wtech
Teknolojide Kadın Derneği

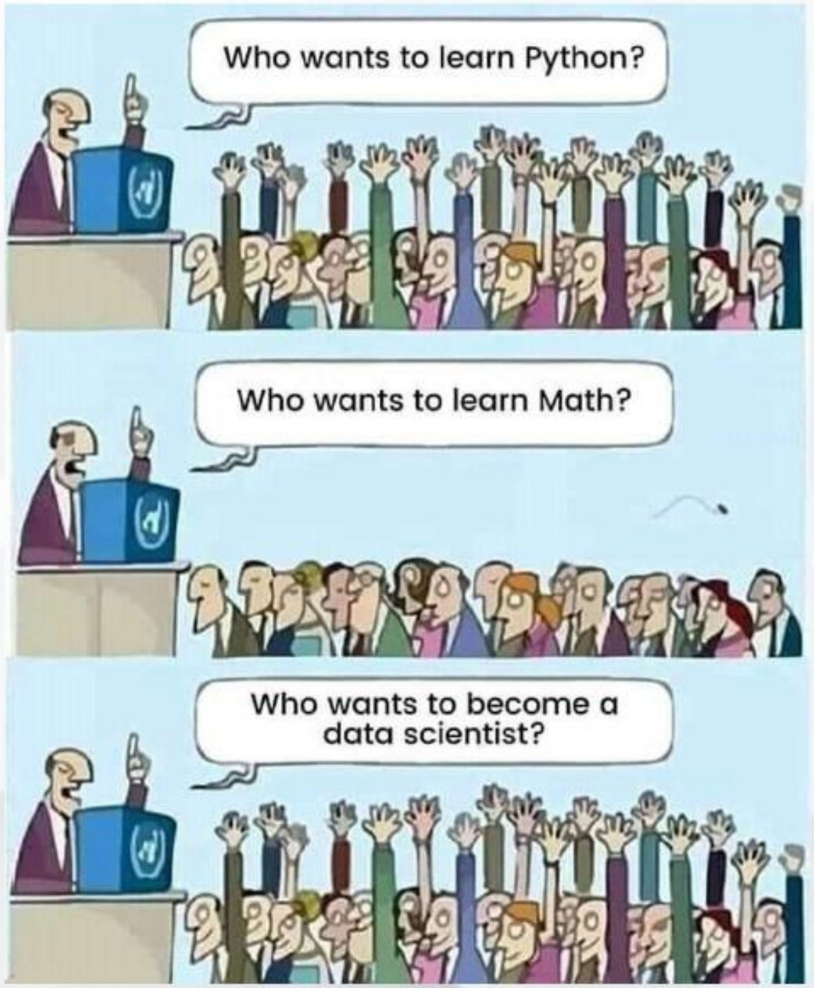
Introduction to Machine Learning with Python

