



Introduction to Machine Learning in Python

Ankit Kariyaa

Tenure-track Assistant Professor
Department of Computer Science (DIKU) &
Geosciences and Natural Resource
Management (IGN)

January 12, 2026

UNIVERSITY OF COPENHAGEN



Concepts, Structure, and Practical Foundations

This lecture introduces the fundamental ideas of machine learning, explains how learning differs from traditional programming. The focus is on clarity and intuition, with practical examples implemented using scikit-learn.

First we introduce the unsupervised clustering method KMeans, followed by the supervised methods k-Nearest Neighbors, Logistic Regression and Decision Trees.

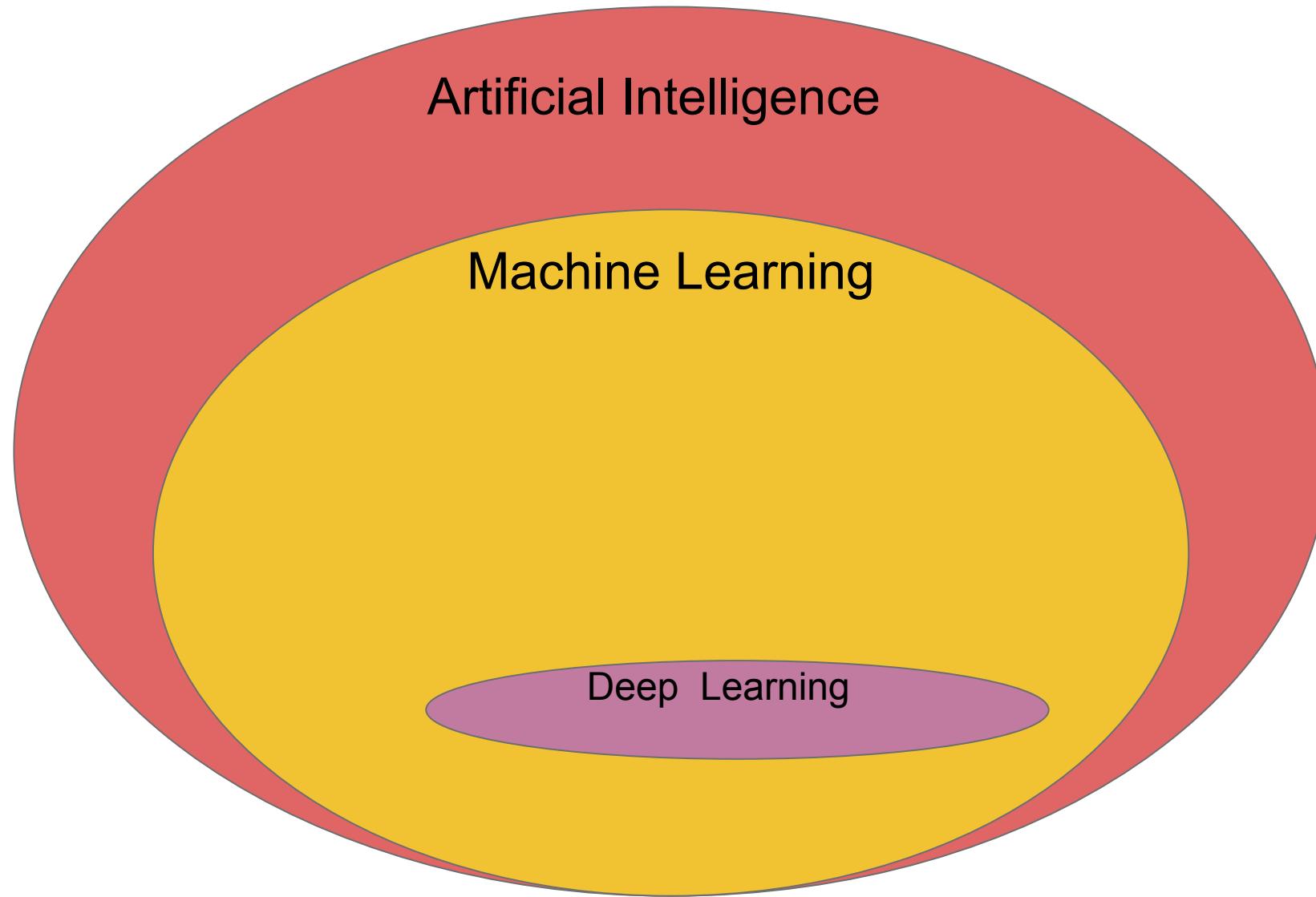
Further resources:

The Microsoft training module: [Introduction to machine learning concepts](#) offers a video-based learning module on these topic.

Further readings:

- Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4). Chapter 9.1. K-means clustering. 7 pages.
- Géron, A. (2022). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc. Chapter 4. Logistic Regression. 9 pages.
- Lindholm, A., Wahlström, N., Lindsten, F., & Schön, T. B. (2022). Machine learning: a first course for engineers and scientists. Cambridge University Press. Chapter 1: Introduction. 13 pages.
- Lindholm, A., Wahlström, N., Lindsten, F., & Schön, T. B. (2022). Machine learning: a first course for engineers and scientists. Cambridge University Press. Chapter 2. Supervised Learning: A First Approach. 23 pages.

Artificial Intelligence, machine learning, and deep learning



What Is Machine Learning?

Machine learning is a branch of artificial intelligence concerned with building systems that **learn patterns from data** rather than relying on explicitly programmed rules.

A machine learning system improves its performance on a task as it gains experience, where experience is represented by data. The goal is not to memorize examples, but to **generalize** from observed data to unseen cases.

Machine Learning vs Traditional Programming

Traditional software follows a rule-based paradigm:

Rules + Input → Output

Machine learning reverses this logic:

Data + Desired Output → Learned Model

New Input + Learned Model → Predicted Output

Instead of writing rules manually, we let algorithms infer rules implicitly by optimizing a mathematical objective.

Important: The new input must be independent and identically distributed (i.i.d) to the original data.

“

Why Machine Learning Is Needed?

Many real-world problems are too complex, uncertain, or high-dimensional to solve with manually designed rules, making data-driven learning essential

Why Machine Learning Is Needed - How many trees are there in this image?



ChatGPT: Approx 230

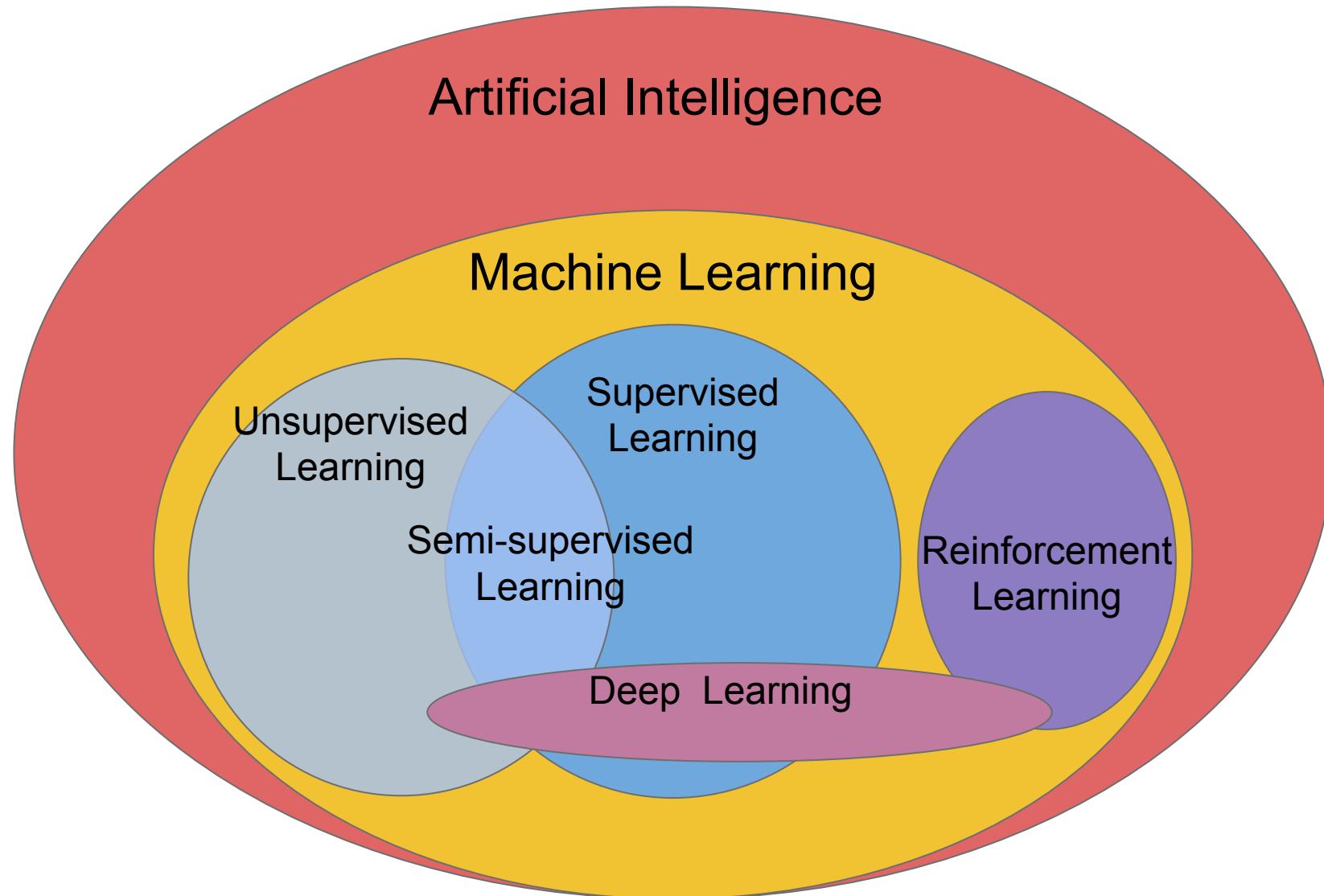
The Machine Learning Workflow

A typical machine learning workflow consists of:

1. Collecting and cleaning data
2. Selecting and transforming features
3. Choosing a learning algorithm
4. Training the model on data
5. Evaluating generalization performance
6. Deploying and monitoring the model

Each step influences the final outcome and must be handled systematically.

