

# Supervised learning

## SUPERVISED LEARNING WITH SCIKIT-LEARN



**Andreas Müller**

Core developer, scikit-learn

# 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

The diagram illustrates the components of supervised learning. On the left, a table titled "Predictor variables" shows five rows of data with columns for sepal length, sepal width, petal length, and petal width. A blue arrow points from the text "Predictor variables" to this table. On the right, a vertical list titled "Target variable" shows five entries all labeled "setosa", with a blue arrow pointing from the text "Target variable" to this list.

	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

Predictor variables

Target variable

species
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

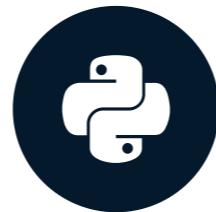
- We will use scikit-learn/sklearn
  - Integrates well with the SciPy stack
- Other libraries
  - TensorFlow
  - keras

# **Let's practice!**

**SUPERVISED LEARNING WITH SCIKIT-LEARN**

# Exploratory data analysis

SUPERVISED LEARNING WITH SCIKIT-LEARN



**Hugo Bowne-Anderson**  
Data Scientist, DataCamp

# The Iris dataset



Features:

- Petal length
- Petal width
- Sepal length
- Sepal width

Target variable: Species

- Versicolor
- Virginica
- Setosa

# The Iris dataset in scikit-learn

```
from sklearn import datasets  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
plt.style.use('ggplot')  
iris = datasets.load_iris()  
type(iris)
```

```
sklearn.datasets.base.Bunch
```

```
print(iris.keys())
```

```
dict_keys(['data', 'target_names', 'DESCR', 'feature_names', 'target'])
```