ML Fundamentals

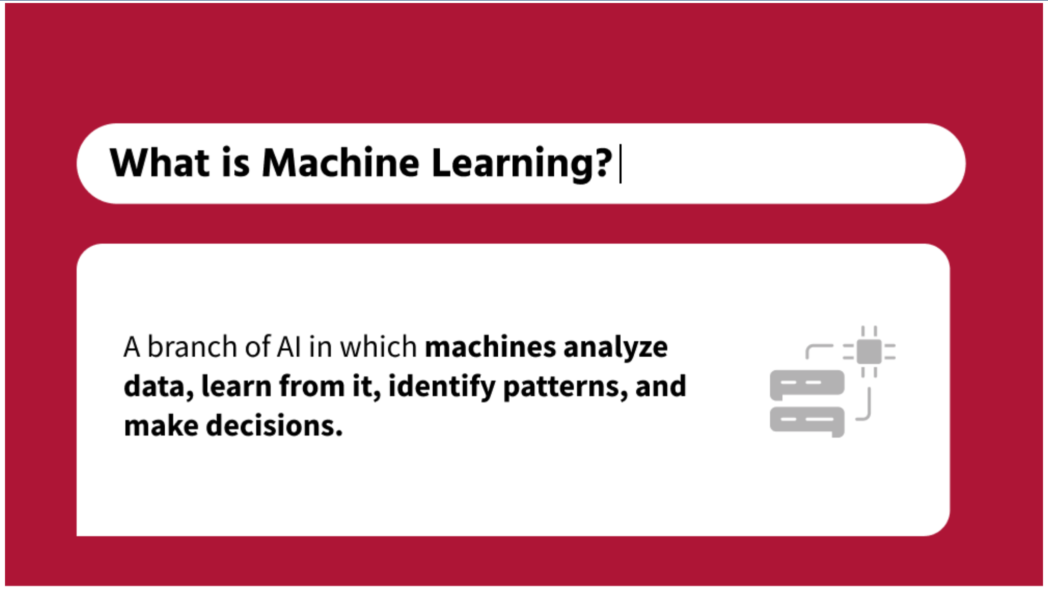
Dr. Süha Tuna
İTÜ Informatics Institute

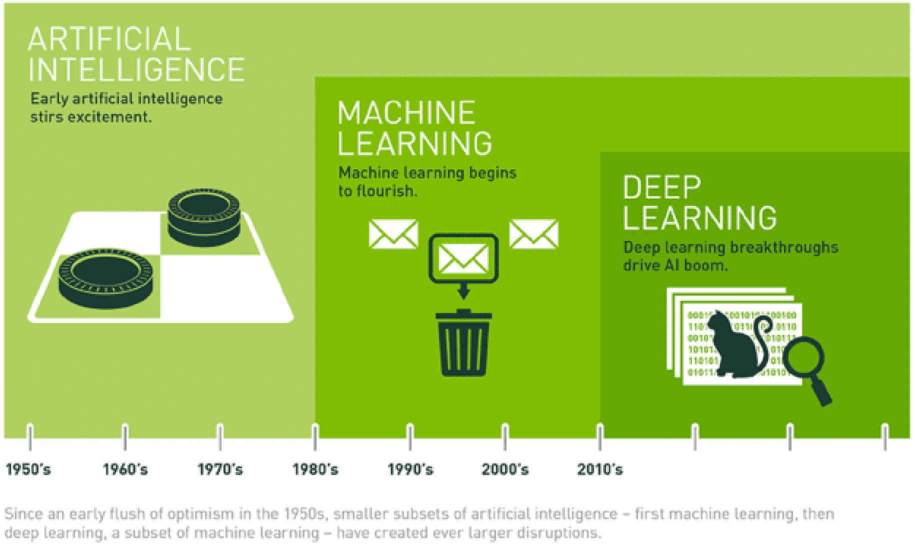
Before start

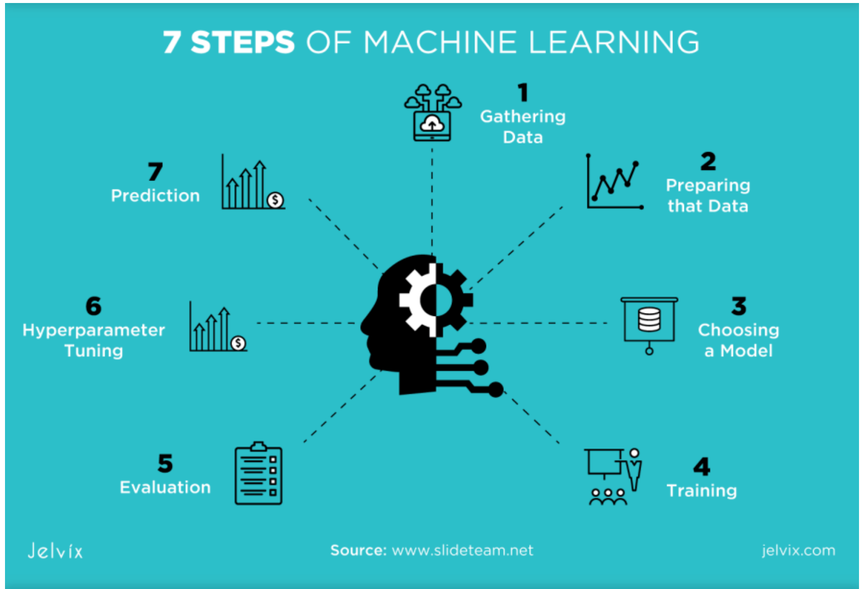
- Linear Algebra
  - Vectors, matrices
  - Eigenvalues, eigenvectors
  - Linear equation solvers
- Optimization
  - Gradients
  - Cost functions
- Statistics & Probability
  - Probability distributions
  - Likelihoods
  - Statistics