Types of Machine Learning



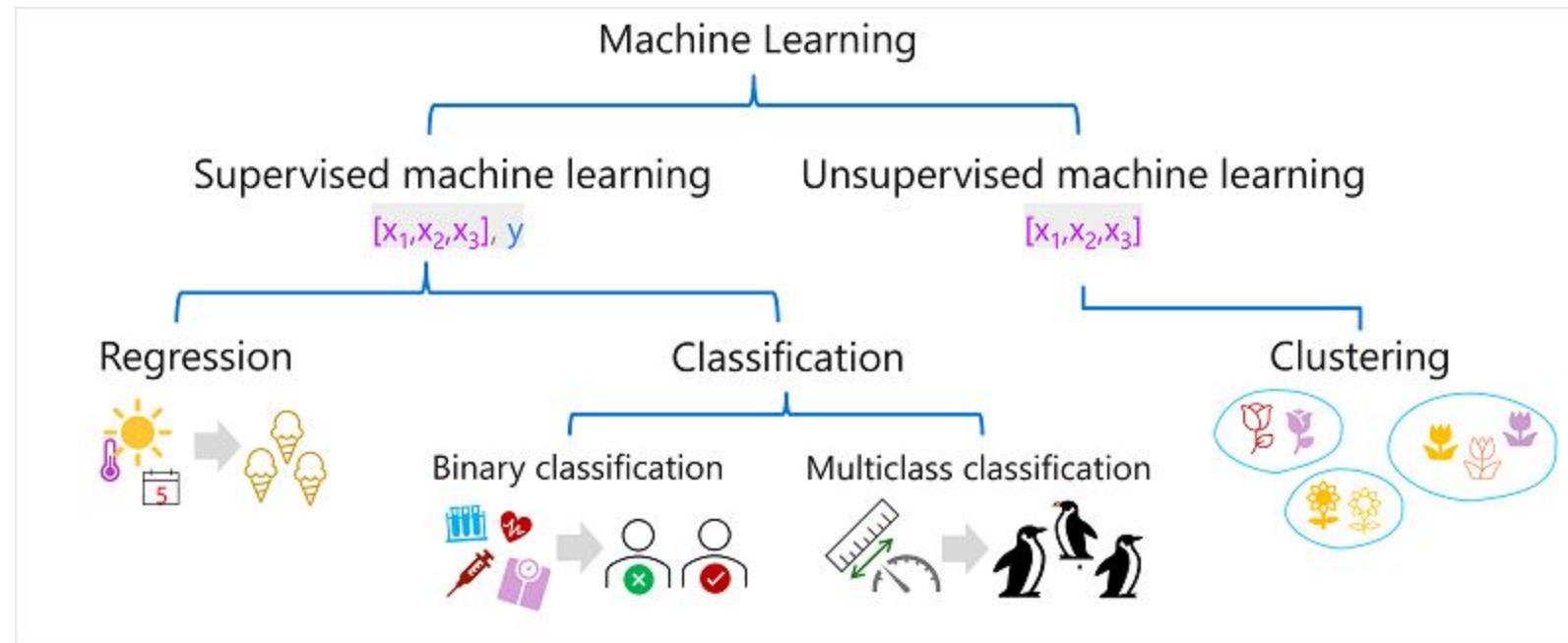
Types of Machine Learning

Machine learning problems are commonly categorized into:

- **Supervised learning:** learn from labeled data
- **Unsupervised learning:** discover structure without labels
- **Semi-supervised learning:** mix labeled and unlabeled data
- **Reinforcement learning:** learn through interaction and reward

This lecture focuses on supervised and unsupervised learning.

Supervised and Unsupervised machine learning



Source: <https://learn.microsoft.com/en-us/training/modules/fundamentals-machine-learning/3-types-of-machine-learning?pivots=text>

„Unsupervised Learning

Unsupervised Learning

Unsupervised learning analyzes data without labels.

The goal is to uncover hidden structure, such as clusters of similar samples or lower-dimensional representations. These methods are often used for exploration, pattern discovery, and preprocessing.

Clustering as Structure Discovery

Clustering groups samples such that points within a cluster are more similar to each other than to points in other clusters.

There is no single correct clustering. Results depend on assumptions about similarity, distance, and cluster shape.

„ KMeans

KMeans Clustering

KMeans is one of the most widely used clustering algorithms due to its simplicity and efficiency.

It partitions data into **K clusters**, where each cluster is represented by its **centroid**, defined as the mean of the points assigned to that cluster.

KMeans Objective Function

KMeans minimizes the **within-cluster sum of squared distances**:

Each point is assigned to the nearest centroid, and centroids are positioned to minimize the average squared distance between points and their assigned centroid.

This objective favors compact, spherical clusters.

The KMeans Algorithm (Step-by-Step)

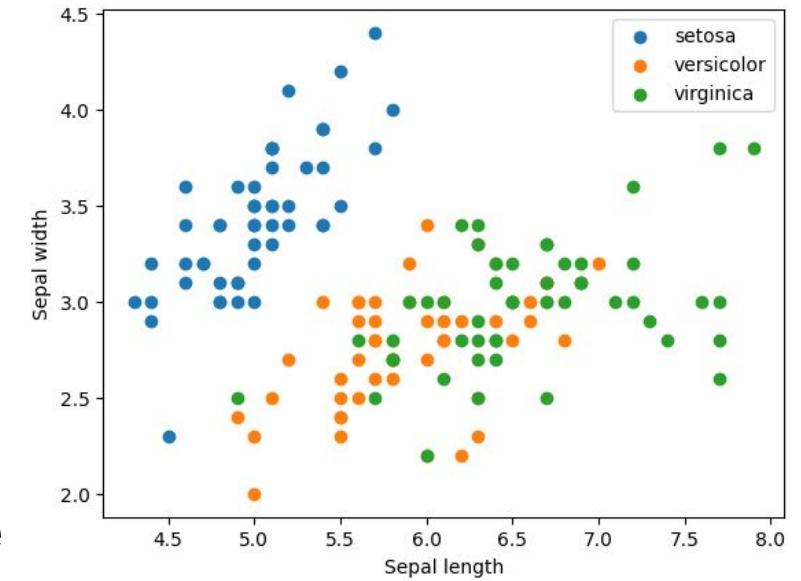
1. Choose the number of clusters K
2. Initialize K centroids (randomly or using heuristics)
3. Assign each point to the nearest centroid
4. Recompute centroids as the mean of assigned points
5. Repeat steps 3–4 until assignments stabilize

The algorithm converges to a local minimum of the objective function.

KMeans on IRIS dataset

The Iris dataset contains 150 samples from three species of iris flowers.

Each sample includes four morphological measurements. Its simplicity and interpretability make it a canonical dataset for demonstrating machine learning algorithms.



Source: By w:ru:Денис Анисимов, Public Domain, <https://commons.wikimedia.org/w/index.php?curid=57593826>

KMeans Clustering on IRIS with K=2

```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import numpy as np

iris = load_iris()
X = iris.data[:, :2] # select first two features

# Fitting KMeans with K=2

kmeans_2 = KMeans(n_clusters=2, random_state=0)
kmeans_2.fit(X)
```

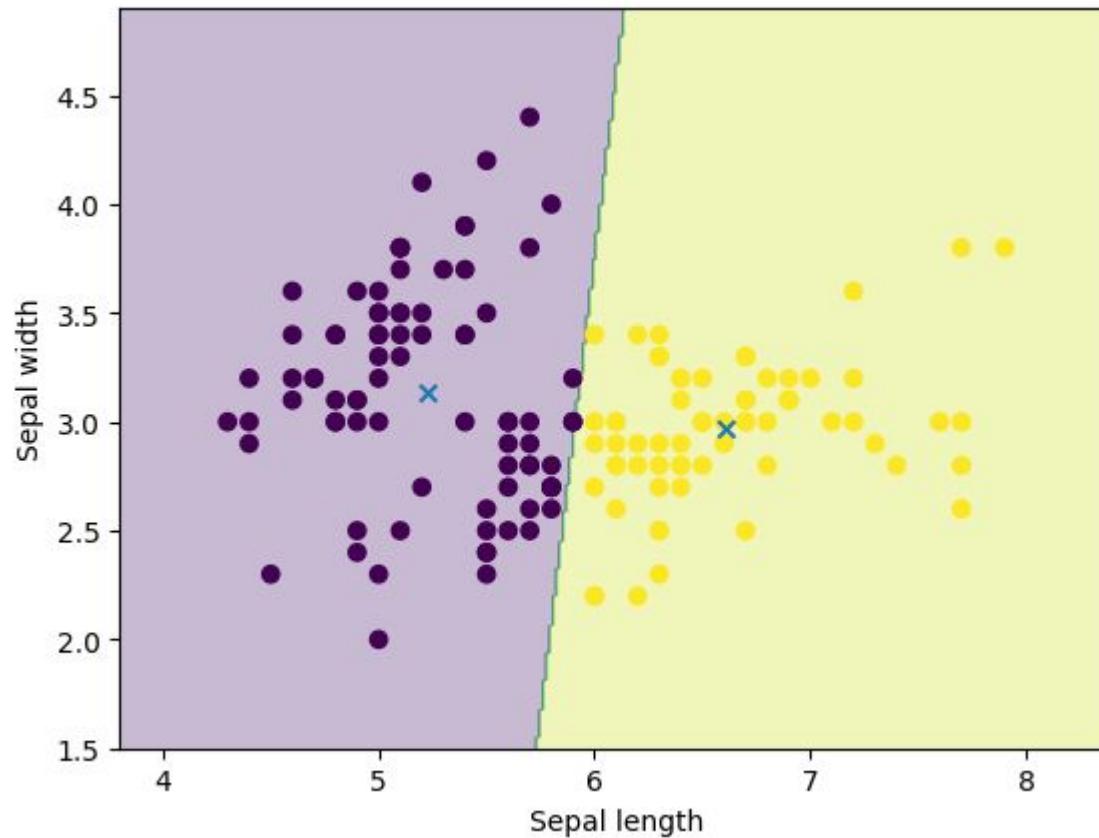
Visualizing Cluster Boundaries

To visualize cluster regions, we generate a grid over feature space and assign each grid point to the nearest centroid.

