Supervised learning

What is machine learning?

- The art *and* science of:
  - Giving computers the ability to learn to make decisions from data
  - ... without being explicitly programmed!
- Examples:
  - Learning to predict whether an email is spam or not
  - Clustering wikipedia entries into different categories
- Supervised learning: Uses labeled data
- Unsupervised learning: Uses unlabeled data

Unsupervised learning

- Uncovering hidden patterns from unlabeled data
- Example:
    - Grouping customers into distinct categories (Clustering)

Reinforcement learning

- Software agents interact with an environment
  - Learn how to optimize their behavior
  - Given a system of rewards and punishments
  - Draws inspiration from behavioral psychology
- Applications
  - Economics
  - Genetics
  - Game playing
- AlphaGo: First computer to defeat the world champion in Go

Supervised learning

- Predictor variables/features and a target variable
- Aim: Predict the target variable, given the predictor variables
    - Classification: Target variable consists of categories
    - Regression: Target variable is continuous

Predictor variables

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Target variable

species
setosa
setosa
setosa
setosa
setosa

Naming conventions

- Features = predictor variables = independent variables
- Target variable = dependent variable = response variable

Supervised learning

- Automate time-consuming or expensive manual tasks
    - Example: Doctor’s diagnosis
- Make predictions about the future
    - Example: Will a customer click on an ad or not?
- Need labeled data
    - Historical data with labels
    - Experiments to get labeled data
    - Crowd-sourcing labeled data

Supervised learning in Python

- We will use scikit-learn/sklearn
  - Integrates well with the SciPy stack
- Other libraries
  - TensorFlow
  - keras

Let's practice!

Exploratory data analysis

The Iris dataset

- Features:
  - Petal length
  - Petal width
  - Sepal length
  - Sepal width
- Target variable: Species
  - Versicolor
  - Virginica
  - Setosa

The Iris dataset in scikit-learn

```
In [1]: from sklearn import datasets

In [2]: import pandas as pd

In [3]: import numpy as np

In [4]: import matplotlib.pyplot as plt

In [5]: plt.style.use('ggplot')

In [6]: iris = datasets.load_iris()

In [7]: type(iris)
Out[7]: sklearn.datasets.base.Bunch

In [8]: print(iris.keys())
dict_keys(['data', 'target_names', 'DESCR', 'feature_names', 'target'])
```

