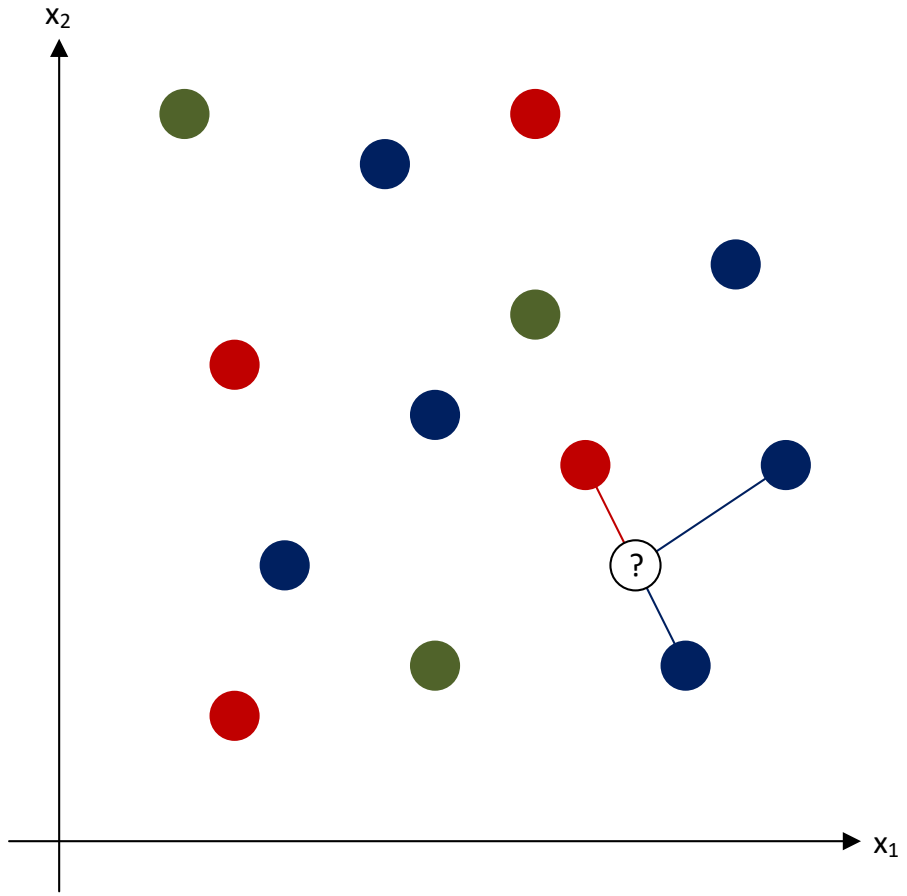


k-Nearest Neighbors | ❹ The original point is classified as the class label with the largest probability (“votes”): “blue”



k-Nearest Neighbors (cont.)

- k-Nearest Neighbors uses distance to predict a class label
- This application of distance is used as a measure of similarity between classifications
  - We are using shared traits to identify the most likely class label



Iris Dataset

Codealong – Part E

k-Nearest Neighbors

k-Nearest Neighbors | What happens if two classes get the same number of votes?

- *sklearn* will choose the class it first "saw" in the training set

- We could also implement a weight, taking into account the distance between a point and its neighbors
- This can be done in *sklearn* by changing the *weights* parameter to *'distance'*

k-Nearest Neighbors

High Dimensionality

k-Nearest Neighbors | What happens in high dimensionality?