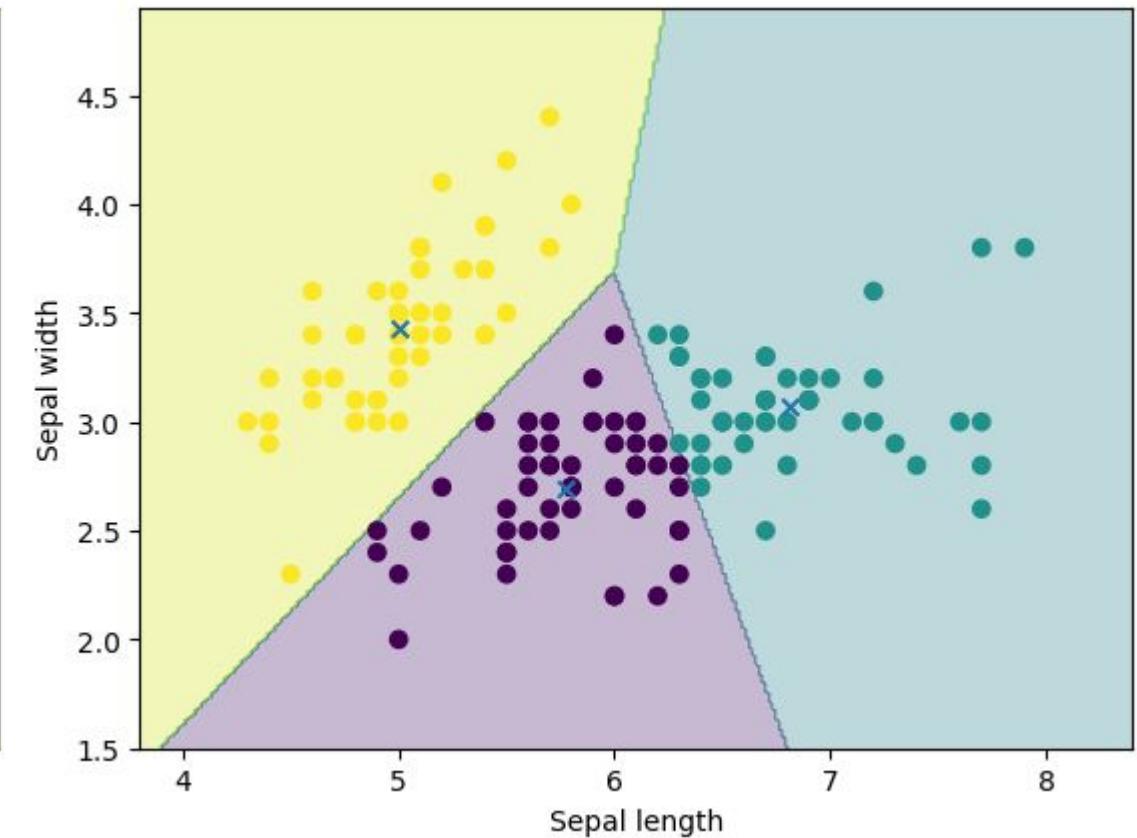
This reveals the implicit decision regions formed by KMeans, even though clustering is unsupervised.

Visualizing Cluster Boundaries

KMeans Clustering (K=2)

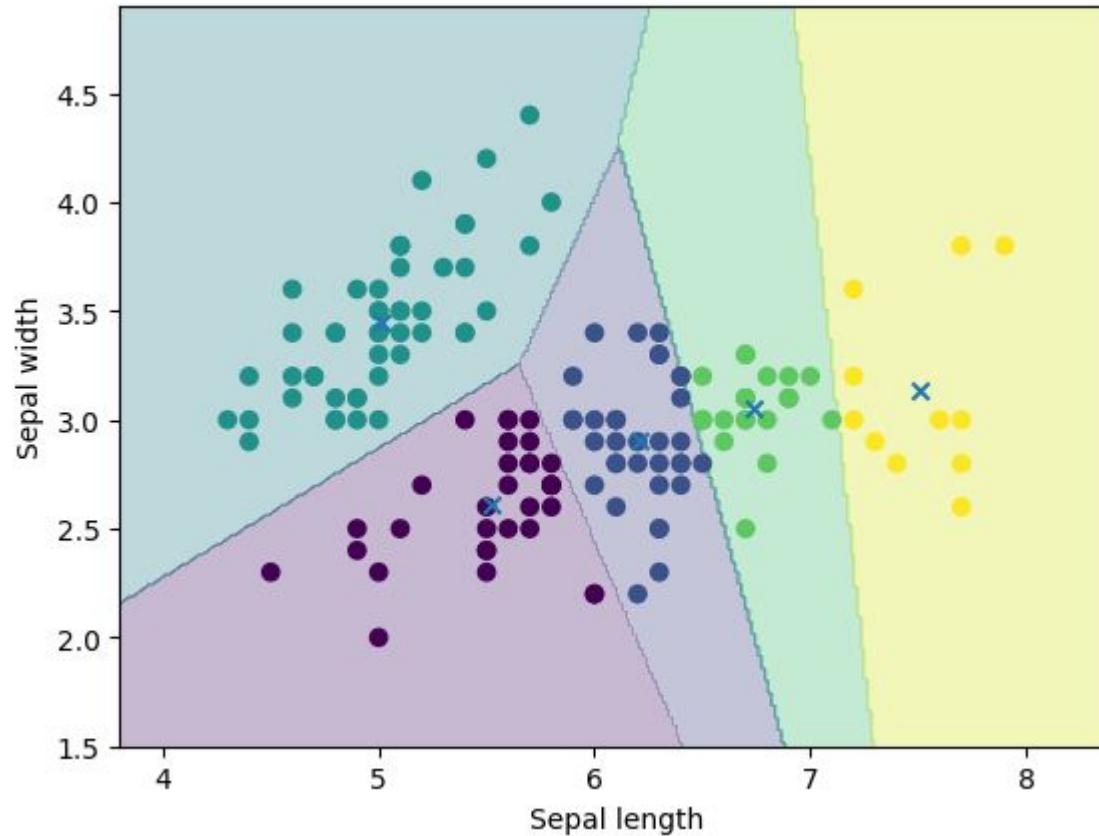


KMeans Clustering (K=3)

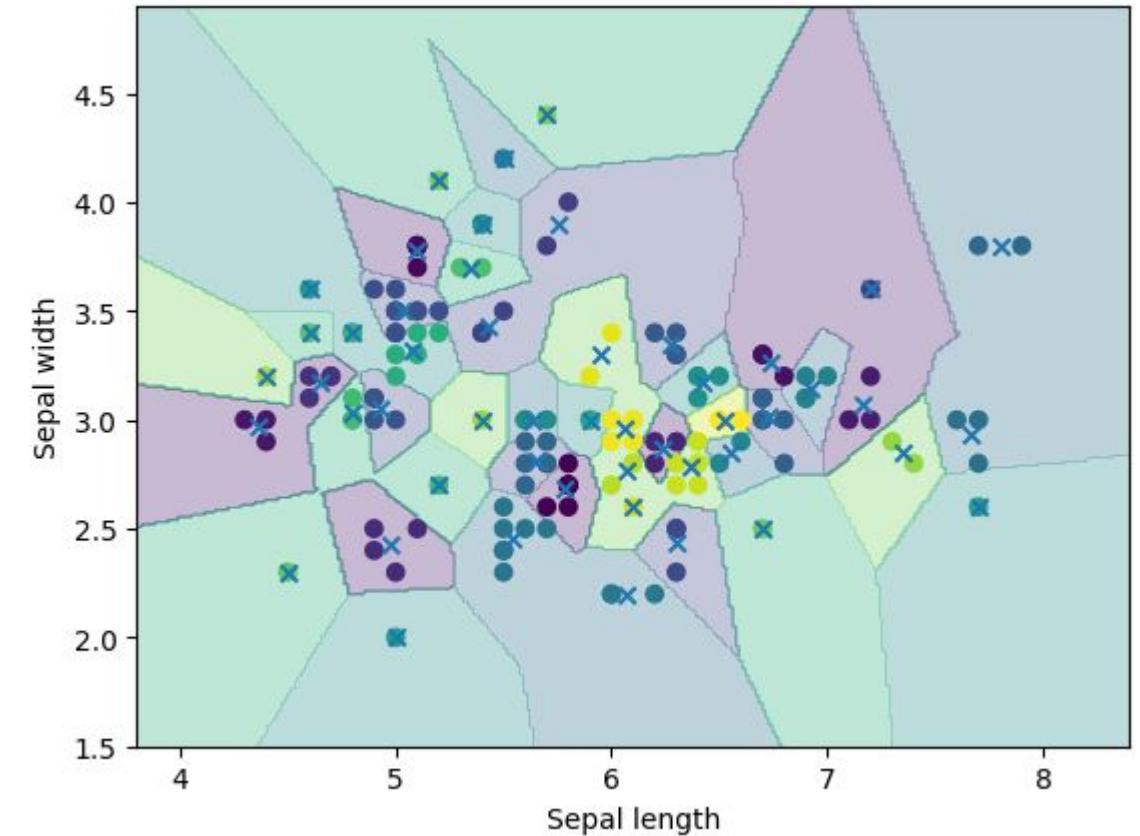


Visualizing Cluster Boundaries

KMeans Clustering (K=5)



KMeans Clustering (K=50)

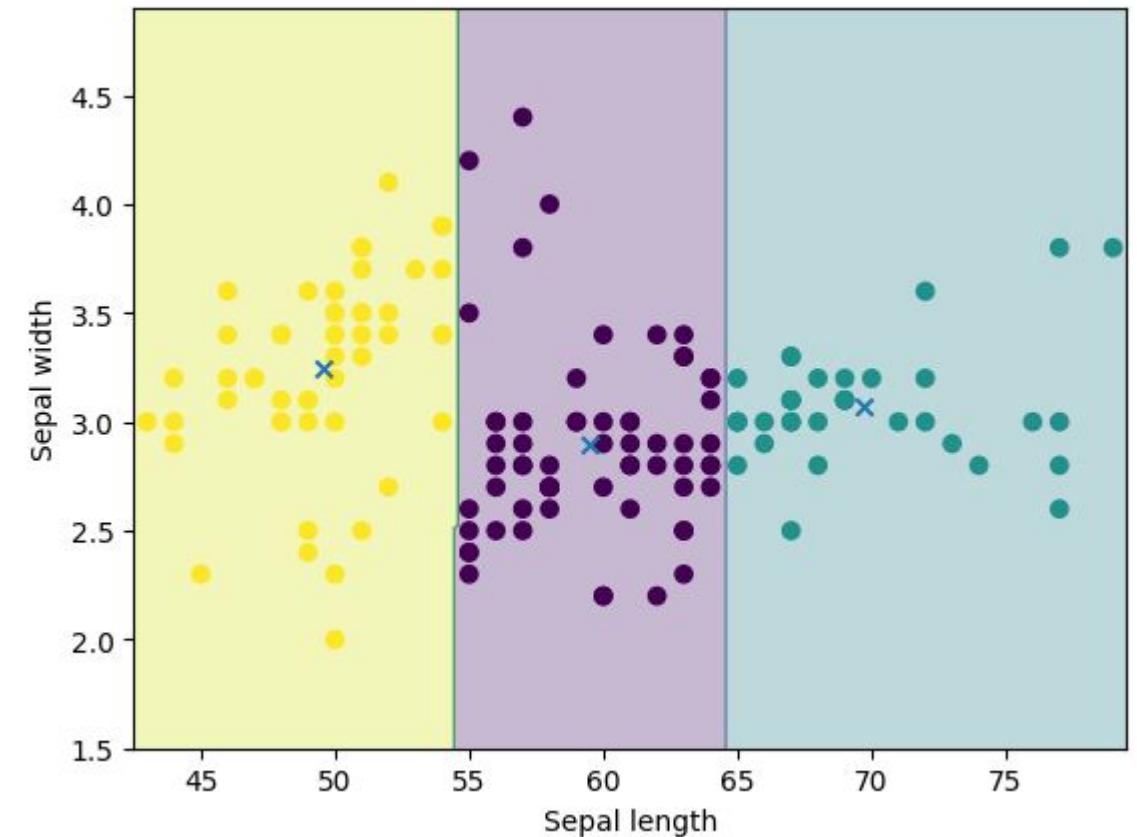


Increasing K further subdivides existing clusters, increasing granularity but potentially reducing semantic meaning. There is no universally correct K.