The Iris dataset in scikit-learn

```
In [9]: type(iris.data), type(iris.target)
Out[9]: (numpy.ndarray, numpy.ndarray)

In [10]: iris.data.shape
Out[10]: (150, 4)

In [11]: iris.target_names
Out[11]: array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

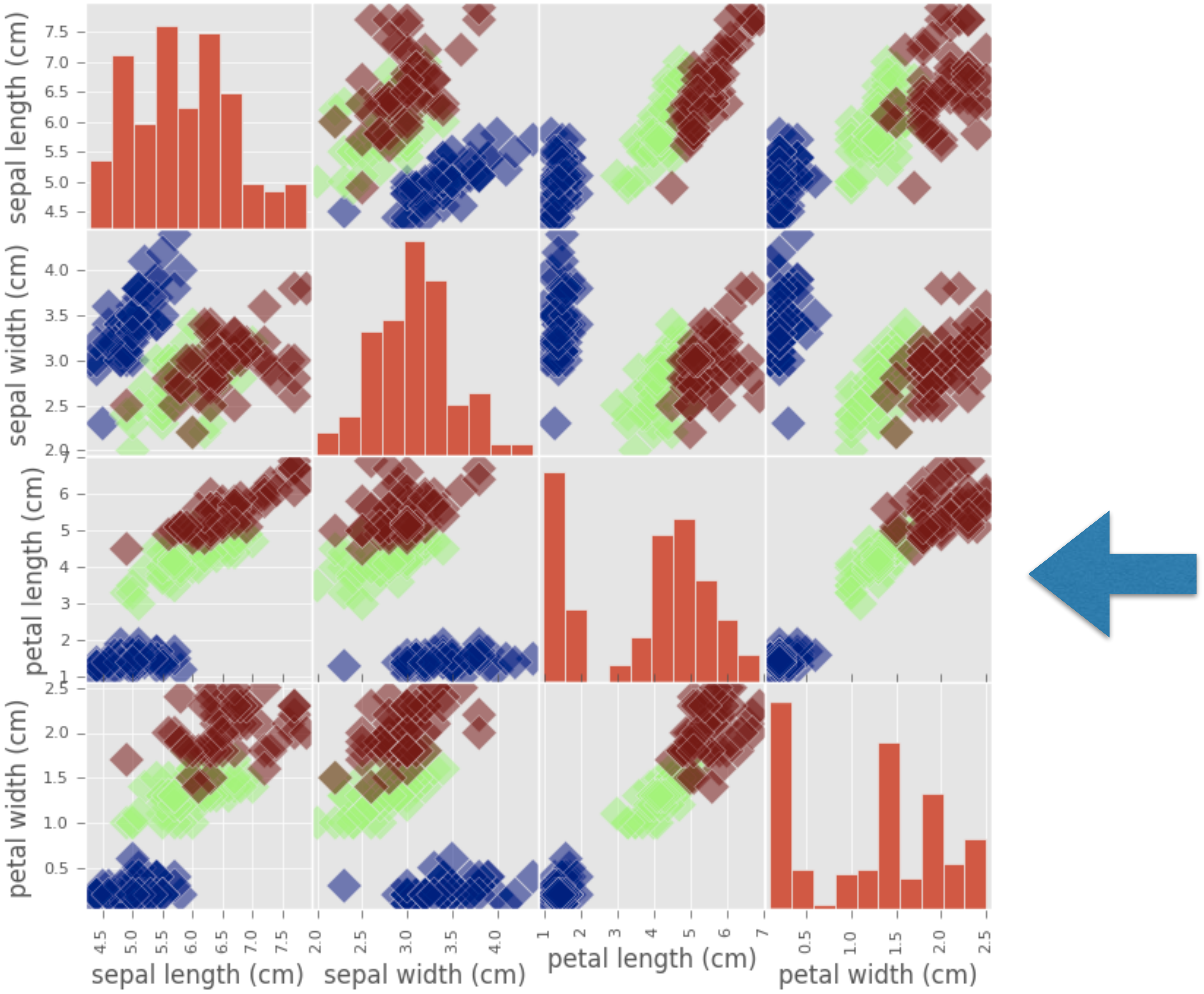
Exploratory data analysis (EDA)

```
In [12]: X = iris.data

In [13]: y = iris.target

In [14]: df = pd.DataFrame(X, columns=iris.feature_names)

In [15]: print(df.head())
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
```

Visual EDA