# The Iris dataset in scikit-learn

```
type(iris.data), type(iris.target)
```

```
(numpy.ndarray, numpy.ndarray)
```

```
iris.data.shape
```

```
(150, 4)
```

```
iris.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

# Exploratory data analysis (EDA)

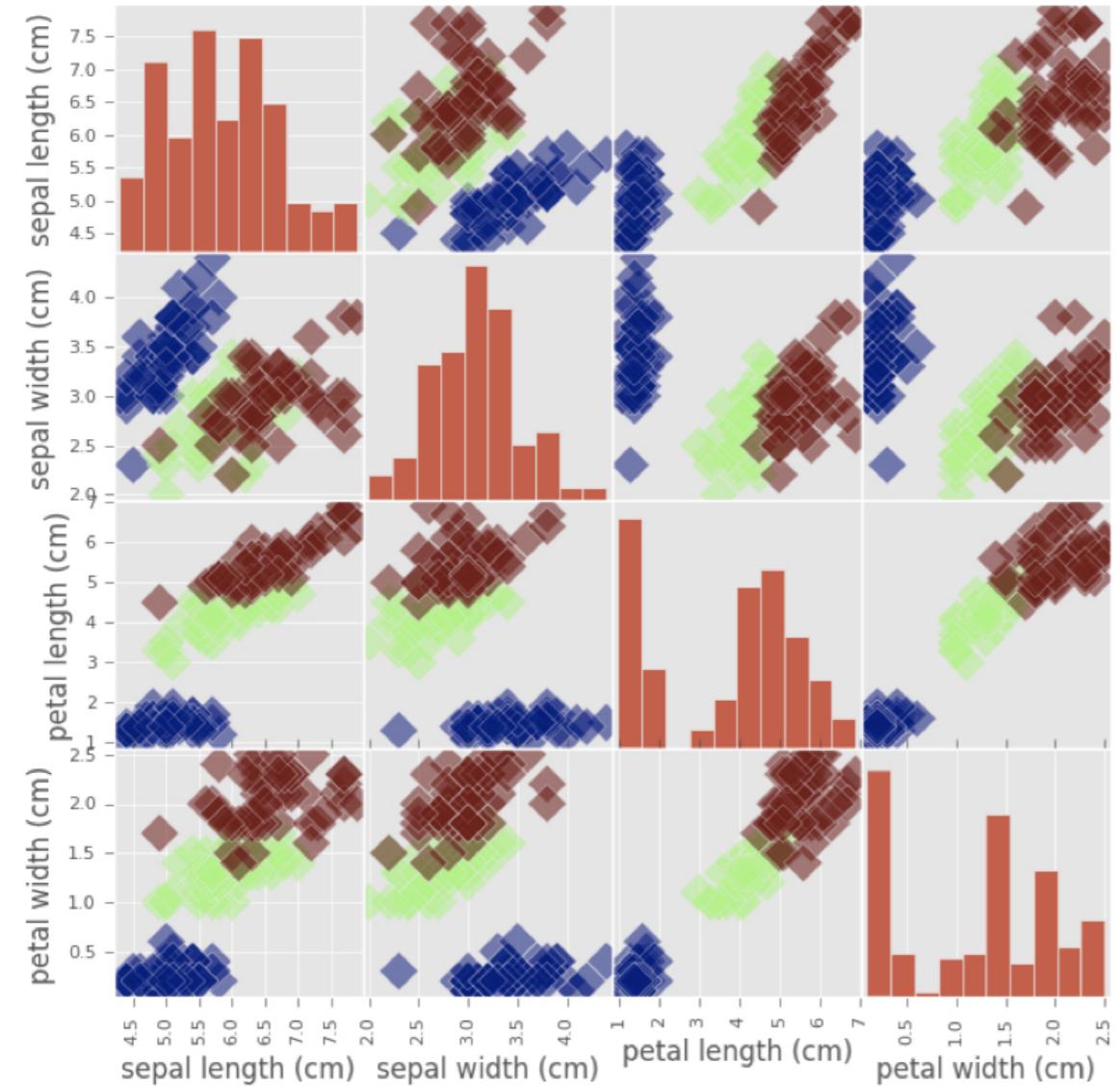
```
X = iris.data  
y = iris.target  
df = pd.DataFrame(X, columns=iris.feature_names)  