- Since k-Nearest Neighbors works with distance, higher dimensionality of data (i.e., more features) requires significantly more samples in order to have the same predictive power
  - With more dimensions, all points slowly start averaging out to be equally distant; this causes significant issues for k-Nearest Neighbors

- Keep the feature space limited and k-Nearest Neighbors will do well; exclude extraneous features when using k-Nearest Neighbors

k-Nearest Neighbors

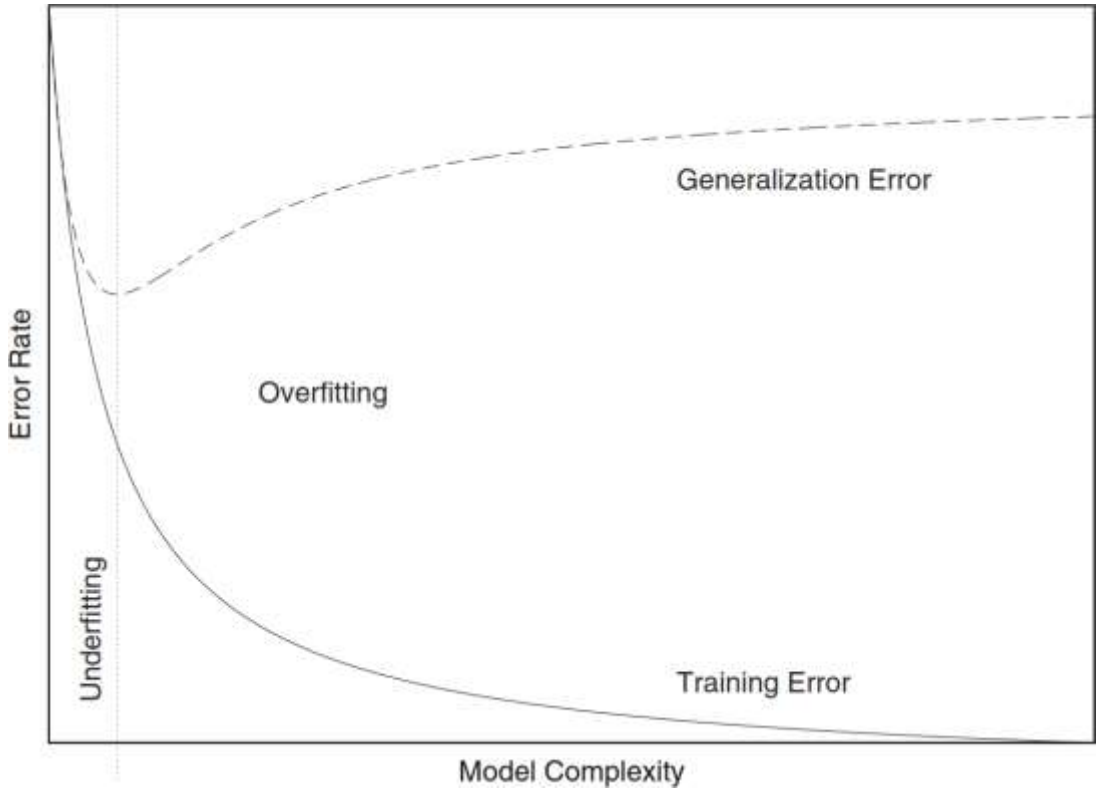
Codealong – Part F

What's the best value for k?