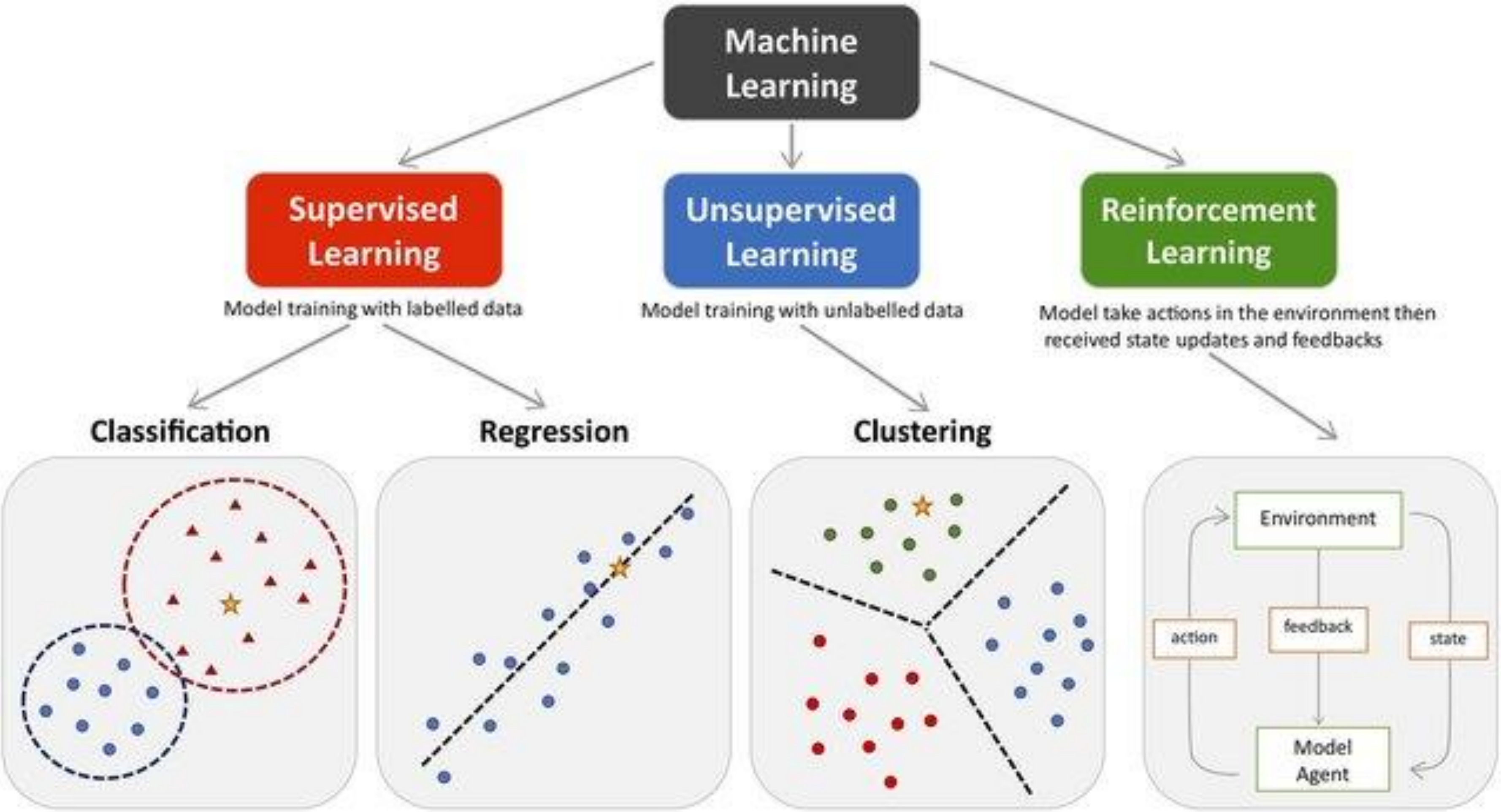
What is ML?


What is Machine Learning?
A branch of AI in which **machines analyze data, learn from it, identify patterns, and make decisions.**


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.


7 STEPS OF MACHINE LEARNING
1 Gathering Data
2 Preparing that Data
3 Choosing a Model
4 Training
5 Evaluation
6 Hyperparameter Tuning
7 Prediction
Jelvix
Source: www.slideteam.net
jelvix.com