Kmean's Sensitivity to Feature Scaling (K=3)

```
# Sensitivity to Feature Scaling  
X_sca = X.copy()  
X_sca[:,0] = X_sca[:,0] * 10
```

```
kmeans = KMeans(n_clusters=3,  
                 random_state=0)  
kmeans.fit(X_sca)  
plot_Kmeans_clusters(X_sca,  
                      kmeans)
```



Summary of KMeans Clustering

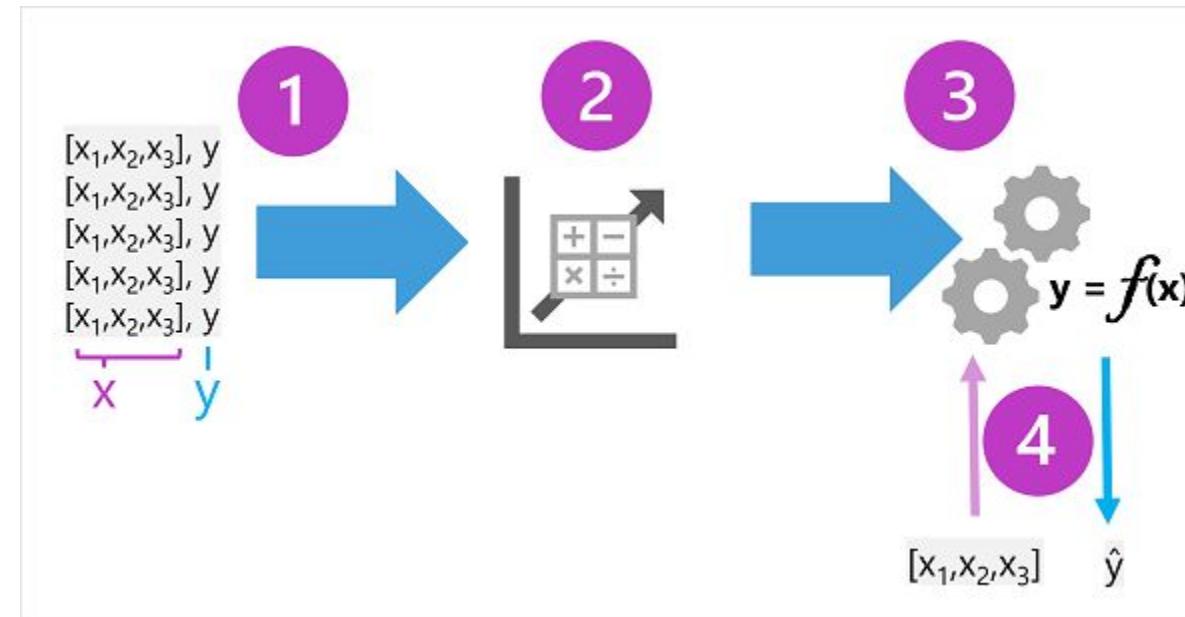
- KMeans is appropriate when clusters are compact, and well-separated.
- It is often used for exploratory analysis, data compression, and as a preprocessing step for other machine learning tasks.
- KMeans provides an intuitive introduction to unsupervised learning and geometric partitioning of feature space.
- Understanding its assumptions and limitations is essential before applying it to real-world data.

„Supervised Learning

Supervised Learning

Supervised learning trains a model using input–output pairs.

The model learns a function that maps features to known outputs and is evaluated on its ability to predict labels for new, unseen data. Classification and regression are the two primary supervised learning tasks.



Classification and Regression

Classification predicts discrete categories, such as species type or land cover class.

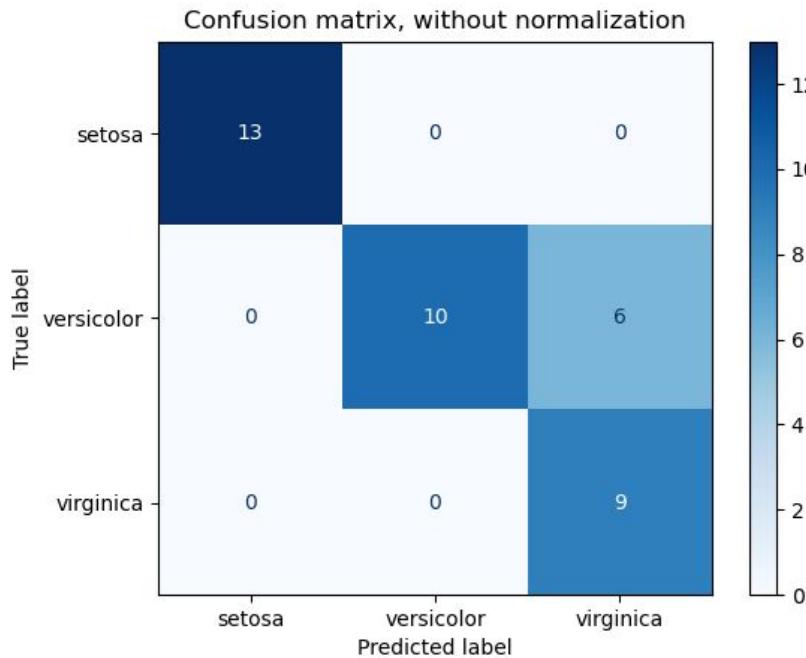
Regression predicts continuous values, such as temperature, biomass, or price.

Although both rely on similar principles, they differ in output type, loss functions, and evaluation metrics.

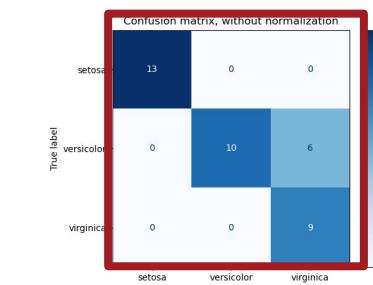
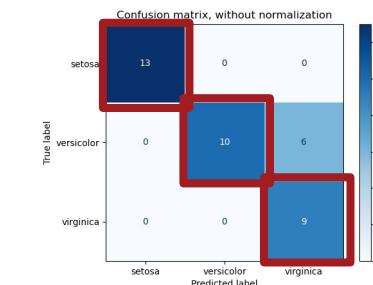
This lecture focuses on classification.

Evaluating a classification model: Confusion matrix and accuracy

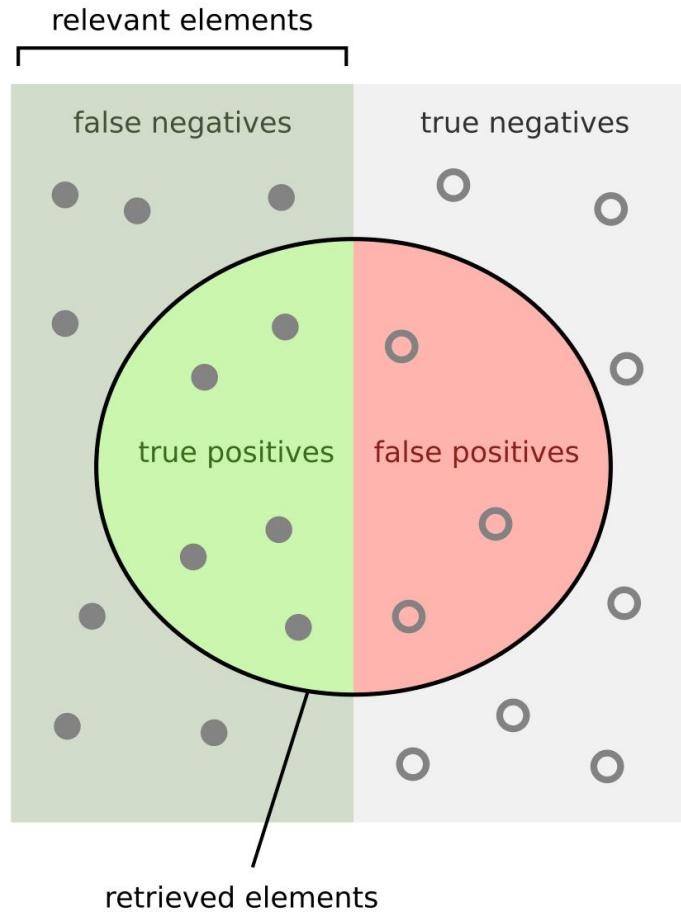
Confusion matrix: a square matrix that compares true classes with the predicted classes.



Accuracy: proportion of correctly classified instances to the total number of instances.