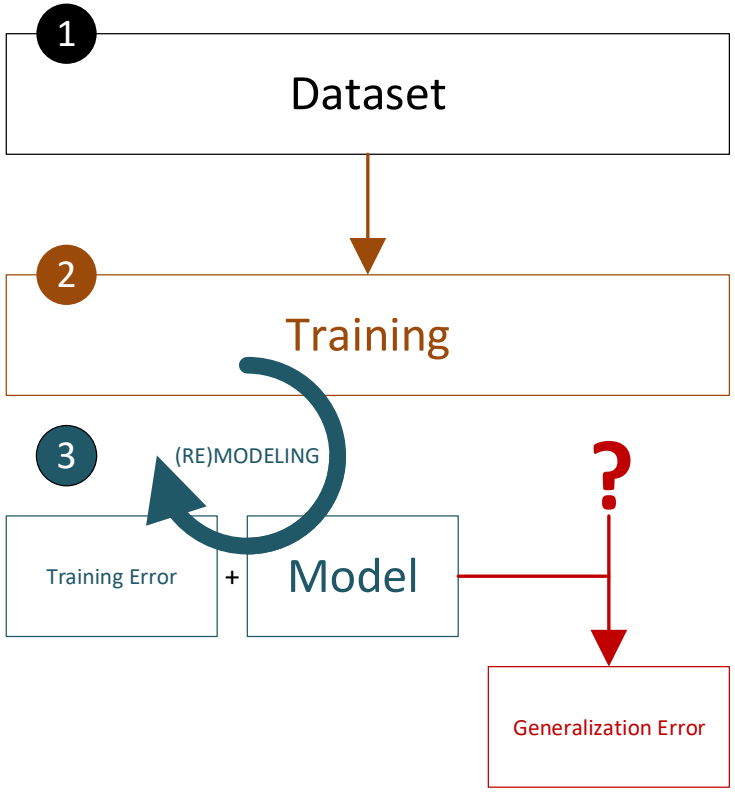
❻ Build a Model

Validation

So far, we used the entire dataset to train the models. How can we estimate the generalization error?

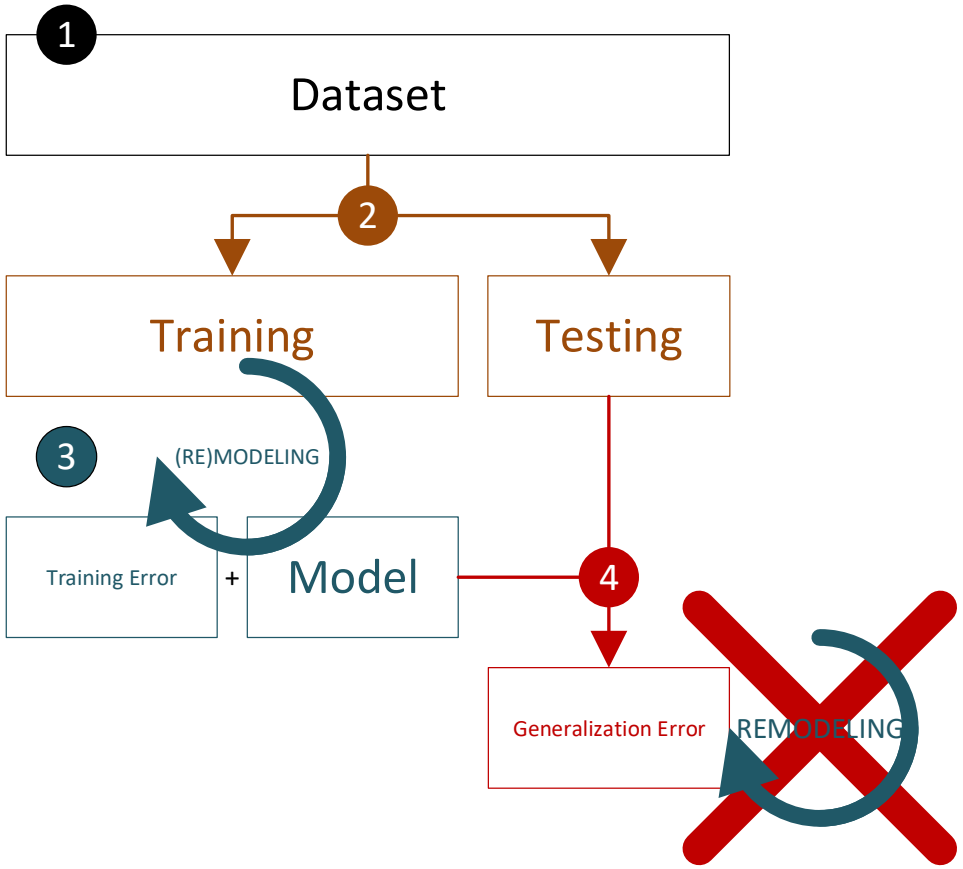


Source: Data Analysis with Open Source Tools



Validation is a possible answer

- Answer: (Randomly) divide the dataset into a training set and a testing set
  - Set aside the testing set; don't look at it
- Train the models with the training set
  - Compute the training set and remodel as needed
- Once you are happy with your model, use the testing set to compute the generalization error
  - But you cannot go back and remodel; otherwise these previously unknown data points are not longer unseen