print(df.head())
```

	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

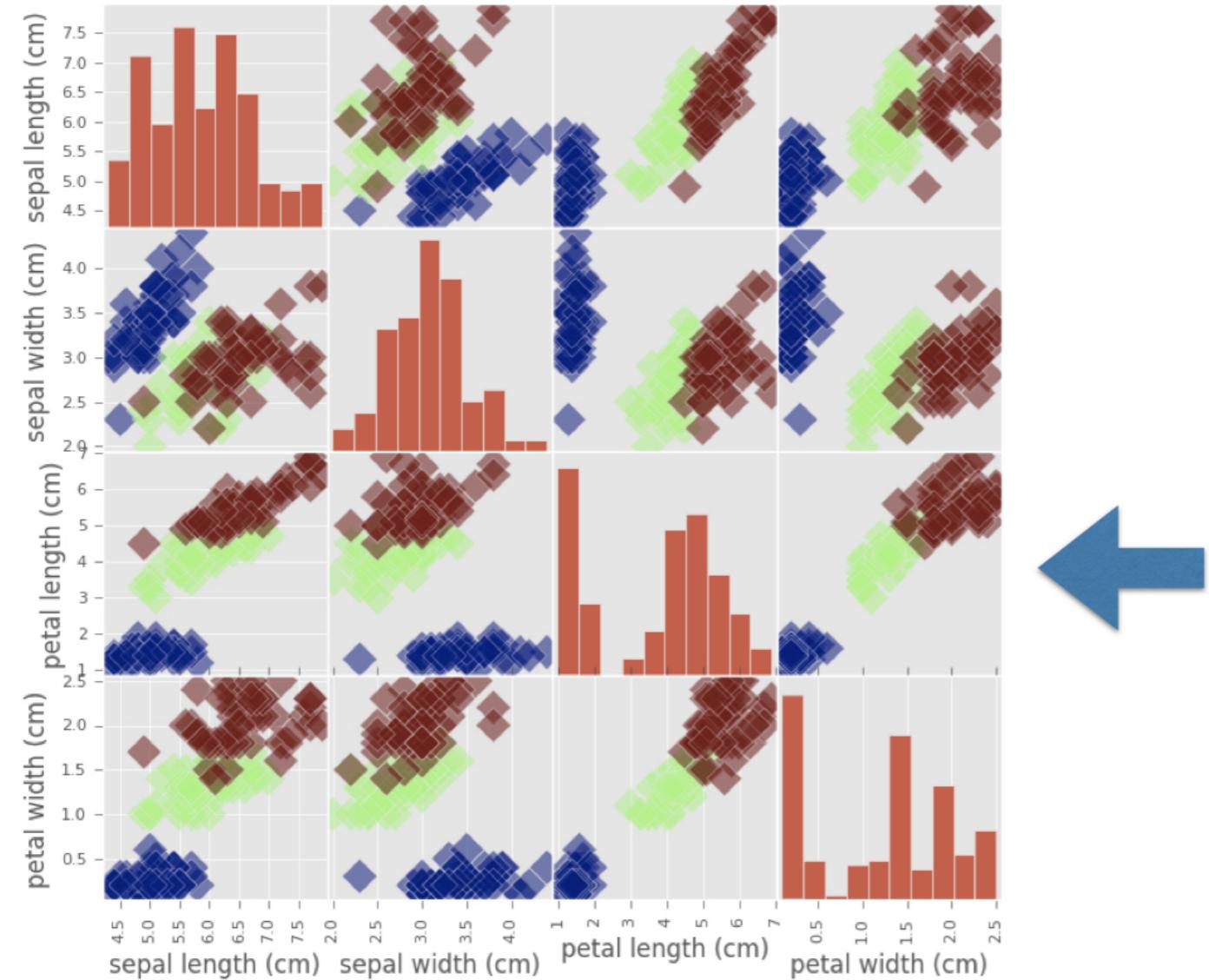
# Visual EDA

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

# Visual EDA



# Visual EDA

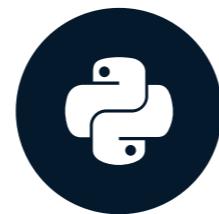


# **Let's practice!**

**SUPERVISED LEARNING WITH SCIKIT-LEARN**

# The classification challenge

SUPERVISED LEARNING WITH SCIKIT-LEARN