Precision and Recall



$$\text{Precision} = \frac{\text{Relevant retrieved instances}}{\text{All retrieved instances}}$$

$$\text{Recall} = \frac{\text{Relevant retrieved instances}}{\text{All relevant instances}}$$

How many retrieved items are relevant?

$$\text{Precision} = \frac{\text{green}}{\text{green} + \text{red}}$$

How many relevant items are retrieved?

$$\text{Recall} = \frac{\text{green}}{\text{green} + \text{blue}}$$

Generalization in Machine Learning Models

- **Generalization** describes how well a classifier performs on unseen data from the same distribution.
- **Unbiased estimation** of performance requires that evaluation data remain independent from model training and selection.
- Hyperparameter tuning, feature engineering, and model comparison must be confined to **training and validation data**.
- Aggregated metrics computed on a held-out test set then provide an unbiased estimate of true generalization performance.
- **Without this separation, reported results tend to be overly optimistic and unreliable!**

„ k-Nearest Neighbors

k-Nearest Neighbors: A Supervised Perspective

- k-Nearest Neighbors (KNN) is a supervised learning algorithm that predicts the label of a sample based on the labels of nearby samples in feature space.
- Unlike many models, KNN performs no explicit training beyond storing the dataset. All computation occurs at prediction time.
- KNN assumes that samples that are close to each other in feature space tend to share the same label.
- Prediction is based on a local majority vote among the K closest labeled samples, where closeness is defined by a distance metric.

KNN Compared to KMeans

KNN and KMeans both rely on distance in feature space.

KMeans is unsupervised and learns centroids, while KNN is supervised and relies directly on labeled samples. KNN does not create clusters; it uses proximity for classification.

KNN Classification on IRIS with K=2

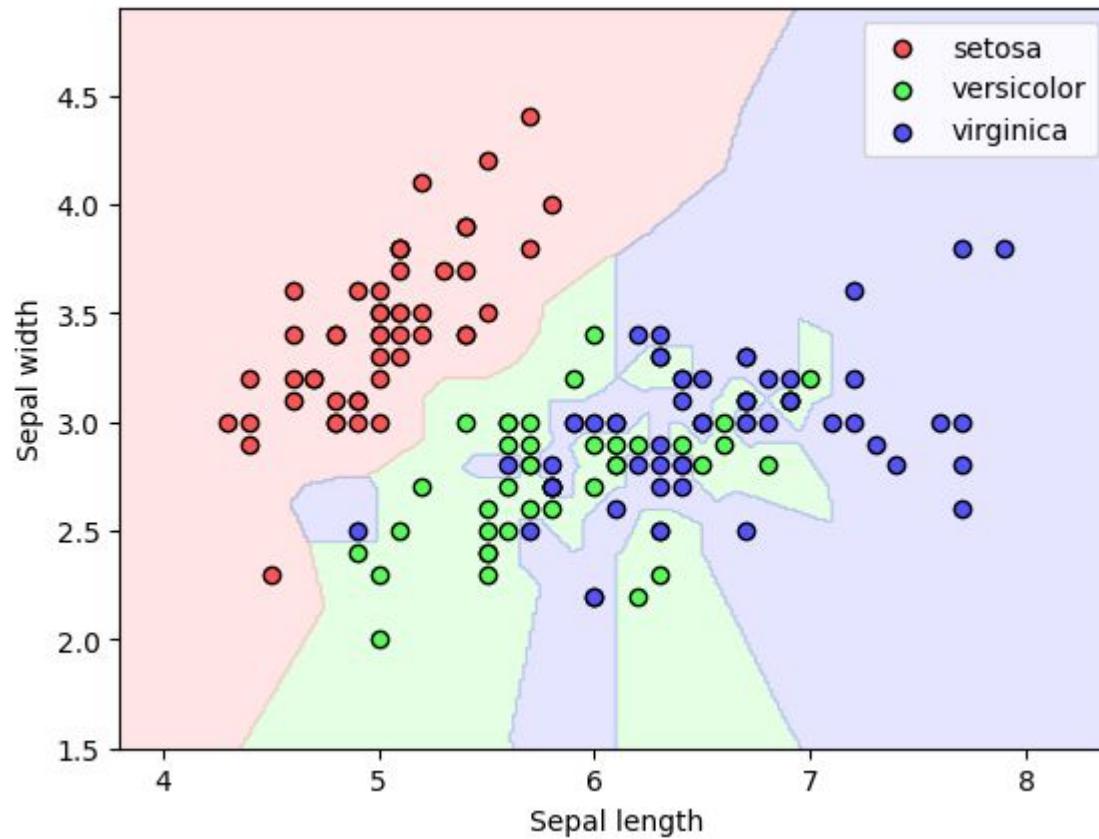
```
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
import numpy as np

iris = load_iris()
X = iris.data[:, :2] # select first two features
y = iris.target

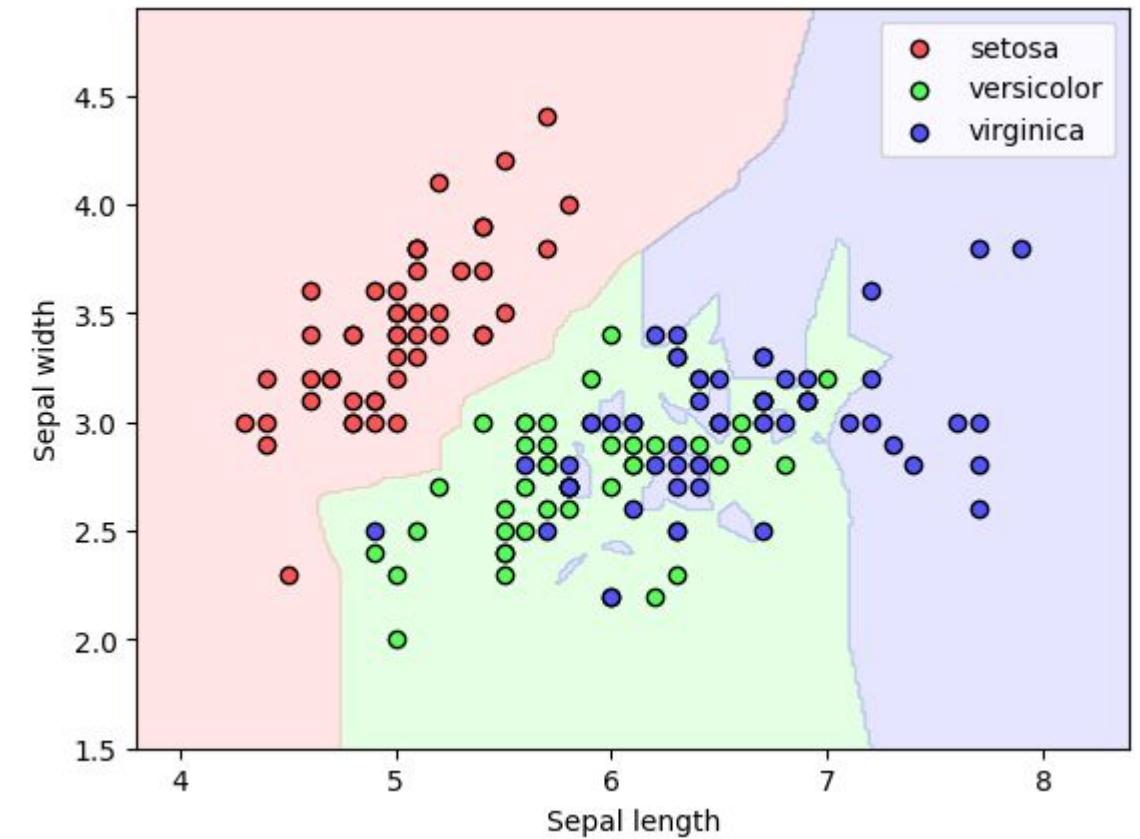
# Fitting KNNs with K=2
knn = KNeighborsClassifier(n_neighbors=2)
knn.fit(X, y)
```

Visualizing Decision Boundaries

KNN Classification (K=1)

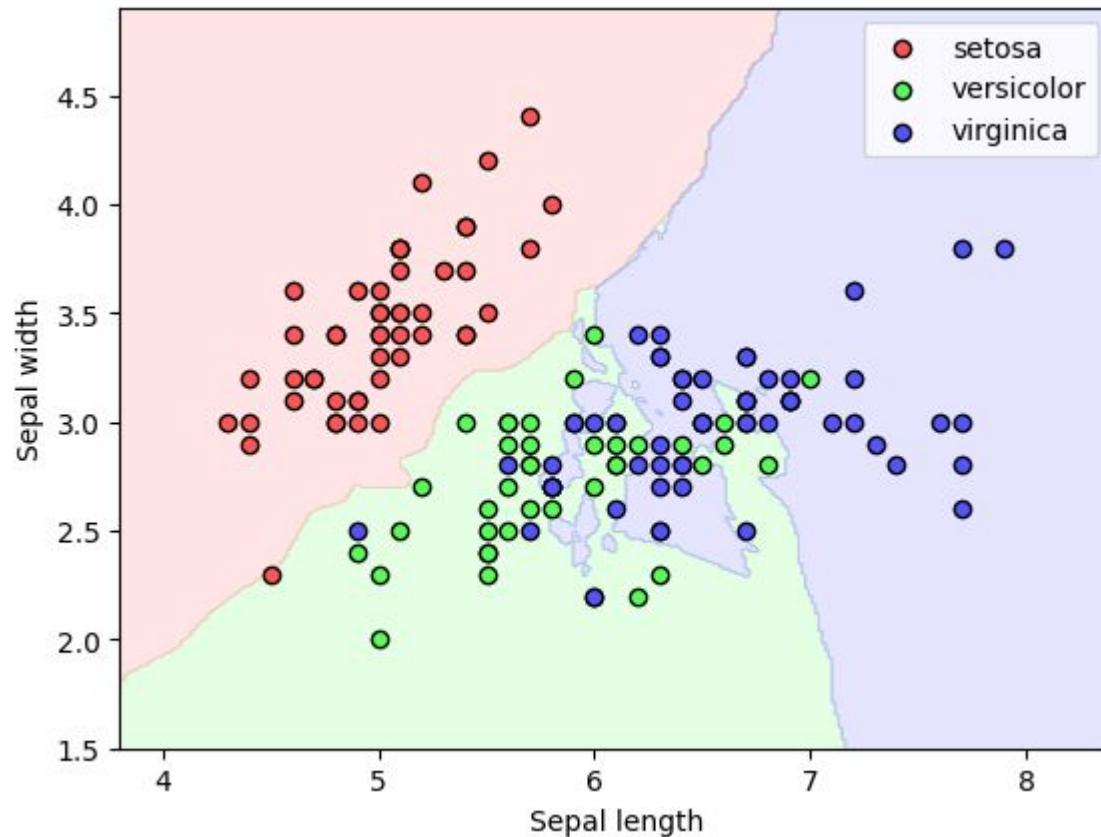


KNN Classification (K=2)