```
In [16]: _ = pd.scatter_matrix(df, c = y, figsize = [8, 8],
    ...:                      s=150, marker = 'D')
```

Visual EDA


sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
7
6
5
4
3
2
1
2.5
2.0
1.5
1.0
0.5
4.5 5.0 5.5 6.0 6.5 7.0 7.5
2.0 2.5 3.0 3.5 4.0
1 2 3 4 5 6 7
0.5 1.0 1.5 2.0 2.5

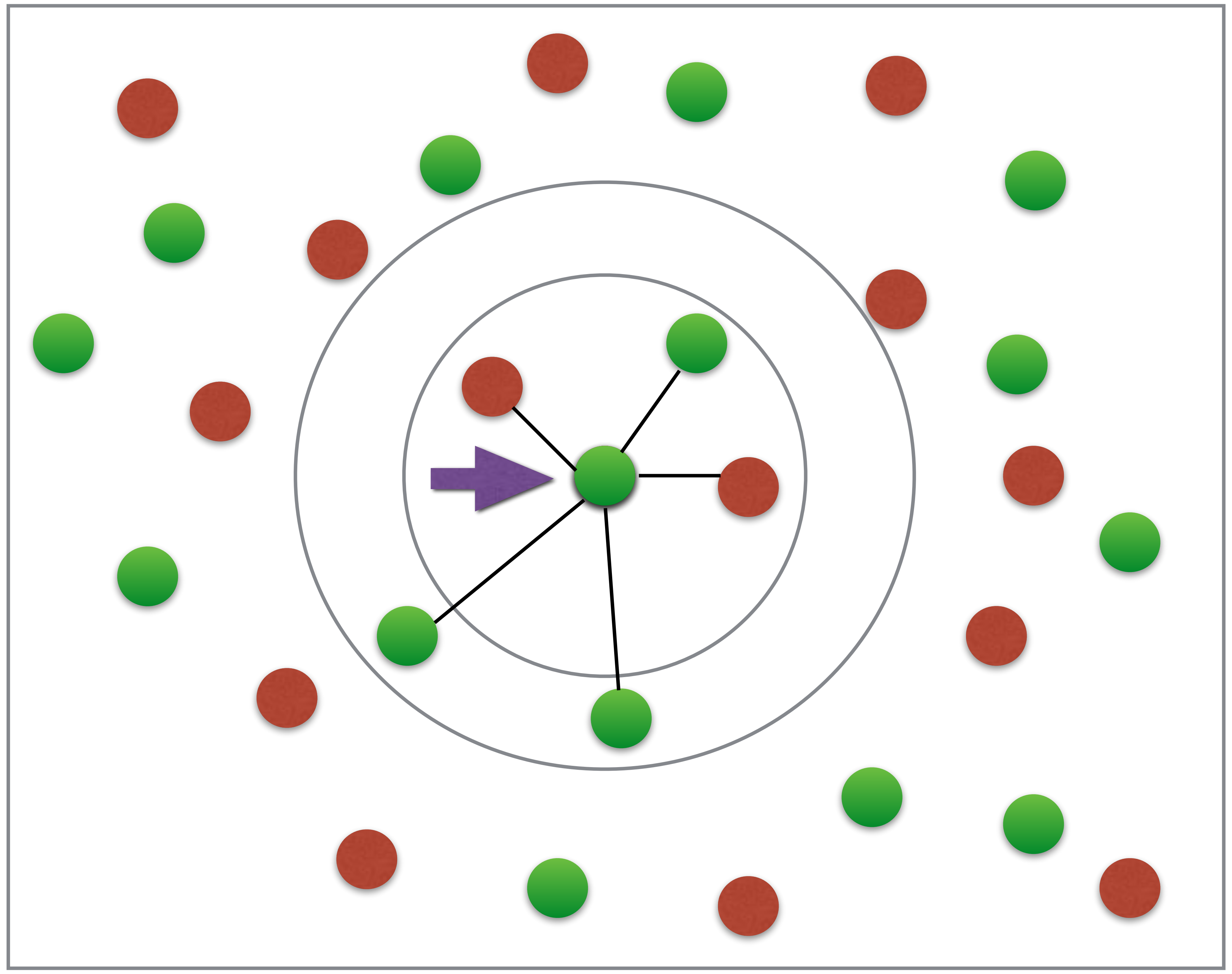
Let's practice!

The classification challenge

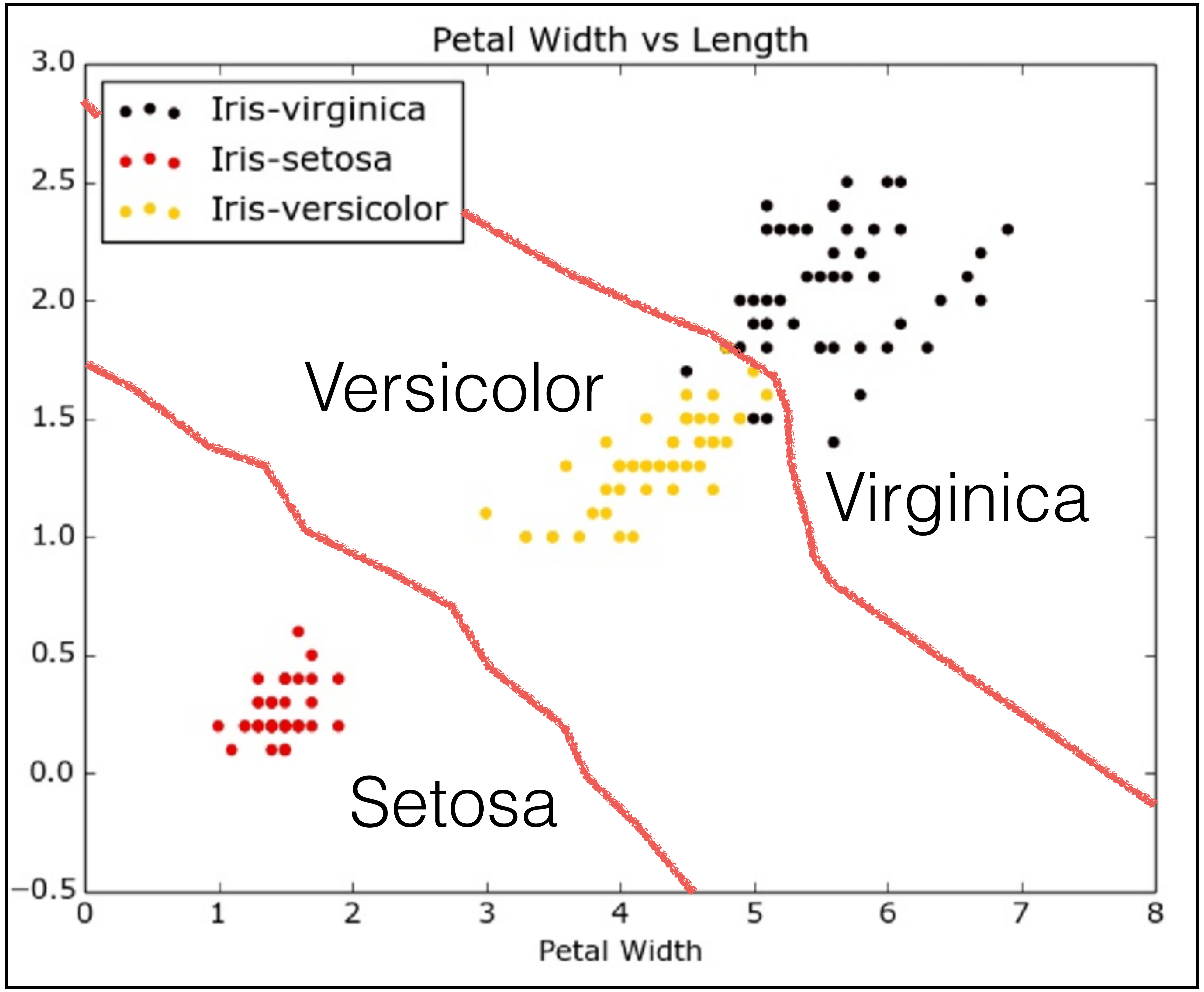
k-Nearest Neighbors

- Basic idea: Predict the label of a data point by
  - Looking at the ‘k’ closest labeled data points
  - Taking a majority vote

k-Nearest Neighbors



k-NN: Intuition



Scikit-learn fit and predict

- All machine learning models implemented as Python classes
  - They implement the algorithms for learning and predicting
  - Store the information learned from the data
- Training a model on the data = ‘fitting’ a model to the data
  - .fit() method
- To predict the labels of new data: .predict() method

Using scikit-learn to fit a classifier