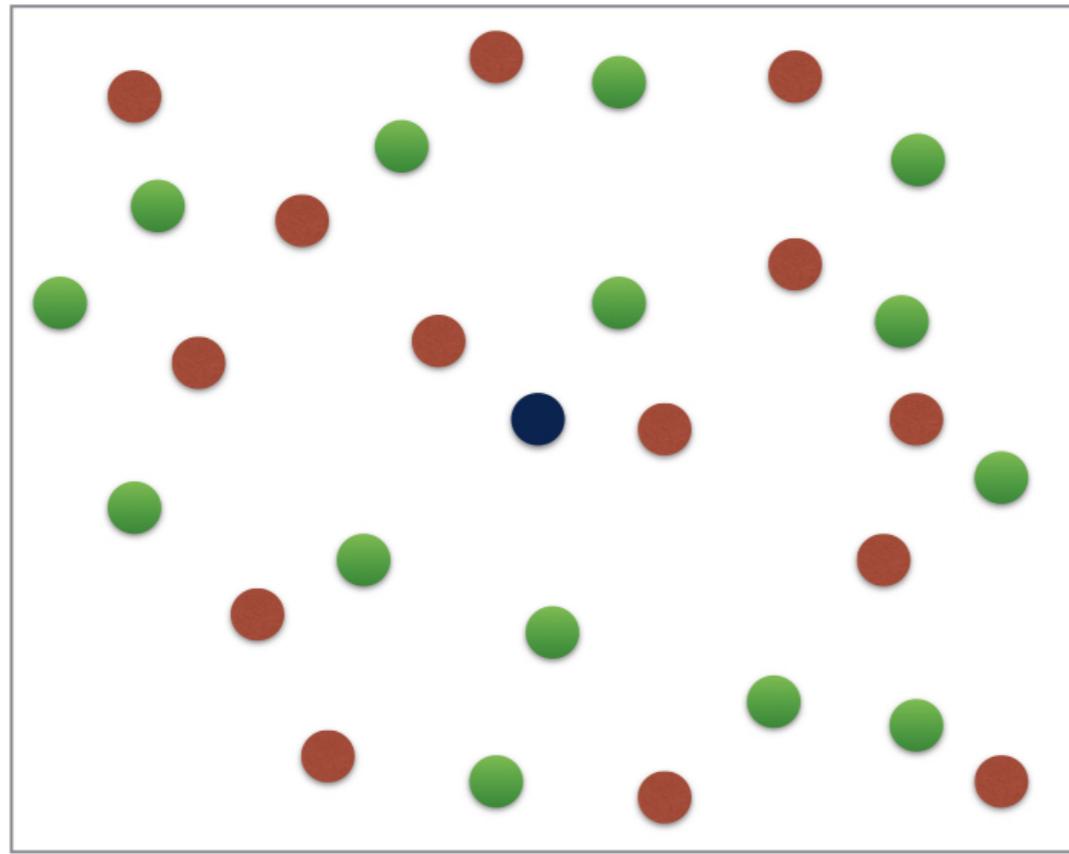


**Hugo Bowne-Anderson**  
Data Scientist, DataCamp

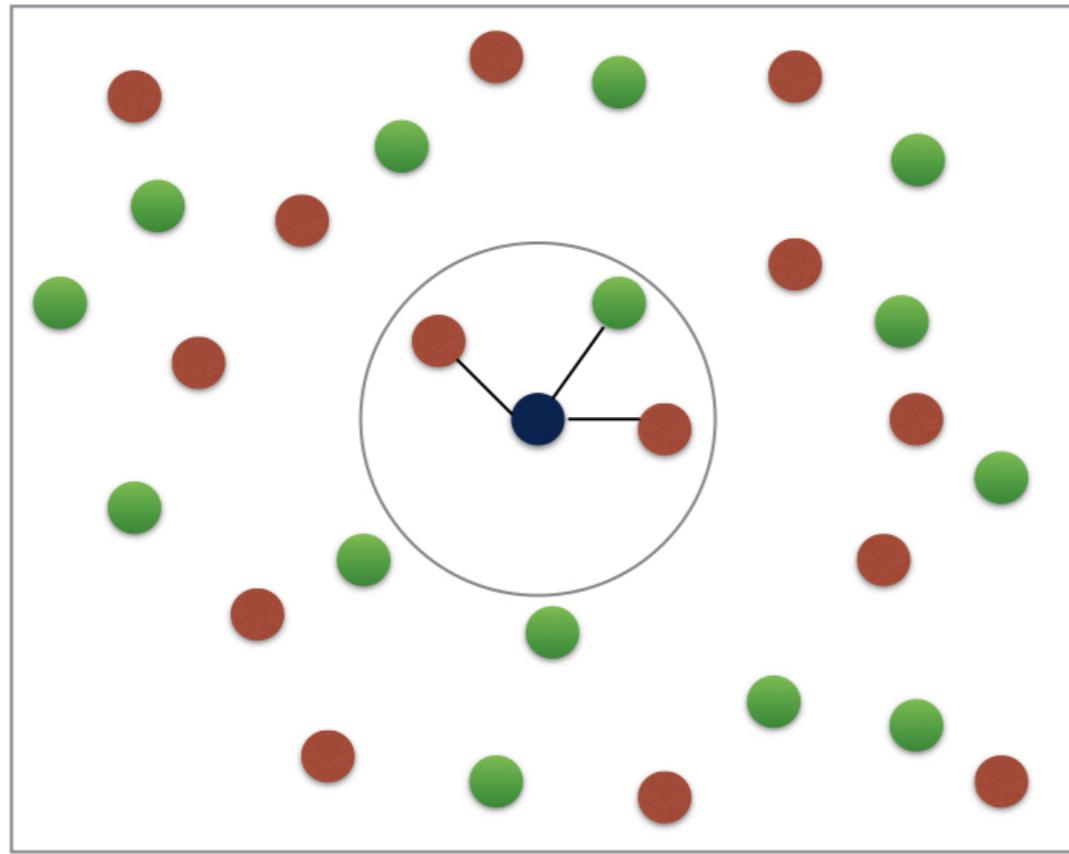
# k-Nearest Neighbors

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

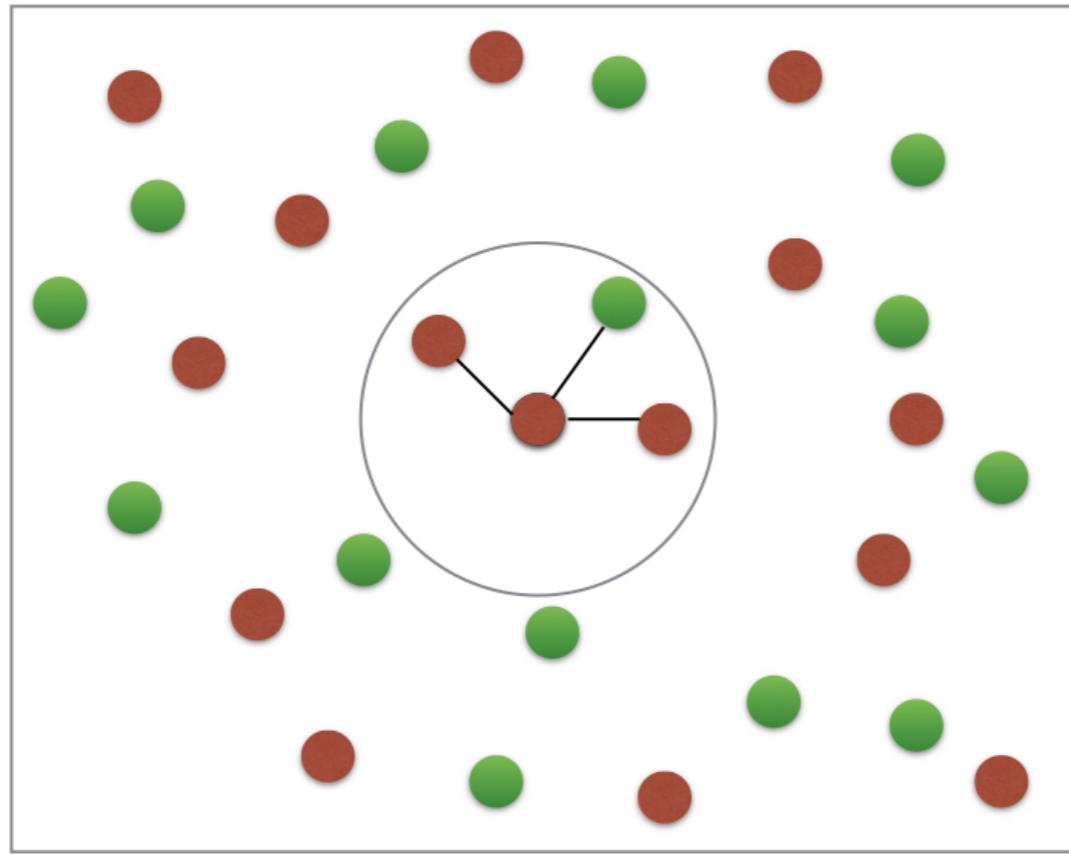
# k-Nearest Neighbors



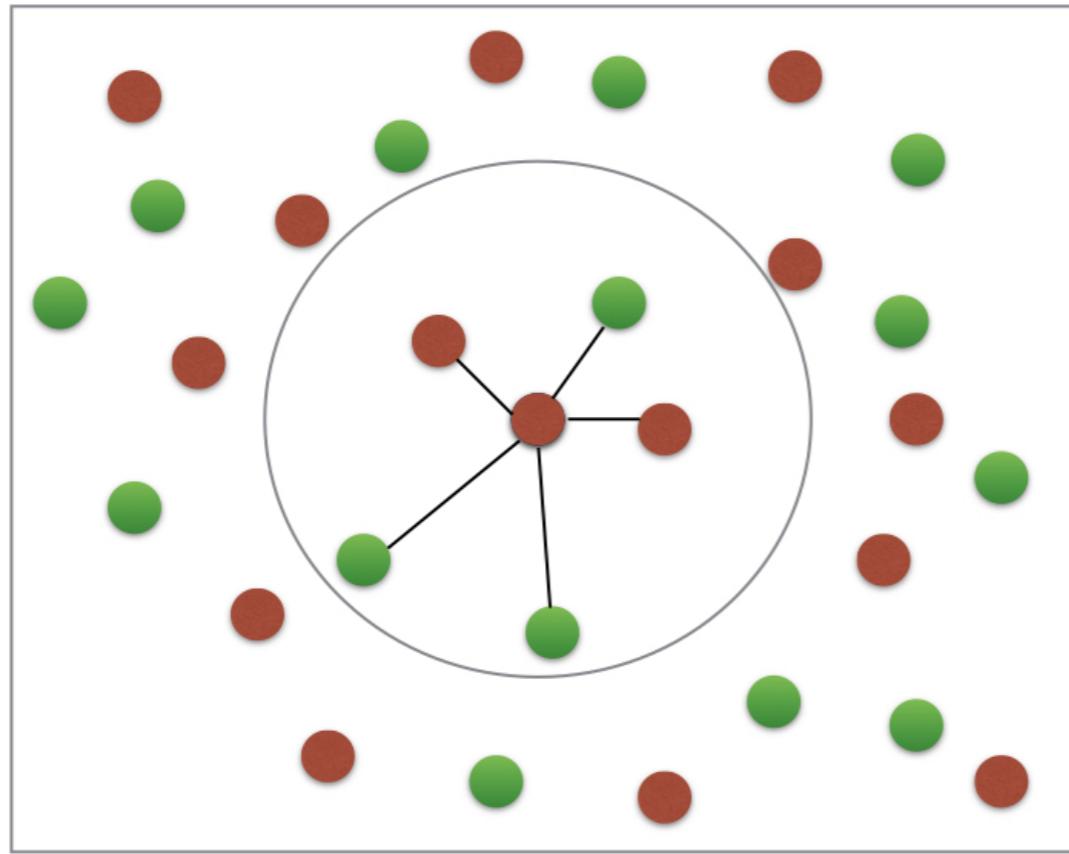
# k-Nearest Neighbors



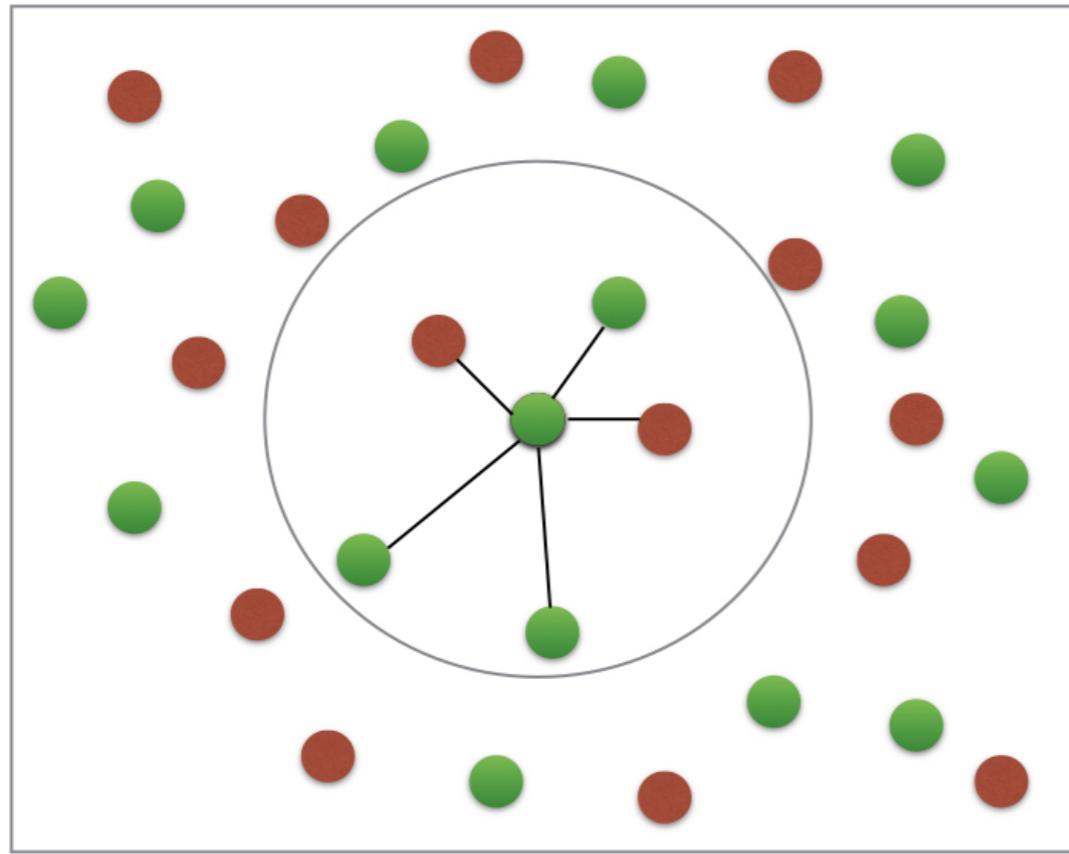