Iris Dataset

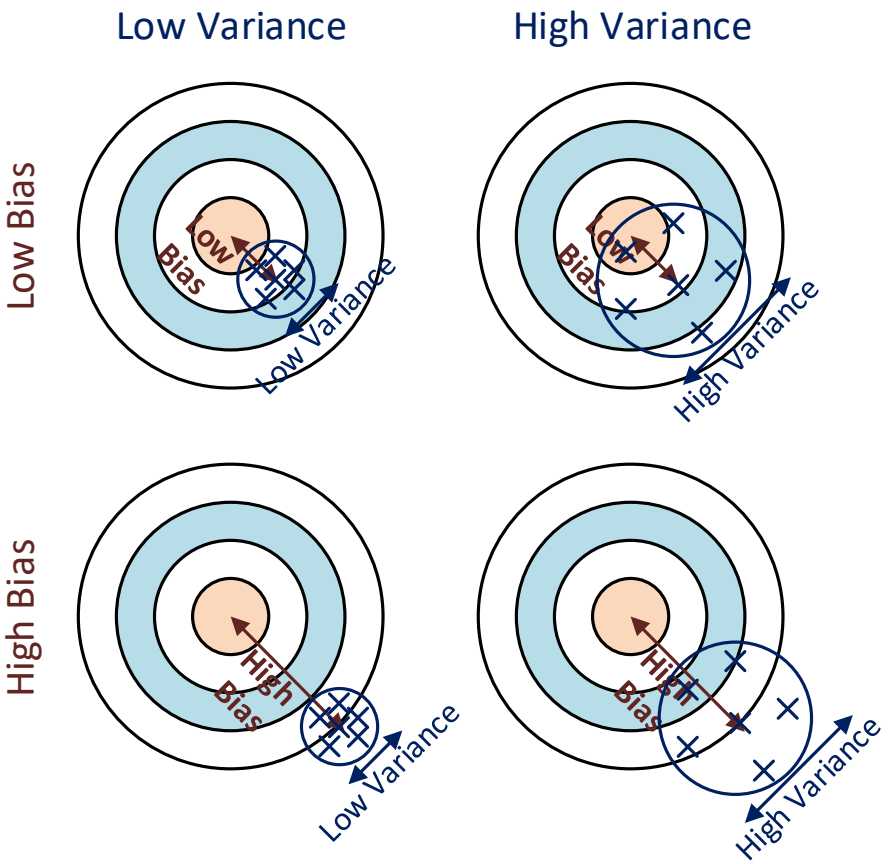
Codealong – Part G

Validation

❻ Build a Model

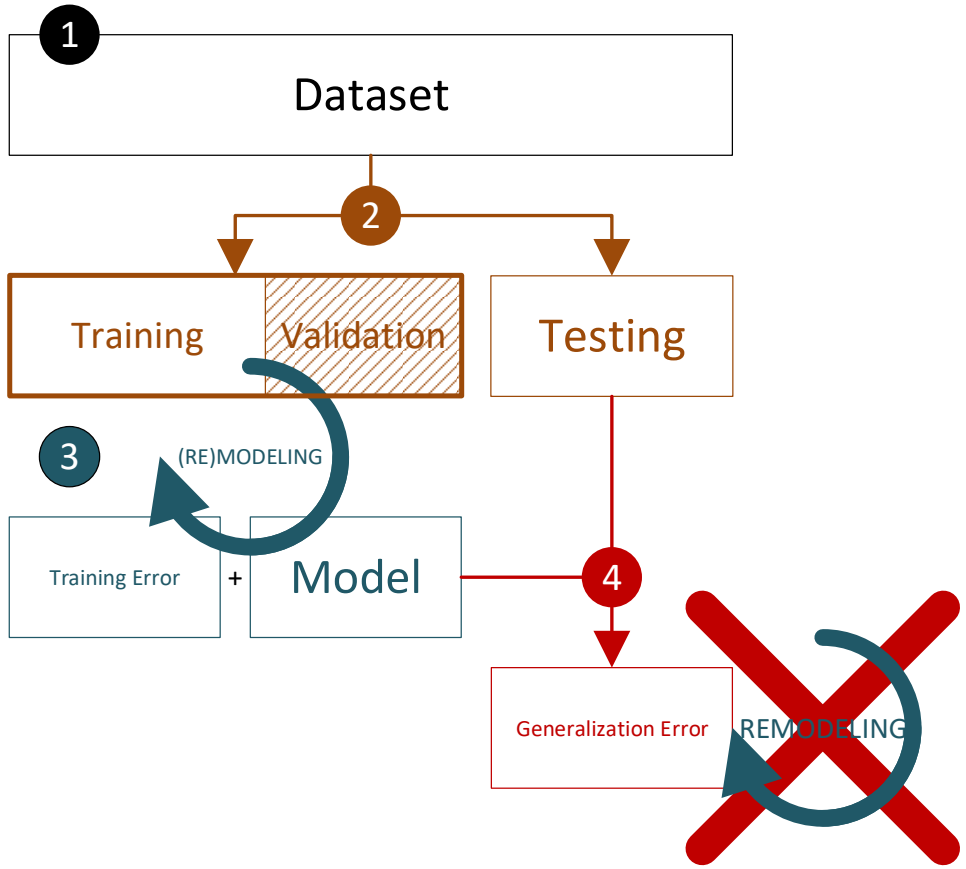
Cross-Validation