ML Problem Types



Bias-Variance Tradeoff

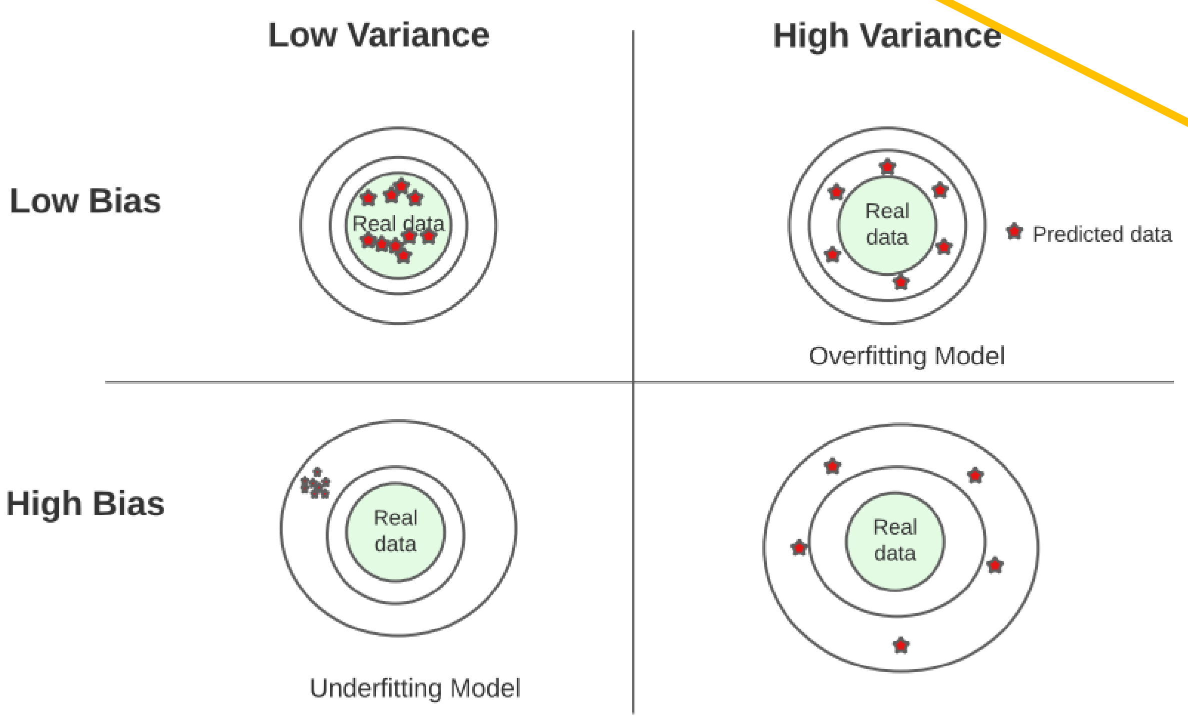
- A good model aims to **balance bias** and **variance** to **minimize total error**.

- **Total Error = Bias² + Variance + Irreducible Error**

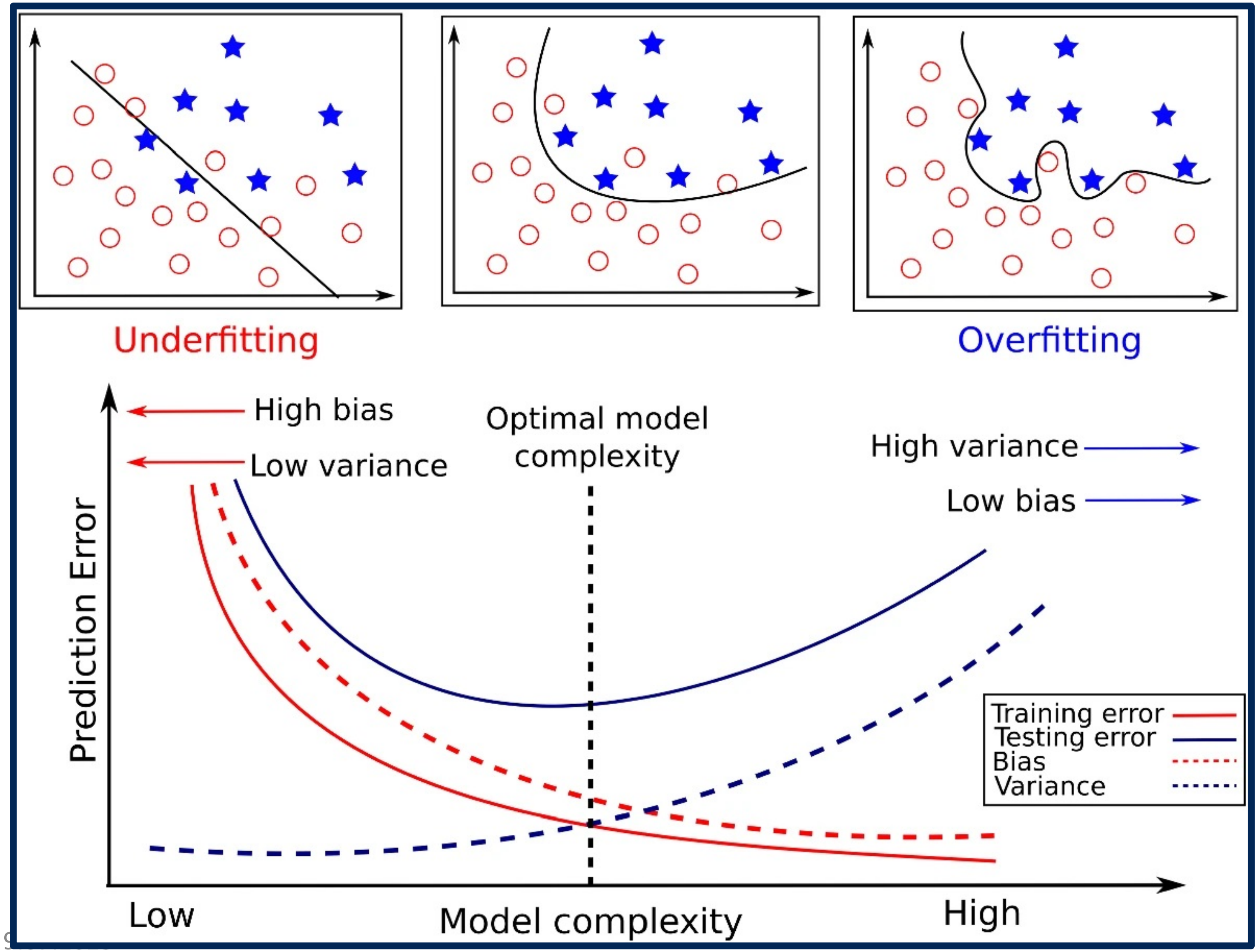
The error occured from the model simplicity