# k-Nearest Neighbors



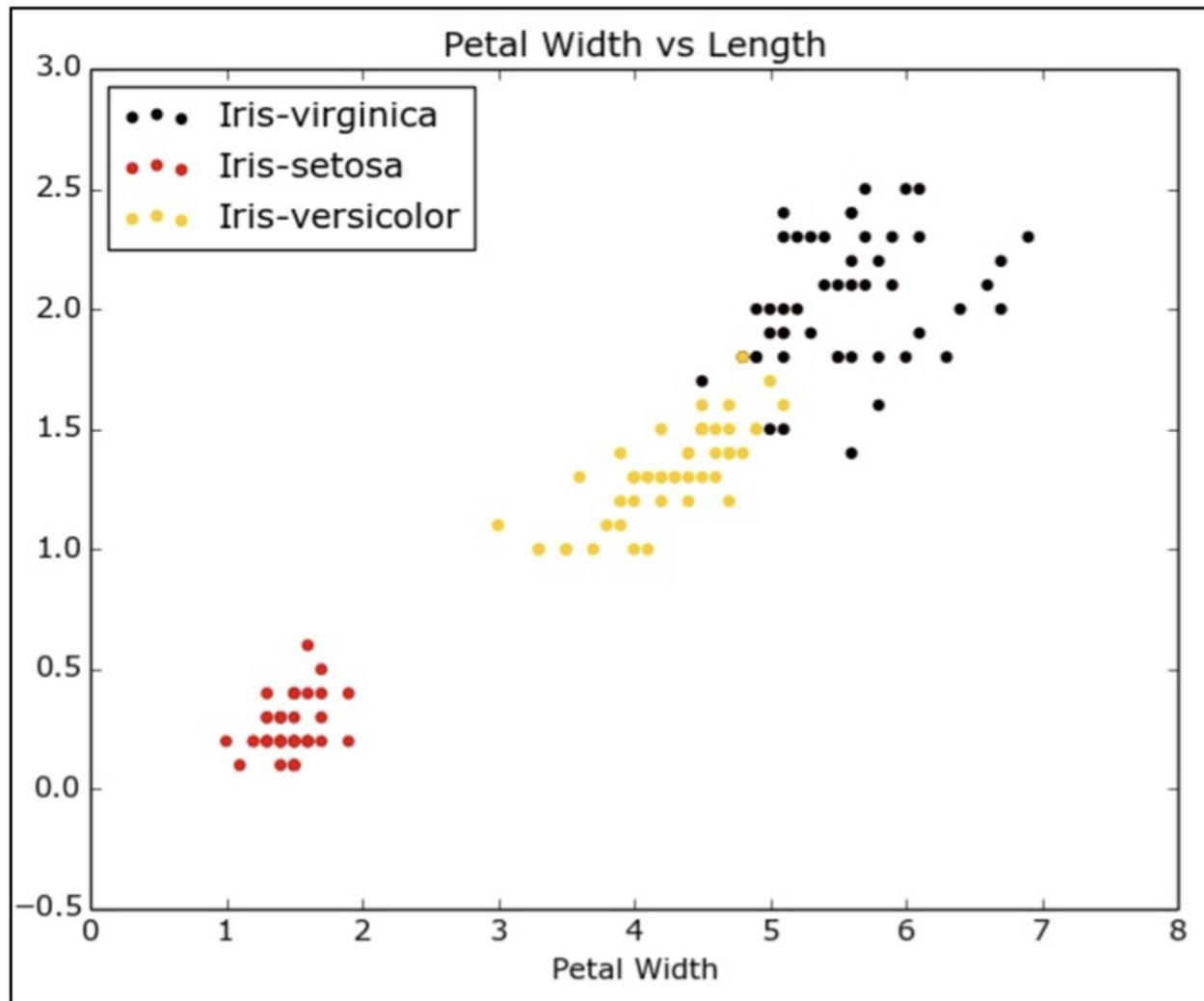
# k-Nearest Neighbors



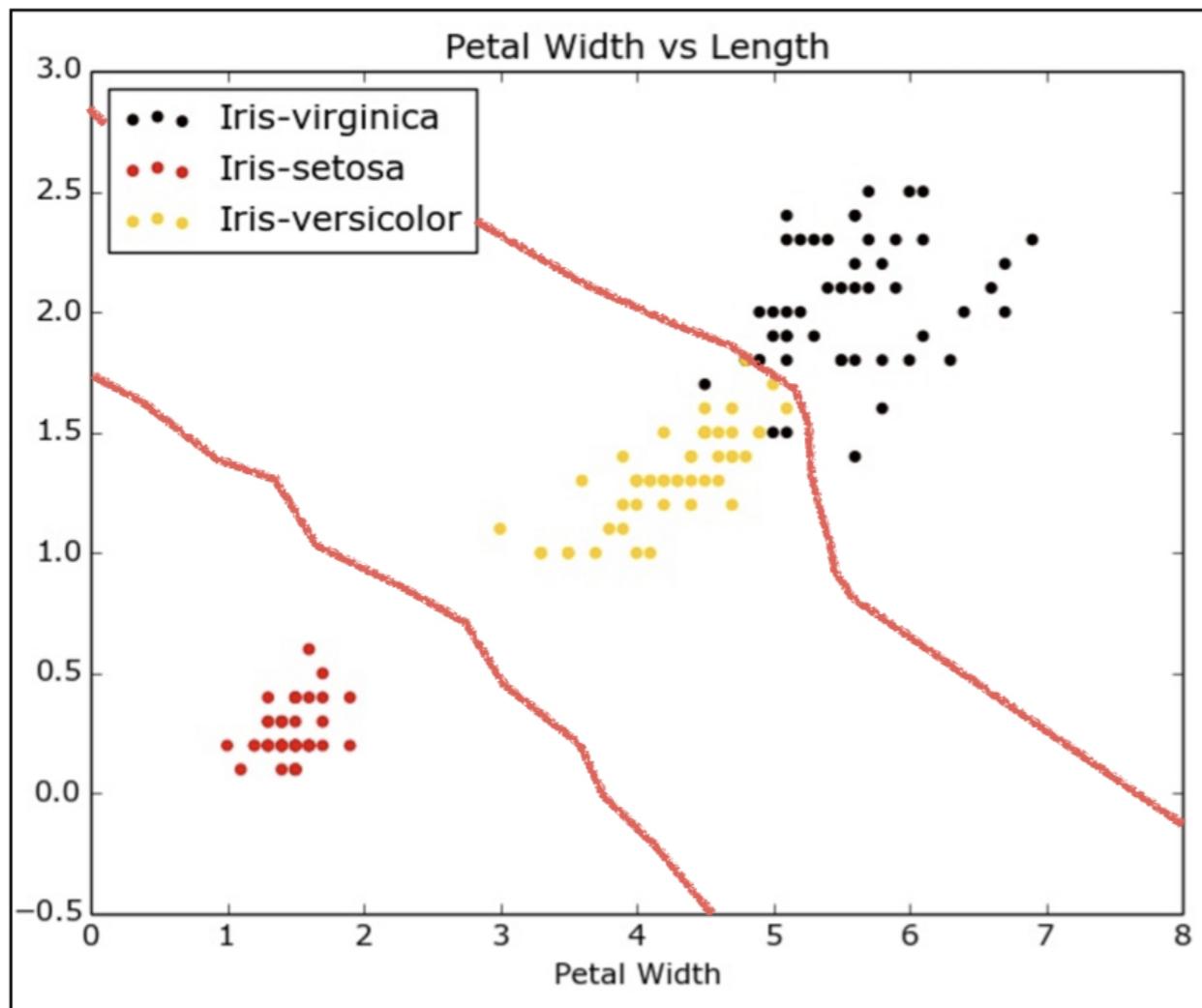
# k-Nearest Neighbors



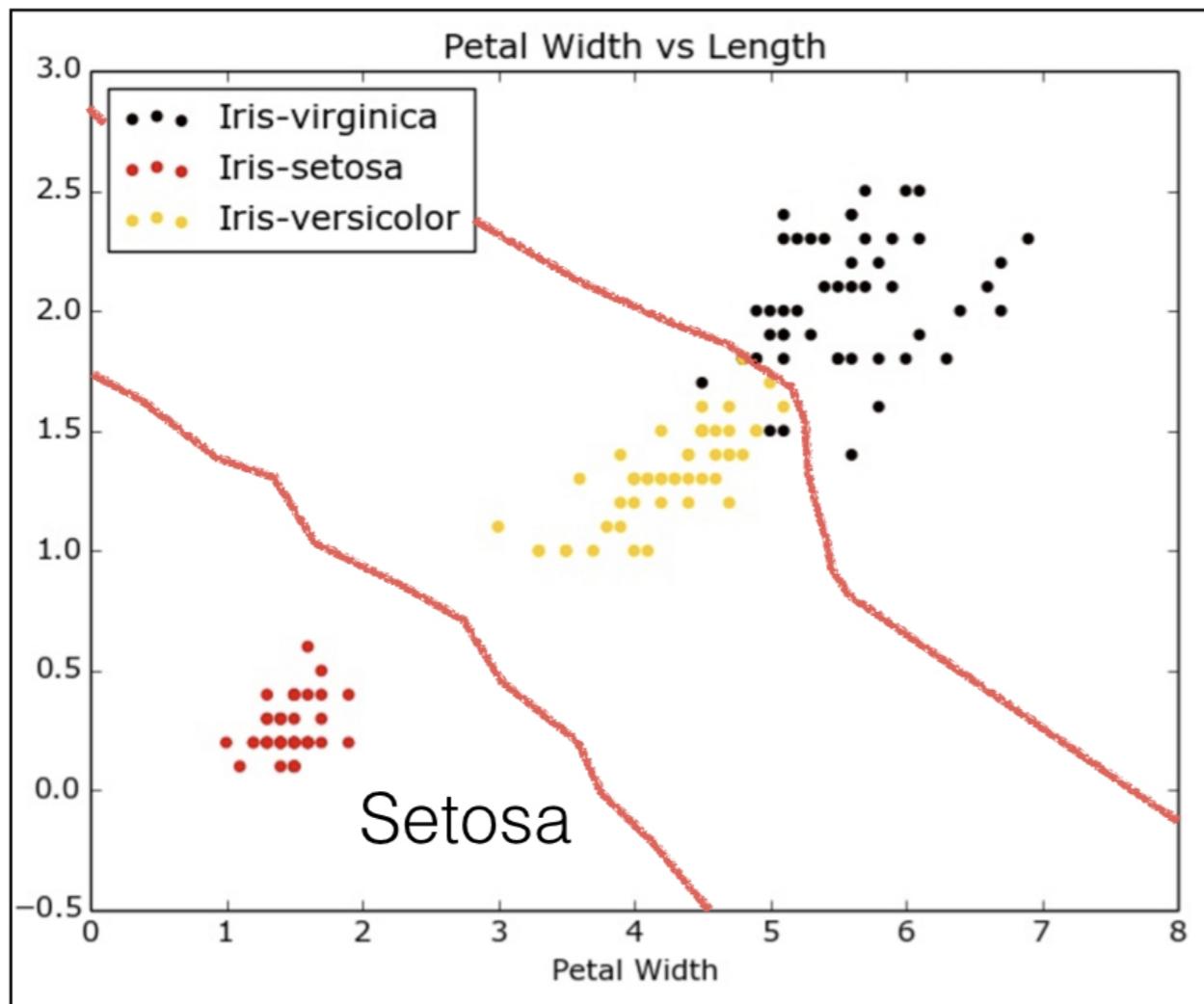
# k-NN: Intuition



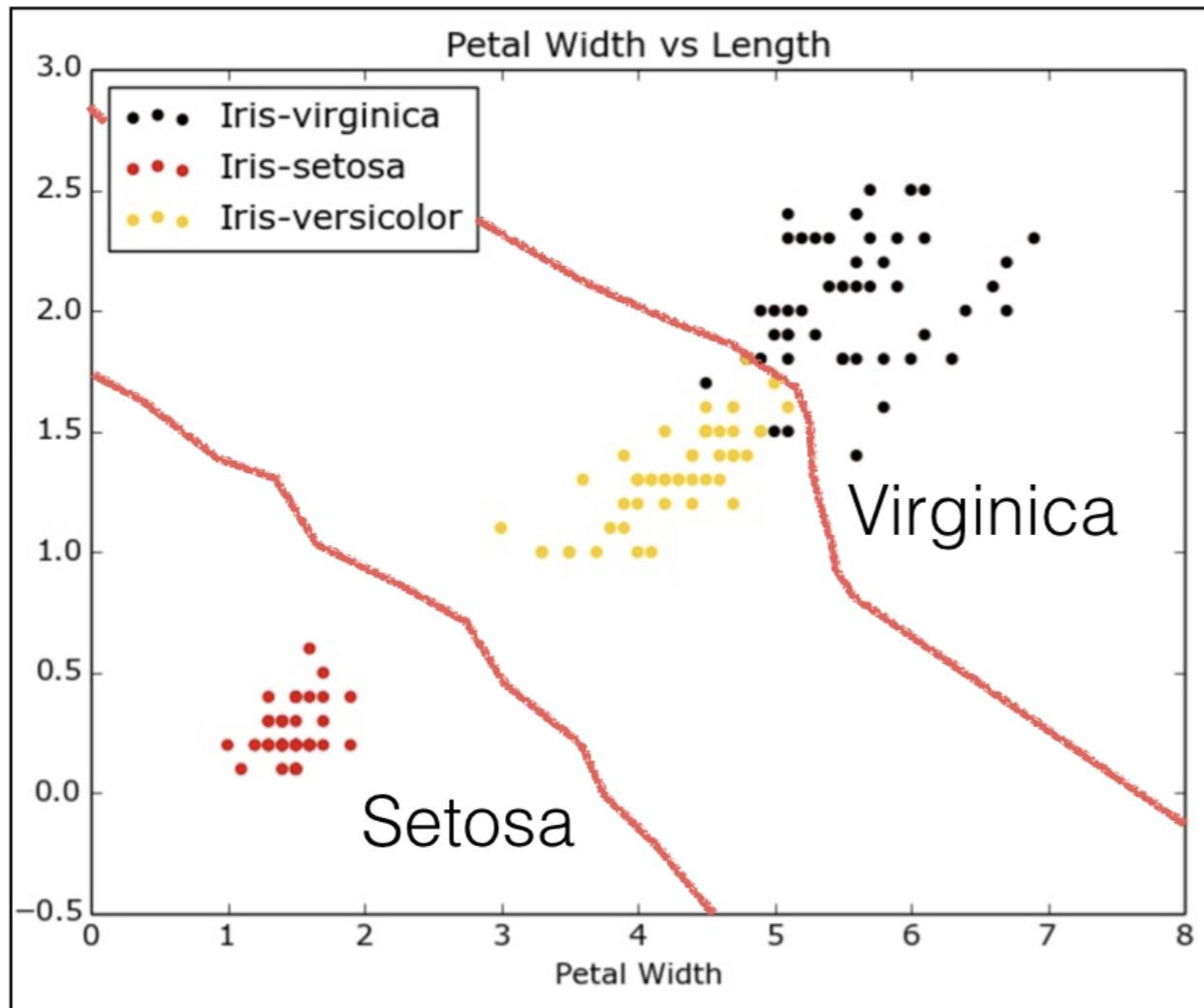
# k-NN: Intuition



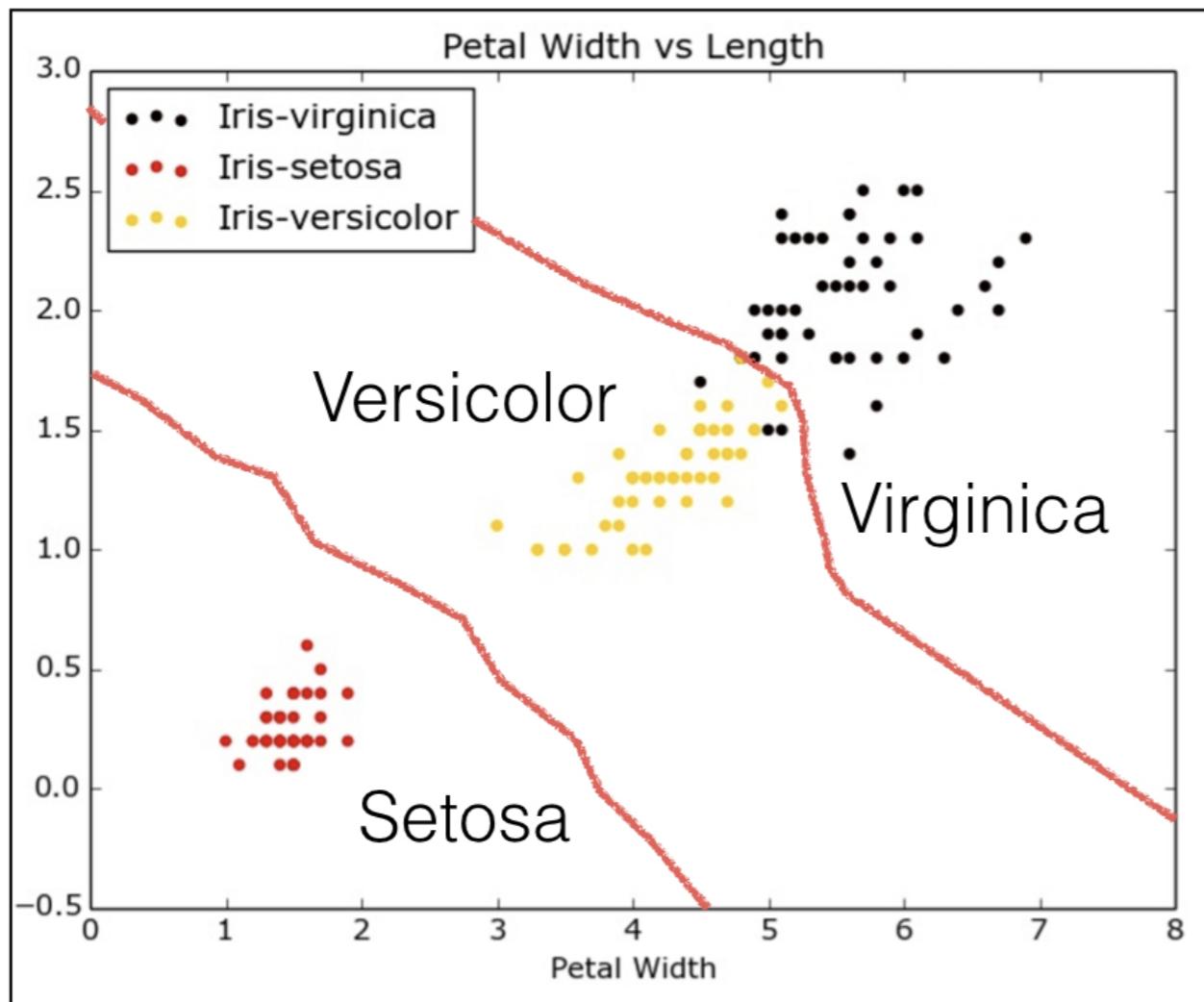
# k-NN: Intuition



# k-NN: Intuition



# 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