```
In [1]: from sklearn.neighbors import KNeighborsClassifier

In [2]: knn = KNeighborsClassifier(n_neighbors=6)

In [3]: knn.fit(iris['data'], iris['target'])
Out[3]: KNeighborsClassifier(algorithm='auto', leaf_size=30,
   ...: metric='minkowski',metric_params=None, n_jobs=1,
   ...: n_neighbors=6, p=2,weights='uniform')

In [4]: iris['data'].shape
Out[4]: (150, 4)

In [5]: iris['target'].shape
Out[5]: (150,)
```

Predicting on unlabeled data

```
In [6]: prediction = knn.predict(X_new)

In [7]: X_new.shape
Out[7]: (3, 4)

In [8]: print('Prediction {}’.format(prediction))
Prediction: [1 1 0]
```

Let's practice!

Measuring model performance

Measuring model performance

- In classification, accuracy is a commonly used metric
- Accuracy = Fraction of correct predictions
- Which data should be used to compute accuracy?
- How well will the model perform on new data?

Measuring model performance

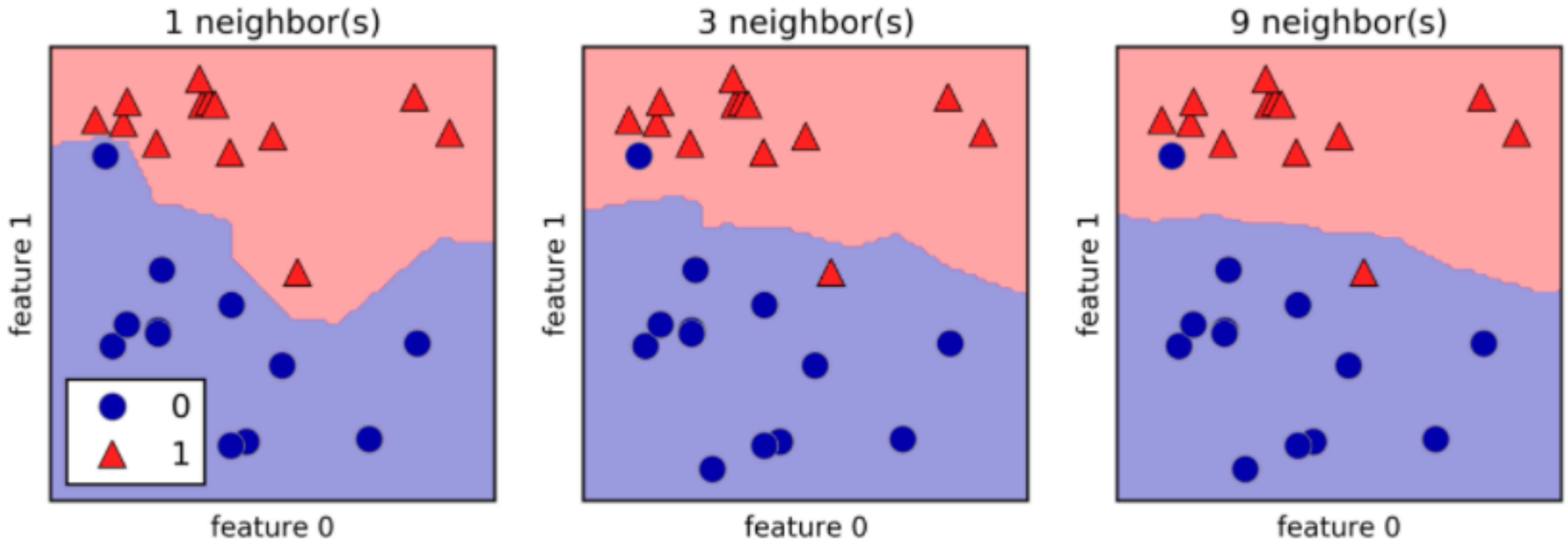
- Could compute accuracy on data used to fit classifier
  - NOT indicative of ability to generalize
- Split data into training and test set
  - Fit/train the classifier on the training set
  - Make predictions on test set
  - Compare predictions with the known labels

Train/test split

```
In [1]: from sklearn.model_selection import train_test_split

In [2]: X_train, X_test, y_train, y_test =
   ...: train_test_split(X, y, test_size=0.3,
   ...:                 random_state=21, stratify=y)

In [3]: knn = KNeighborsClassifier(n_neighbors=8)

In [4]: knn.fit(X_train, y_train)

In [5]: y_pred = knn.predict(X_test)

In [6]: print("Test set predictions:\n {}".format(y_pred))
Test set predictions:
 [2 1 2 2 1 0 1 0 0 1 0 2 0 2 2 0 0 0 1 0 2 2 2 0 1 1 1 0 0
 1 2 2 0 0 2 2 1 1 2 1 1 0 2 1]