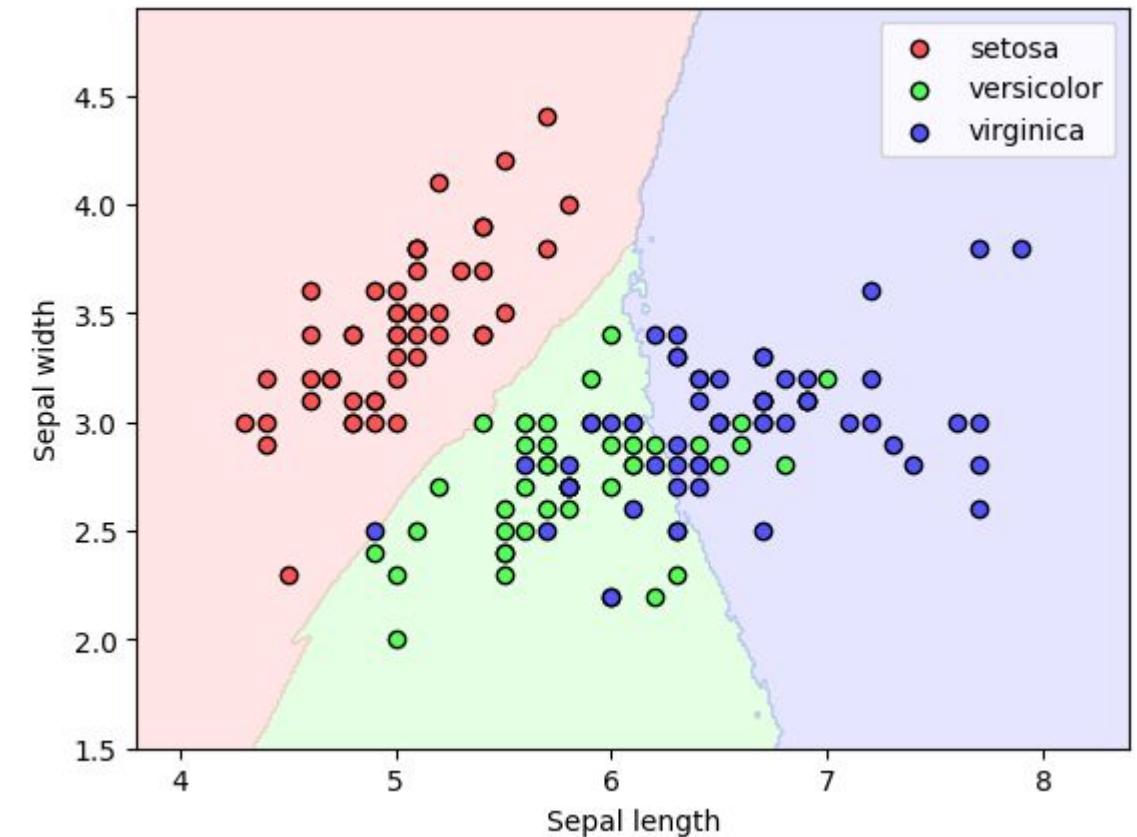


Visualizing Decision Boundaries

KNN Classification (K=5)



KNN Classification (K=50)



Small K values produce low bias and high variance. Large K values produce higher bias and lower variance. Selecting K balances flexibility and generalization.

Summary of KNN Classification

- KNN requires storing the full training dataset. Prediction involves computing distances to all stored samples, making KNN computationally expensive for large datasets.
- Therefore KNN is effective for small datasets with meaningful distance metrics and well-scaled features.
- It is less suitable for high-dimensional or large-scale problems.
- Increasing K smooths decision boundaries by incorporating more neighbors into each prediction.
- This reduces sensitivity to noise but may blur fine-grained structure in the data.

Transition to Parametric Models

We now move to models that learn explicit parameters during training.

These models summarize the data through learned parameters, enabling faster prediction and improved scalability.

Linear Classification

Linear classifiers separate classes using linear decision boundaries.

They assume that classes can be separated by a weighted combination of input features.

„ Logistic Regression

Linear Classification with Logistic Regression

Logistic regression models the probability of class membership using a linear function passed through a sigmoid or softmax transformation.

Despite its name, logistic regression is a classification algorithm.

Logistic Regression on IRIS

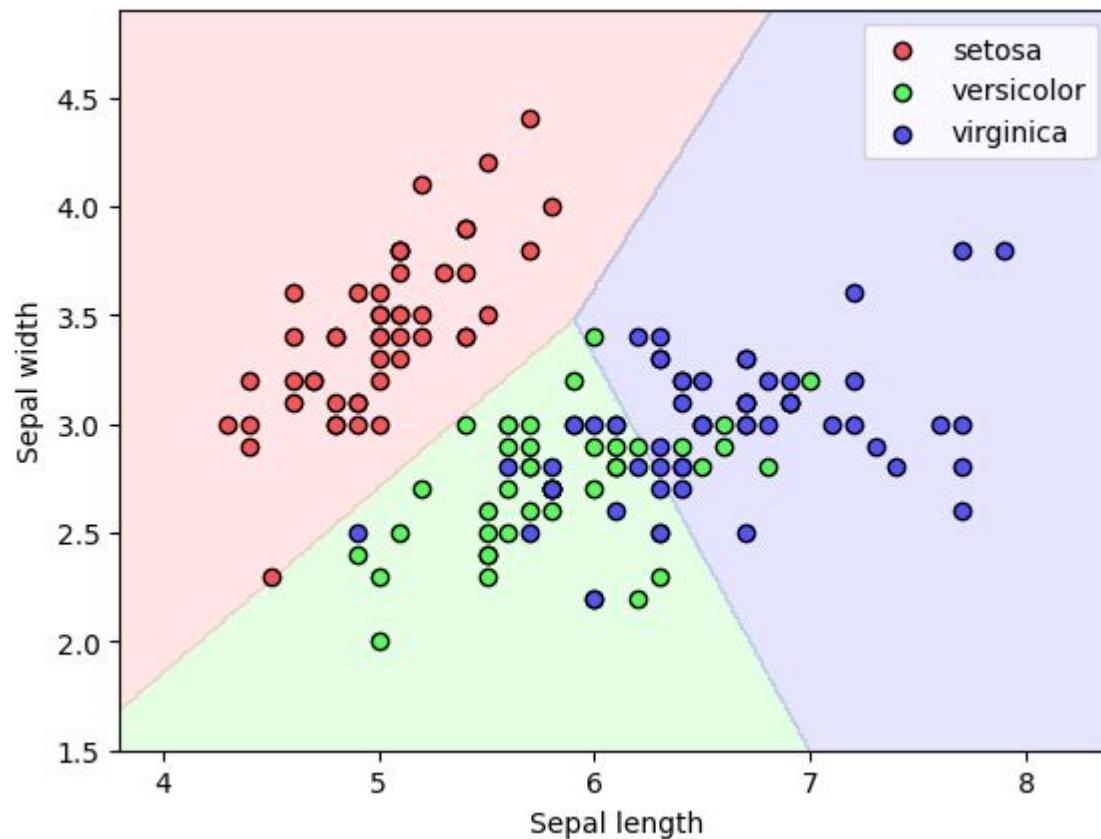
```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
import numpy as np

iris = load_iris()
X = iris.data[:, :2] # select first two features
y = iris.target

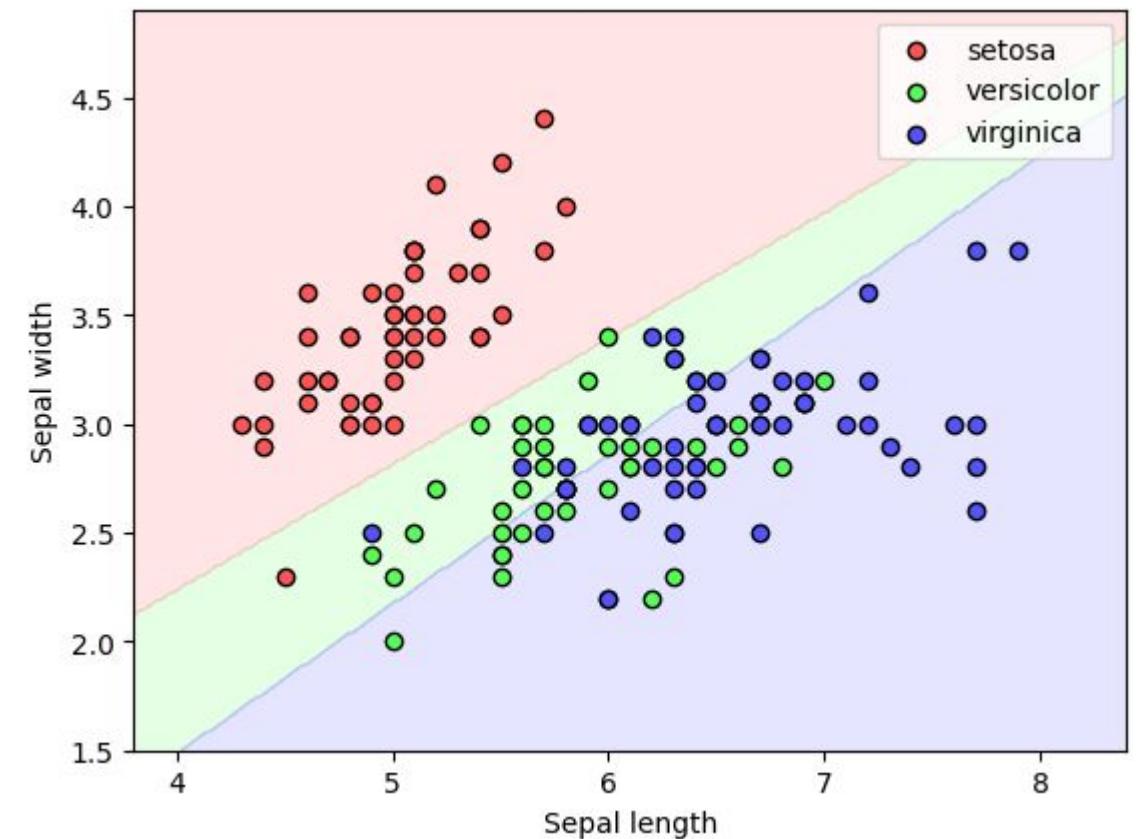
# Fitting LogisticRegression
logistic_reg = LogisticRegression()
logistic_reg.fit(X, y)
```

Visualizing Decision Boundaries

Logistic Regression



Logistic Regression (max_iter=10)



Model did not converge in 10 steps!