The generalization error has a bias component (systematic; non-random) and a variance component (idiosyncratic; random)



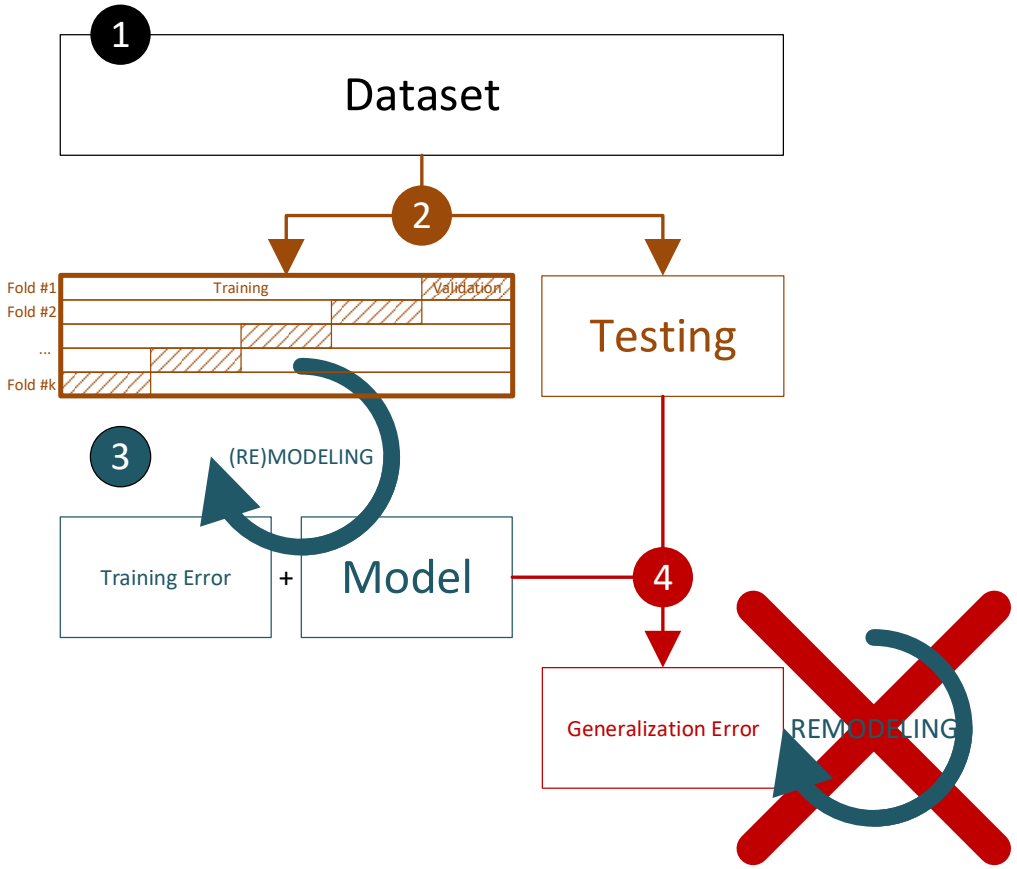
Cross-validation (CV) helps lowering the bias error