The error occured from the model complexity

Noise coming from the observations

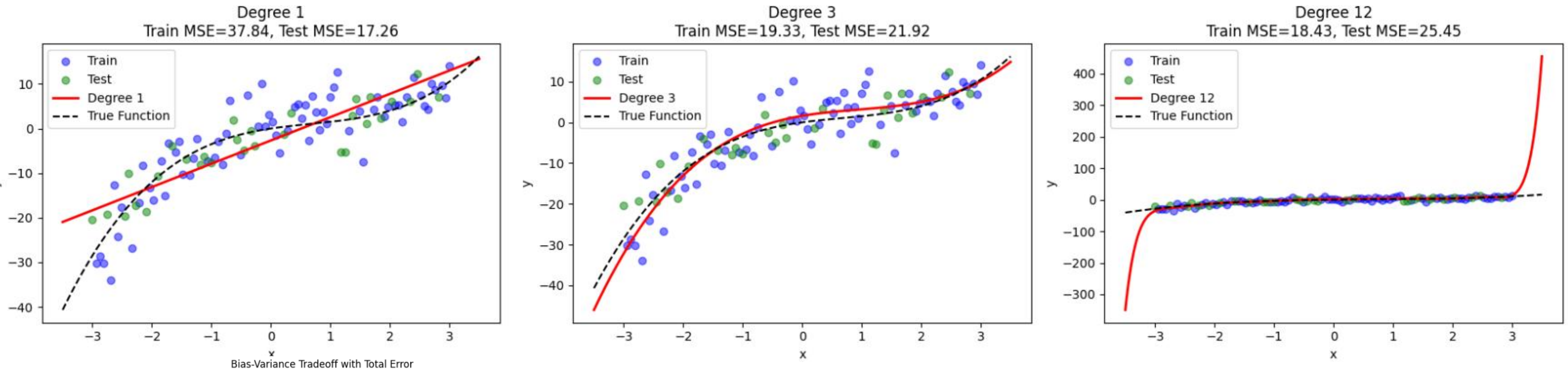


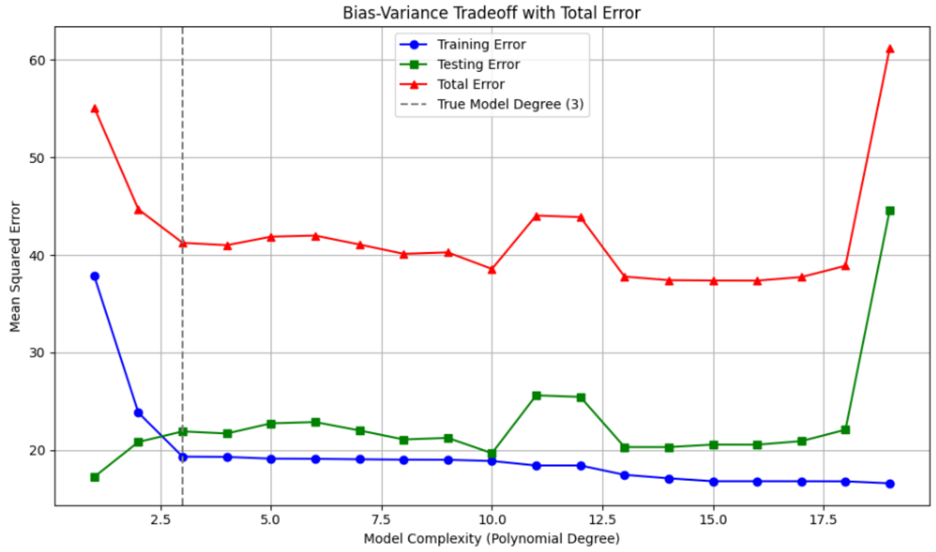
Bias-Variance Tradeoff



- Bias:
  - The error due to **overly simplistic assumptions** in the learning algorithm.
  - **High bias** means the **model is not complex enough** to capture the underlying patterns of the data.
- Variance:
  - The error due to the **model's sensitivity to small fluctuations** in the training data.
  - **High variance** means the **model fits the training data** very closely but **fails to generalize** well.

Overfitting vs Ideal Fit vs Underfitting





- $y = 0.5x^3 - x^2 + 2x + \text{noise}$
  - Degree 1: Underfit (linear), high bias.
  - Degree 3: Ideal fit, low bias and variance.
  - Degree 12: Overfit, capturing noise, high variance.
- The error curve clearly drops then rises again after degree 3.

Managing the tradeoff

1. Choose the Right Model Complexity

Simple models (e.g., linear regression) tend to have **high bias** and **low variance**, while **complex models** (e.g., deep neural networks) have **low bias** and **high variance**. Start with a simple model and increase complexity gradually, guided by validation performance.

2. Use More Training Data

More training data helps **reduce variance**, particularly for **complex models**, by enabling them to better capture the underlying data distribution.

3. Use Cross-Validation

Cross-validation (e.g., k-fold) provides a reliable estimate of model performance on unseen data. It helps in **identifying overfitting** (high variance) or **underfitting** (high bias).

4. Apply Regularization

Regularization techniques such as L2 (Ridge), L1 (Lasso), and dropout (for neural networks) penalize model complexity. They **reduce variance** by slightly increasing bias to prevent overfitting.