Logistic Regression – High-Level Overview

$$\hat{y}(w, x) = \text{sigmoid}(w_0 + w_1 x_1 + \dots + w_p x_p)$$

Training:

- Learn coefficients (w) that maximize the likelihood of observed outcomes.
- Uses gradient descent or other optimization algorithms.

Characteristics:

- **Interpretable:** Coefficients indicate feature influence.
- **Linear Decision Boundary:** Separates classes with a straight line (or hyperplane).
- **Probabilistic Output:** Supports thresholding for classification and confidence estimates.

Summary of Logistic Regression

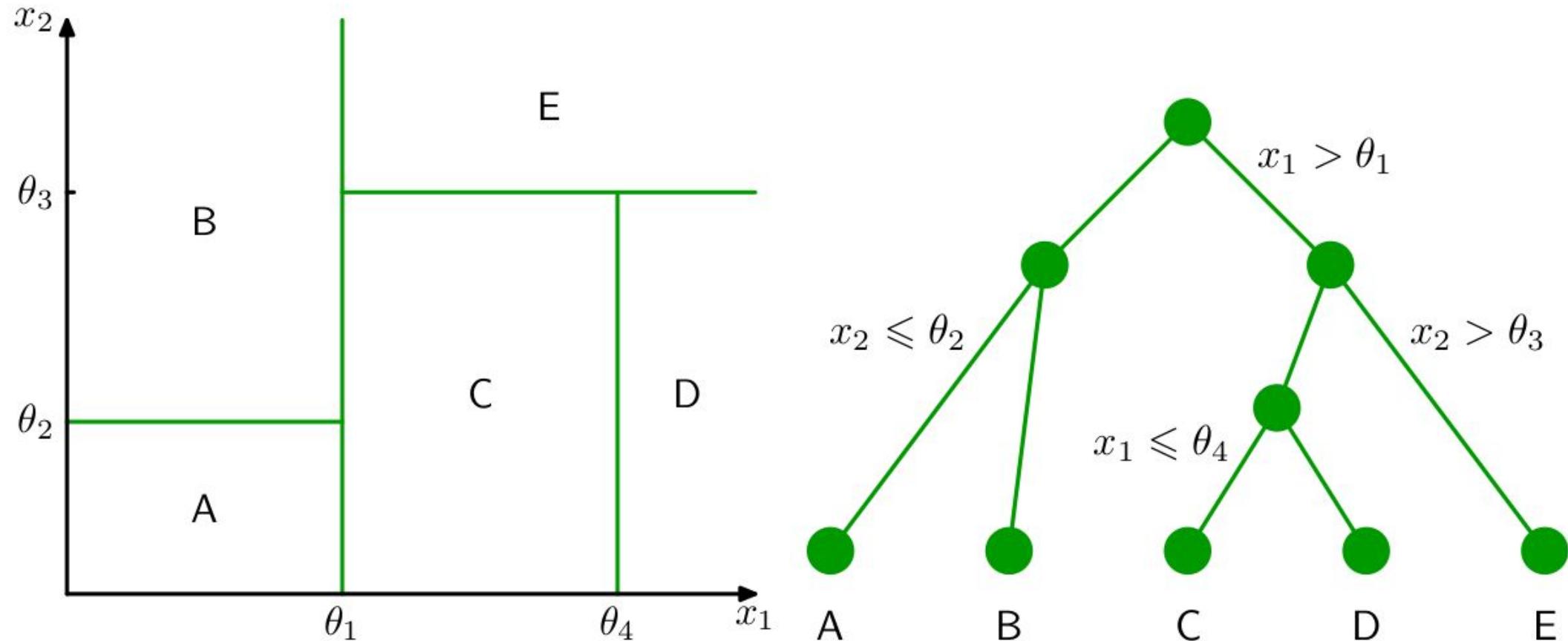
- The decision boundary of a linear classifier is a straight line in two dimensions or a hyperplane in higher dimensions. This simplicity provides stability and interpretability.
- Model coefficients indicate how strongly each feature influences the prediction.
- Linear classifiers, such as logistic regression, cannot represent nonlinear class boundaries. When class structure is complex, linear models underfit the data.

„ Decision Trees

Classification with Decision Trees

- Tree-based models are simple but powerful and are human interpretable.
- Tree-based methods partition the feature space into rectangular regions and assign a simple model, usually a constant value/label, to each region .
- Several tree-based methods exist (C4.5, ID3, ...), we will focus on CART (Classification and Regression Trees). CART trees are binary trees.

Decision Trees



Decision Trees on IRIS

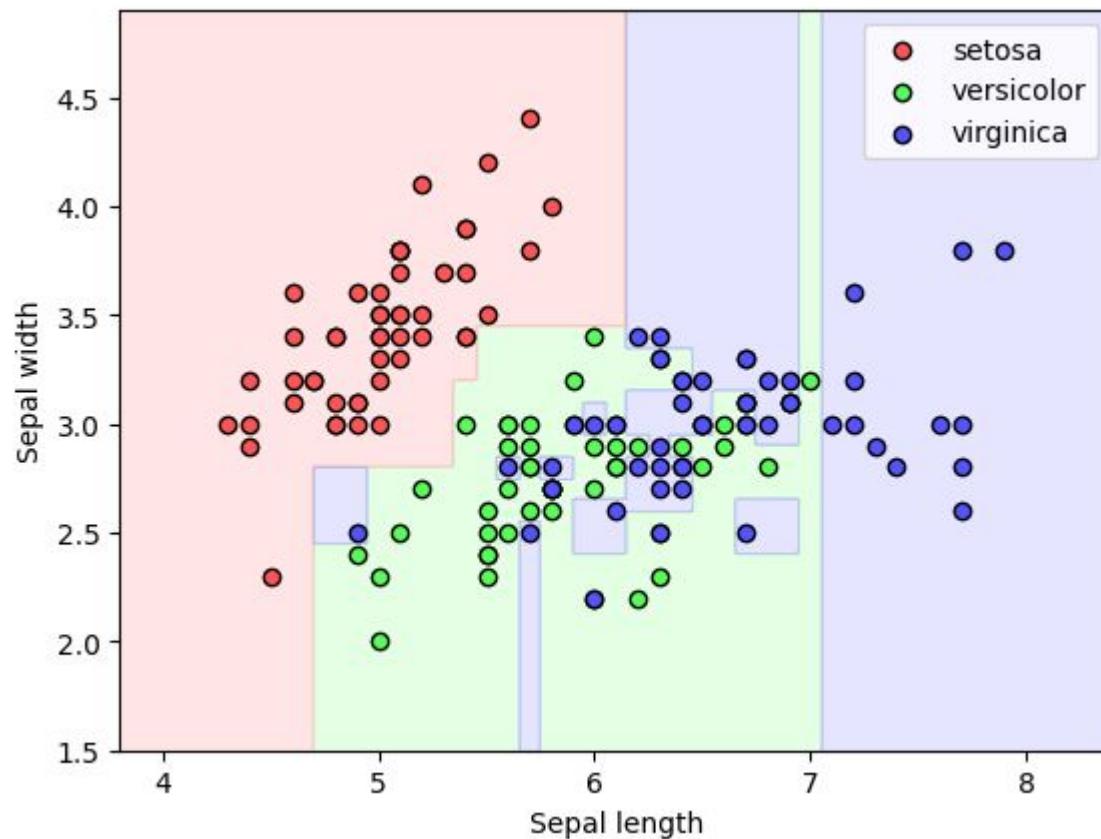
```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
import numpy as np

iris = load_iris()
X = iris.data[:, :2] # select first two features
y = iris.target

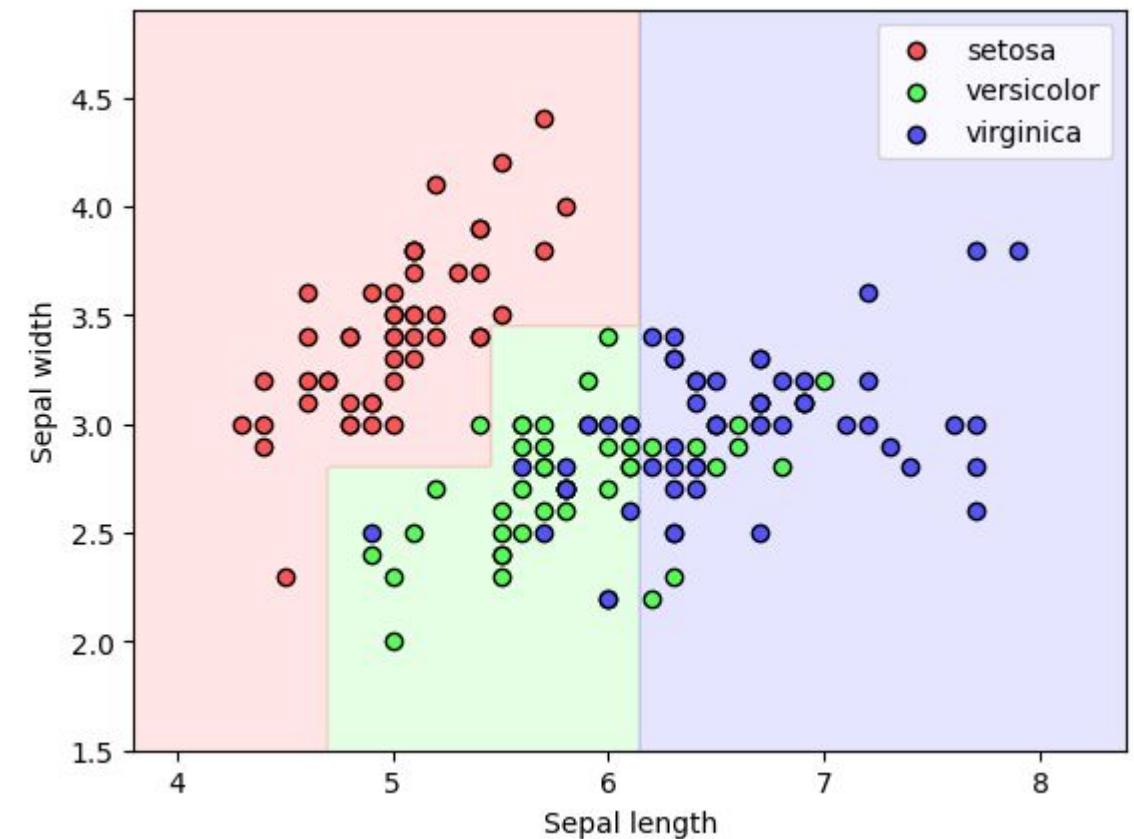
# Fitting LogisticRegression
tree = DecisionTreeClassifier()
tree.fit(X, y)
```

Visualizing Decision Boundaries

Decision Trees



Decision Trees (max_depth=3)



max_depth=3 leads to a simpler tree!