```
from sklearn.neighbors import KNeighborsClassifier  
knn = KNeighborsClassifier(n_neighbors=6)  
knn.fit(iris['data'], iris['target'])
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30,  
metric='minkowski', metric_params=None, n_jobs=1,  
n_neighbors=6, p=2, weights='uniform')
```

```
iris['data'].shape
```

```
(150, 4)
```

```
iris['target'].shape
```

```
(150,)
```

# Predicting on unlabeled data

```
X_new = np.array([[5.6, 2.8, 3.9, 1.1],  
                 [5.7, 2.6, 3.8, 1.3],  
                 [4.7, 3.2, 1.3, 0.2]])
```

```
prediction = knn.predict(X_new)  
X_new.shape
```

```
(3, 4)
```

```
print('Prediction: {}'.format(prediction))
```

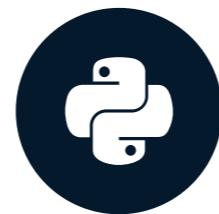
```
Prediction: [1 1 0]
```

# **Let's practice!**

**SUPERVISED LEARNING WITH SCIKIT-LEARN**

# Measuring model performance

SUPERVISED LEARNING WITH SCIKIT-LEARN



**Hugo Bowne-Anderson**  
Data Scientist, DataCamp

# 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

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.3,
                     random_state=21, stratify=y)
knn = KNeighborsClassifier(n_neighbors=8)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Test set predictions:\n {}".format(y_pred))
```