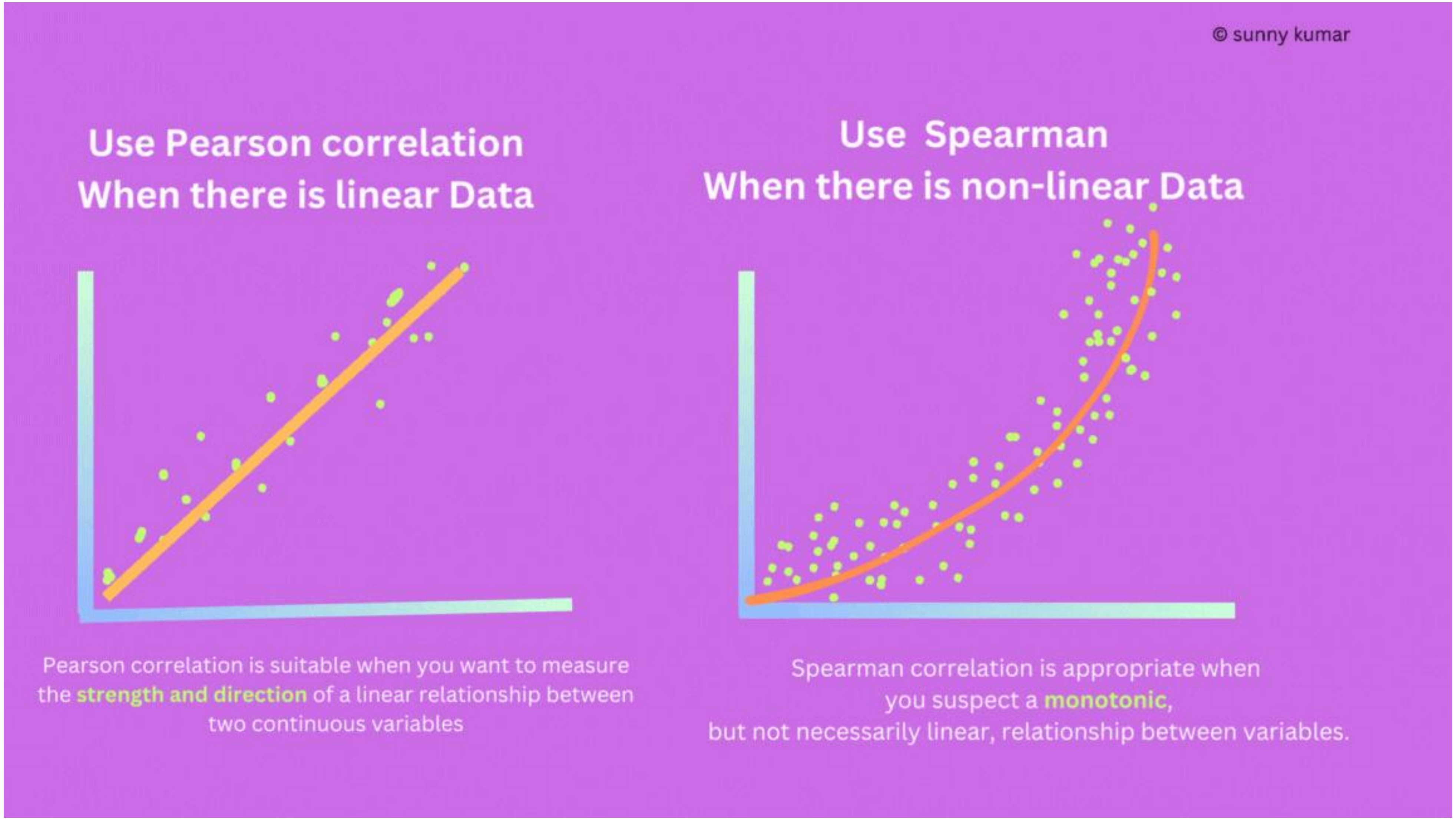
Managing the tradeoff

5. Use Ensemble Methods
Ensemble techniques like bagging (e.g., Random Forest) reduce variance, while boosting (e.g., XGBoost) can **reduce bias**. They **combine** multiple **models** to improve **generalization**.