Decision Tree evaluation, basic idea

To evaluate a tree given an input x :

- Start at the root node
- Each inner node corresponds to some if-then rule assigning x to one of its children, e.g.:
 - if $x_d < Q$ then goto left child node,
 - else goto the right child node.
- When a leaf node is reached, x is assigned to the value or label (or distribution over labels) associated with that leaf node.
- The leaf nodes define the regions in the feature space.

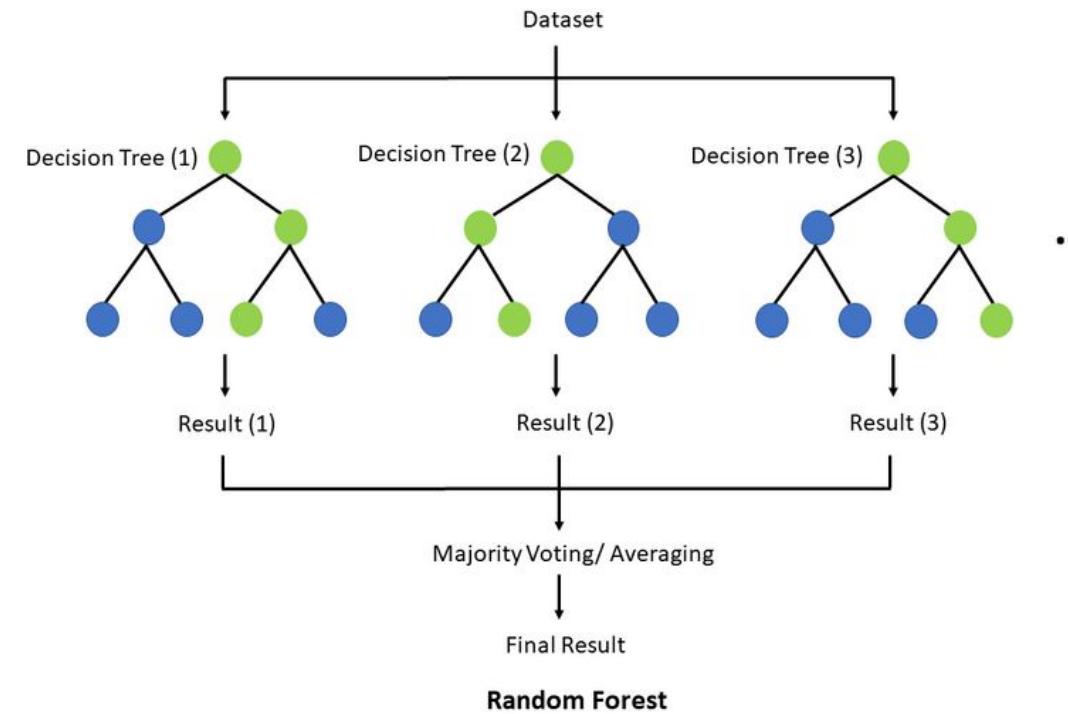
Summary of Decision Trees

- Decision trees produce axis-aligned decision boundaries. Shallow trees yield coarse partitions, while deeper trees create complex, fragmented regions.
- Unrestricted trees can perfectly fit training data but generalize poorly. Regularization through depth limits, pruning, or minimum leaf size improves robustness.
- Decision trees handle nonlinear relationships and feature interactions naturally. They remain interpretable and form the basis of powerful ensemble methods.

„ Random Forests

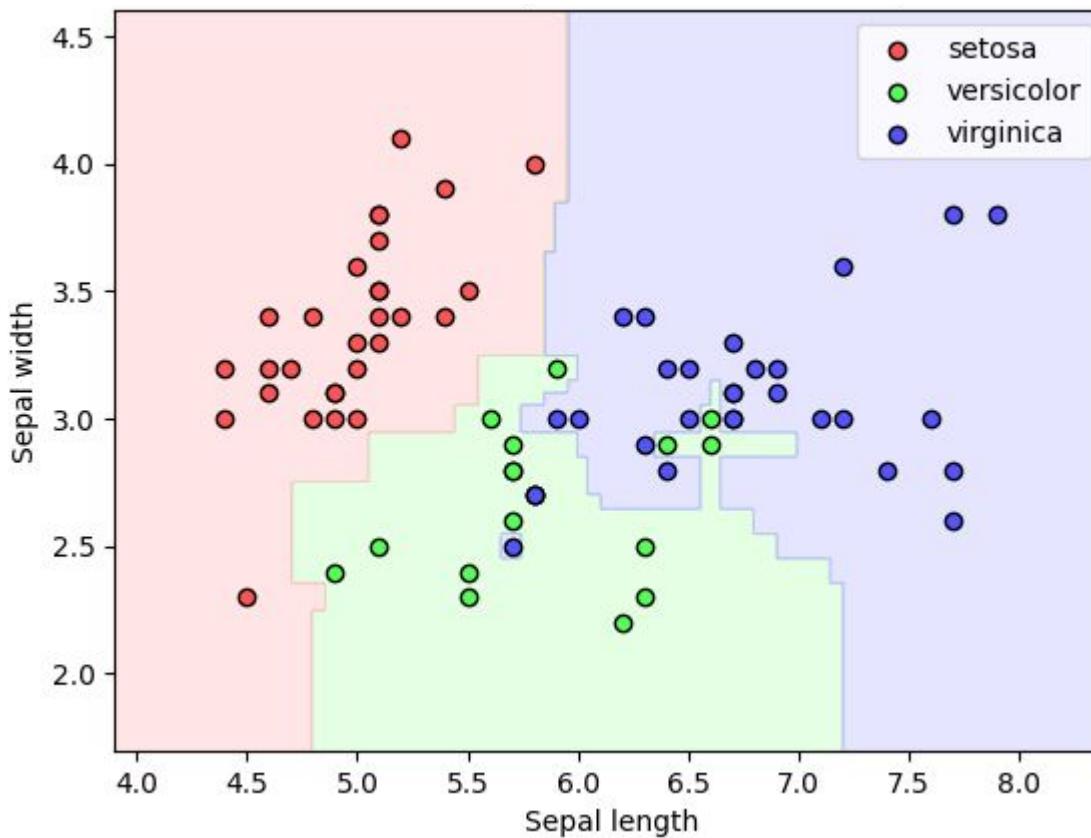
Random Forests

Random forests is an ensemble learning method works by creating a multitude of decision trees during training.

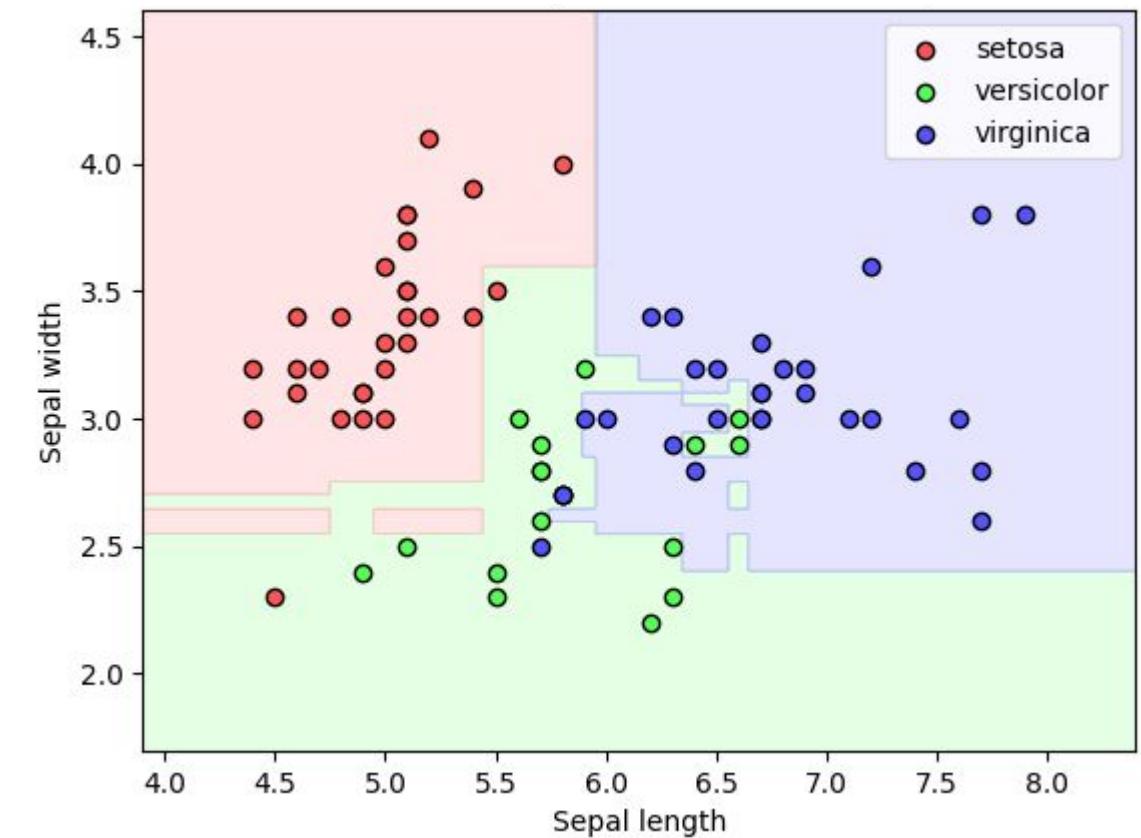


Visualizing Decision Boundaries

Random Forest



Random Forest (n_estimators=3)



Further resources:

The Microsoft training module: [Introduction to machine learning concepts](#) offers a video-based learning module on these topic.

Further readings:

- Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4). Chapter 9.1. K-means clustering. 7 pages.
- Géron, A. (2022). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc. Chapter 4. Logistic Regression. 9 pages.
- Lindholm, A., Wahlström, N., Lindsten, F., & Schön, T. B. (2022). Machine learning: a first course for engineers and scientists. Cambridge University Press. Chapter 1: Introduction. 13 pages.
- Lindholm, A., Wahlström, N., Lindsten, F., & Schön, T. B. (2022). Machine learning: a first course for engineers and scientists. Cambridge University Press. Chapter 2. Supervised Learning: A First Approach. 23 pages.

”

Thank you!