Test set predictions:

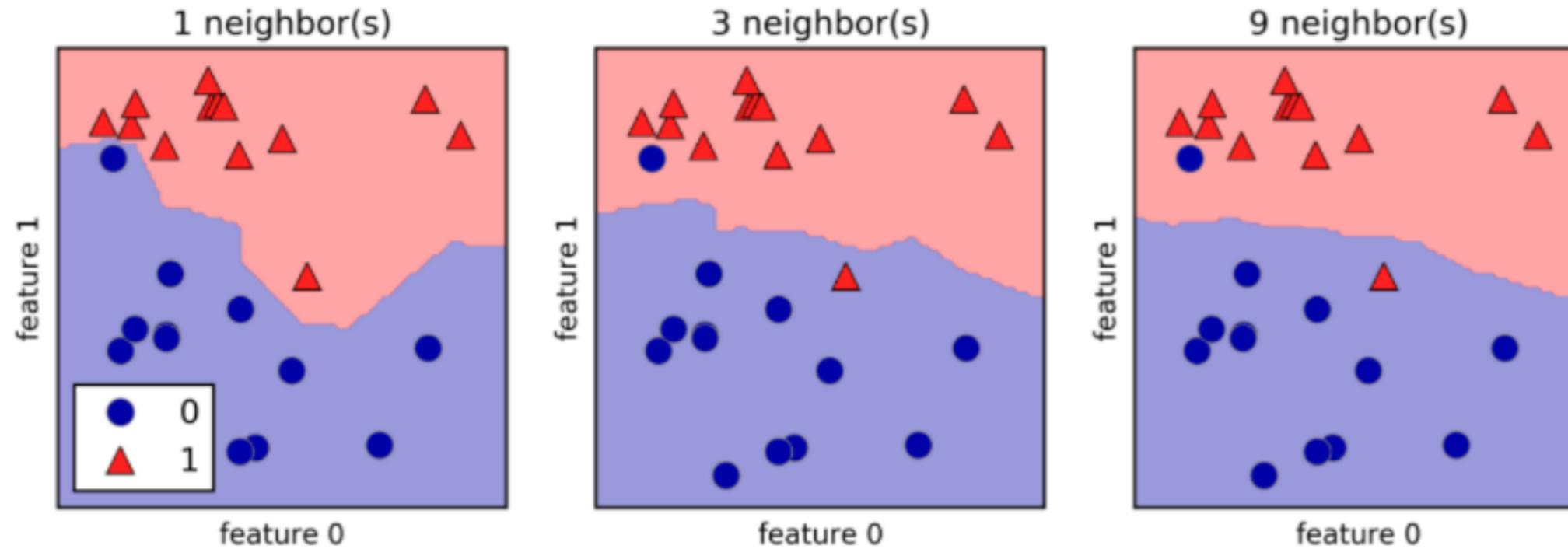
```
[2 1 2 2 1 0 1 0 0 1 0 2 0 2 2 0 0 0 1 0 2 2 2 0 1 1 1 0 0
 1 2 2 0 0 2 2 1 1 2 1 1 0 2 1]
```

```
knn.score(X_test, y_test)
```

```
0.9555555555555556
```