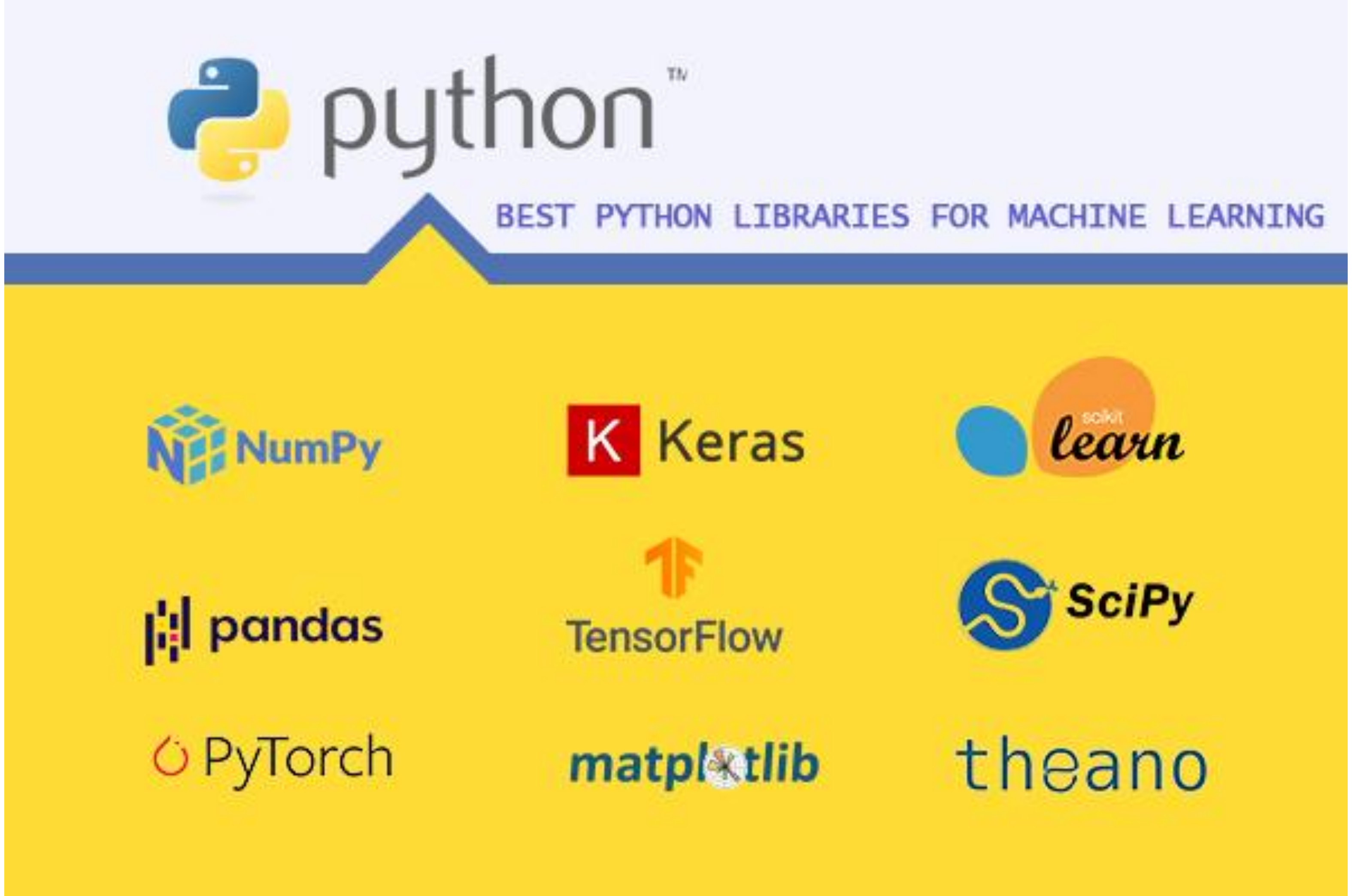
6. Feature Engineering and Selection
Creating informative features reduces bias, while **removing irrelevant or noisy features reduces variance**. Dimensionality reduction methods like PCA also help control variance.