In [7]: knn.score(X_test, y_test)
Out[7]: 0.9555555555555556
```

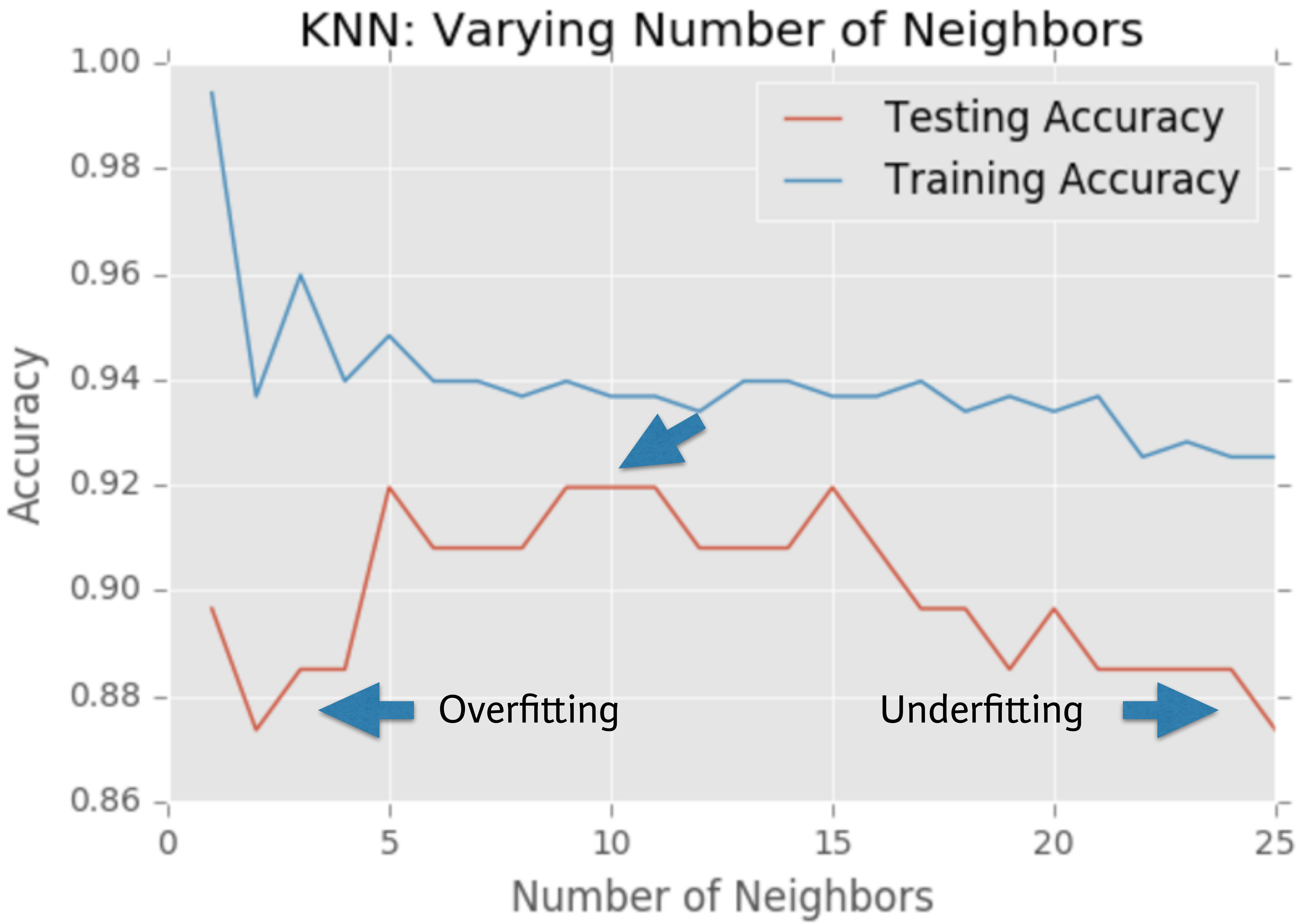
Model complexity

- Larger k = smoother decision boundary = less complex model
- Smaller k = more complex model = can lead to overfitting



Source: Andreas Müller & Sarah Guido, Introduction to Machine Learning with Python

Model complexity and over/underfitting



Let's practice!