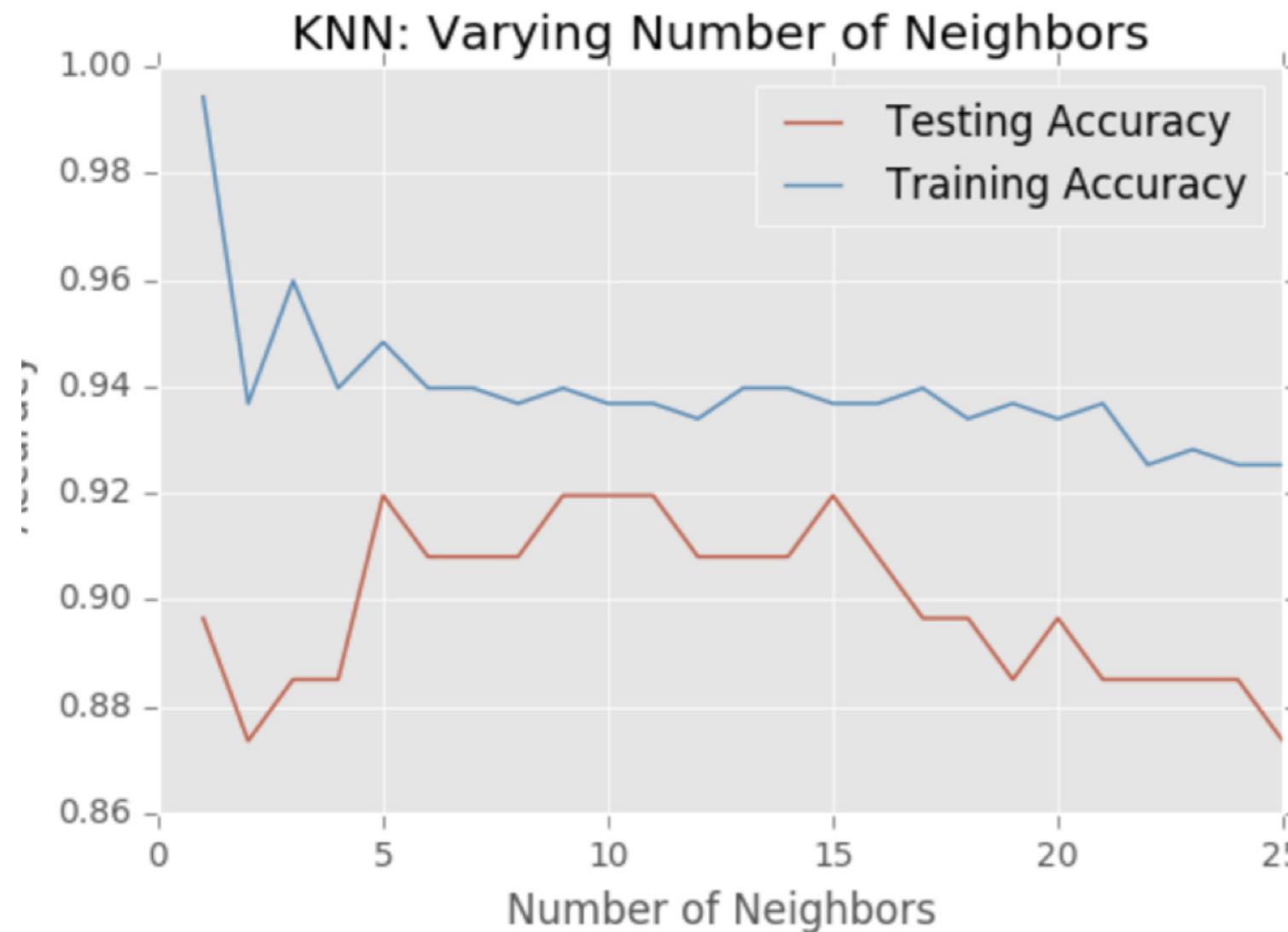
# Model complexity

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

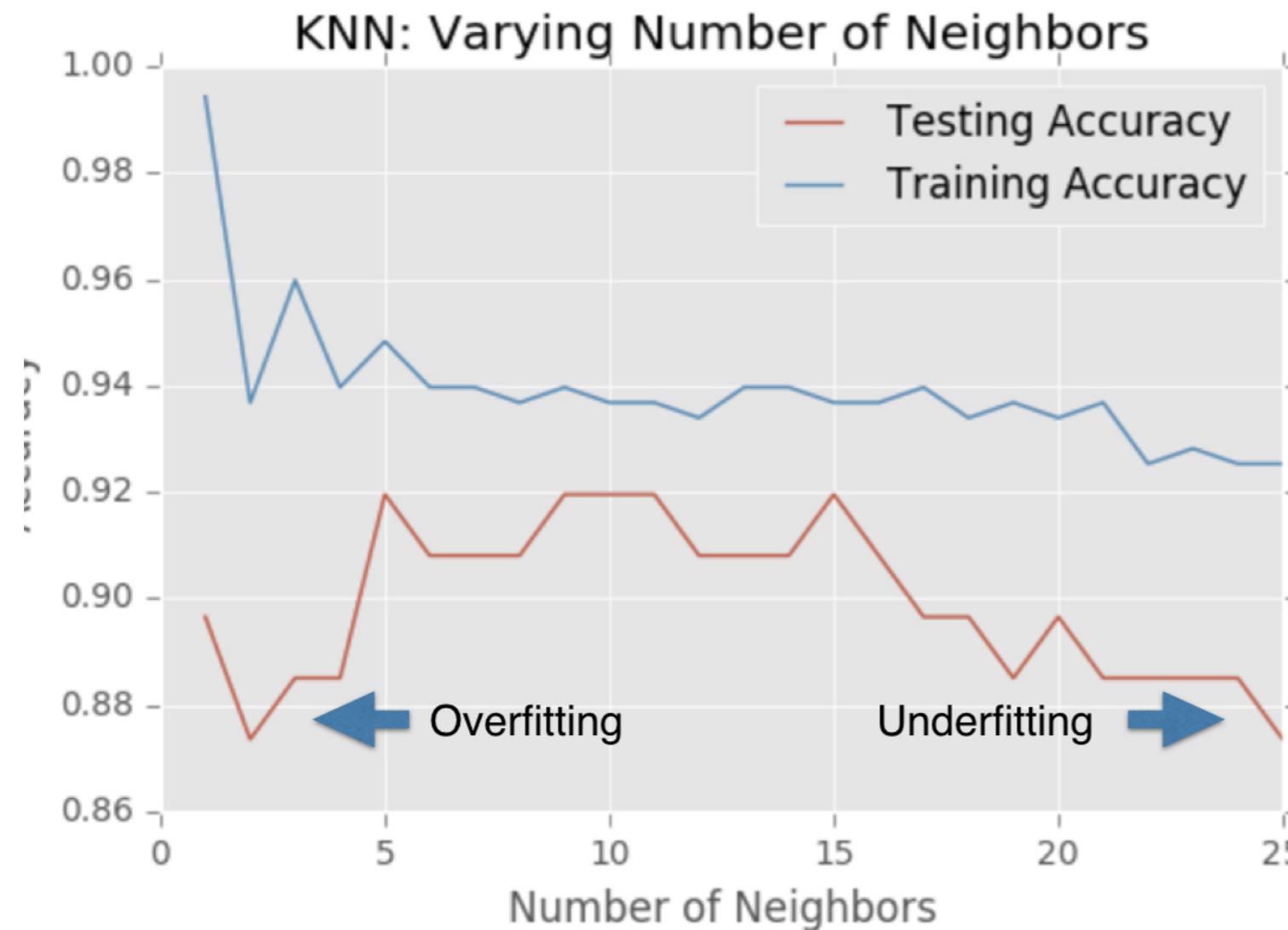


<sup>1</sup> Source: Andreas Müller & Sarah Guido, Introduction to Machine Learning with Python

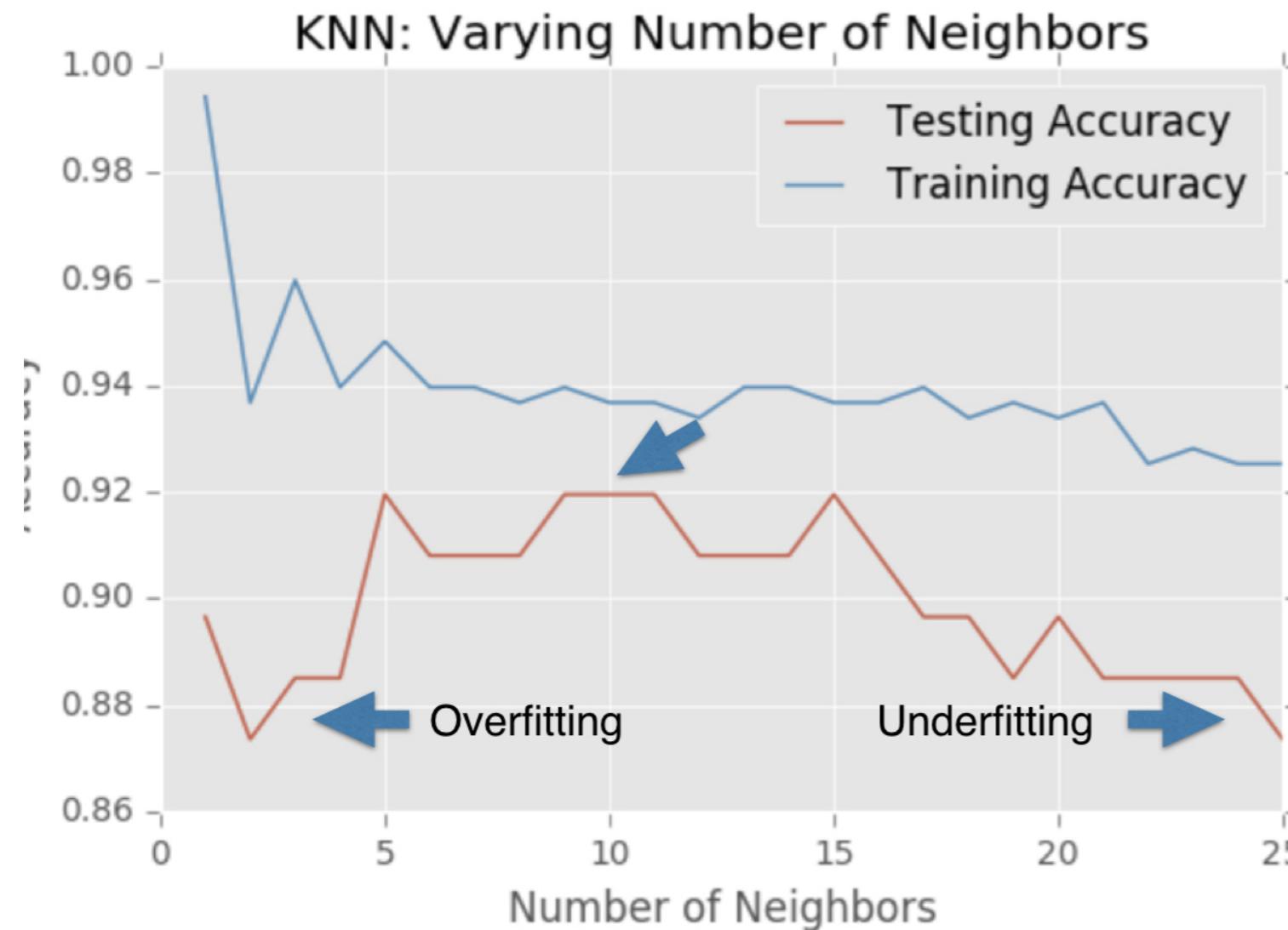
# Model complexity and over/underfitting



# Model complexity and over/underfitting



# Model complexity and over/underfitting



# **Let's practice!**

**SUPERVISED LEARNING WITH SCIKIT-LEARN**