- Cross-validation
  - Another technique to validate models
  - Used to estimate how accurately the model generalize to unseen data
  - You can iterate as much as you want with the data
  - You then build a final model that uses all the data (cross-validation is used for model checking, not model building)
- [You still create an unseen testing set to estimate how well your model generalize to unseen data (and you stop there; no remodeling)]



k-fold cross-validation

- k-fold cross-validation
  - Popular
  - Typically, $k = 5$ or 10 with each sample being used both for training ($k - 1$ times) and validation (1 time)
  - The training error is the average training error of all folds
  - Again, after selecting the model that minimize the training error, you then build a final model that uses all the data
- You still create an unseen testing set to estimate how well your model generalize to unseen data (and you stop there; no remodeling)



Iris Dataset

Codealong – Part H

Cross-Validation

k-Nearest Neighbors

Pros and Cons

k-Nearest Neighbors | Pros and cons

- Pros
  - Intuitive and simple to explain
  - Training phase is fast
  - Non-parametric (does not presume a "form" of the "decision boundary")
  - Easily capture non-linearity

- Cons
  - Not interpretable
  - Prediction phase can be slow when n (number of observations) is large
  - Very sensitive to feature scaling; need to standardize the data
  - Sensitive to irrelevant features
  - Cannot be used if you have sparse data and feature space with dimension ≥ 4

Classification and k-Nearest Neighbors

Further Readings

Further Readings

- ISLR
  - An Overview of Classification (section 4.1, pp. 128 – 129)
- ESLII
  - k-Nearest-Neighbor Classifiers (section 13.3, pp. 463 – 475)
  - Cross-Validation (section 7.10, pp. 241 – 249)

Lab

k-Nearest Neighbors

Review

Review

- What are class labels? What does it mean to classify?
- How is a classification problem different from a regression problem? How are they similar?
- How does the k-Nearest Neighbors algorithm work?
- What primary parameters are available for tuning a k-Nearest Neighbors estimator?
- How do you define accuracy and misclassification?

Review (cont.)

You should now be able to:

- Define class label and classification
- Build a k-Nearest Neighbors model using *sklearn*
- Evaluate and tune model by using metrics such as classification accuracy/error

Before Next Class

Before Next Class

Before the next lesson, you should already be able to:

- Implement a linear model (`LinearRegression`) with *sklearn*
- Define the concept of coefficients
- Recall metrics for accuracy and misclassification

Next Class

Logistic Regression

Learning Objectives

After the next lesson, you should be able to:

- Build a logistic regression classification model using *sklearn*
- Describe the logit and sigmoid functions, odds and odds ratios, as well as how they relate to logistic regression
- Evaluate a model using metrics such as classification accuracy/error

Exit Ticket

Don’t forget to fill out your exit ticket here