7. Monitor Learning Curves
Plot training and validation errors against the number of training samples to diagnose bias and variance. High training and validation errors indicate high bias; low training error and high validation error indicate high variance.

Pearson vs Spearman



ML Libraries in Python



Python Libraries

- **NumPy** (Numerical Python)
  - Purpose: Foundation for numerical computing in Python.
  - Key Feature: ndarray, a powerful N-dimensional array object.
  - Speed: Vectorized operations using compiled C backend (fast linear algebra).
  - Uses: Matrix computations, random sampling, broadcasting, Fourier transforms.

- **Pandas**
  - Purpose: Data manipulation & analysis with labeled axes.
  - Core Structures: Series (1D) and DataFrame (2D).
  - Features:
    - Intuitive slicing, filtering, merging, and groupby operations.
    - Handles missing data, time series, categorical variables.
  - Best For: Tabular data processing and transformation pipelines.

Python Libraries

- **Scikit-learn**
  - Purpose: Standard ML toolkit for supervised & unsupervised learning.
  - Strengths:
    - Unified API for models: fit(), predict(), score()
    - Pipelines, cross-validation, hyperparameter tuning (GridSearchCV, RandomizedSearchCV)
  - Algorithms: SVM, Random Forest, KNN, PCA, KMeans, Logistic Regression, etc.
  - Integration: Works seamlessly with NumPy/pandas.

- **SciPy** (Scientific Python)
  - Purpose: Advanced scientific computing built on top of NumPy.
  - Submodules:
    - scipy.linalg: Linear algebra (beyond NumPy)
    - scipy.optimize: Optimization & curve fitting
    - scipy.spatial: Distance metrics, KD-trees
    - scipy.stats: Statistical functions, distributions, hypothesis testing
  - Use Case: Numerical routines required in engineering, physics, and ML.

Plotting and Typical Workflow

- **Matplotlib**
  - Purpose: 2D plotting library for visualizing data and models.
  - Main Interface: pyplot (similar to MATLAB).
  - Features:Line plots, scatter plots, bar charts, histograms, heatmaps.Highly customizable (styles, ticks, annotations).
  - Extensions: Integrates with seaborn for statistical plots.

`NumPy` → `pandas` (data wrangling) → `scikit-learn` (ML modeling) → `SciPy` (scientific routines) → `Matplotlib` (visualization)

Importing Libraries

```
import sys
print("Python version: {}".format(sys.version))

import pandas as pd
print("pandas version: {}".format(pd.__version__))

import matplotlib
print("matplotlib version: {}".format(matplotlib.__version__))

import numpy as np
print("NumPy version: {}".format(np.__version__))

import scipy as sp
print("SciPy version: {}".format(sp.__version__))

import IPython
print("IPython version: {}".format(IPython.__version__))

import sklearn
print("scikit-learn version: {}".format(sklearn.__version__))
```

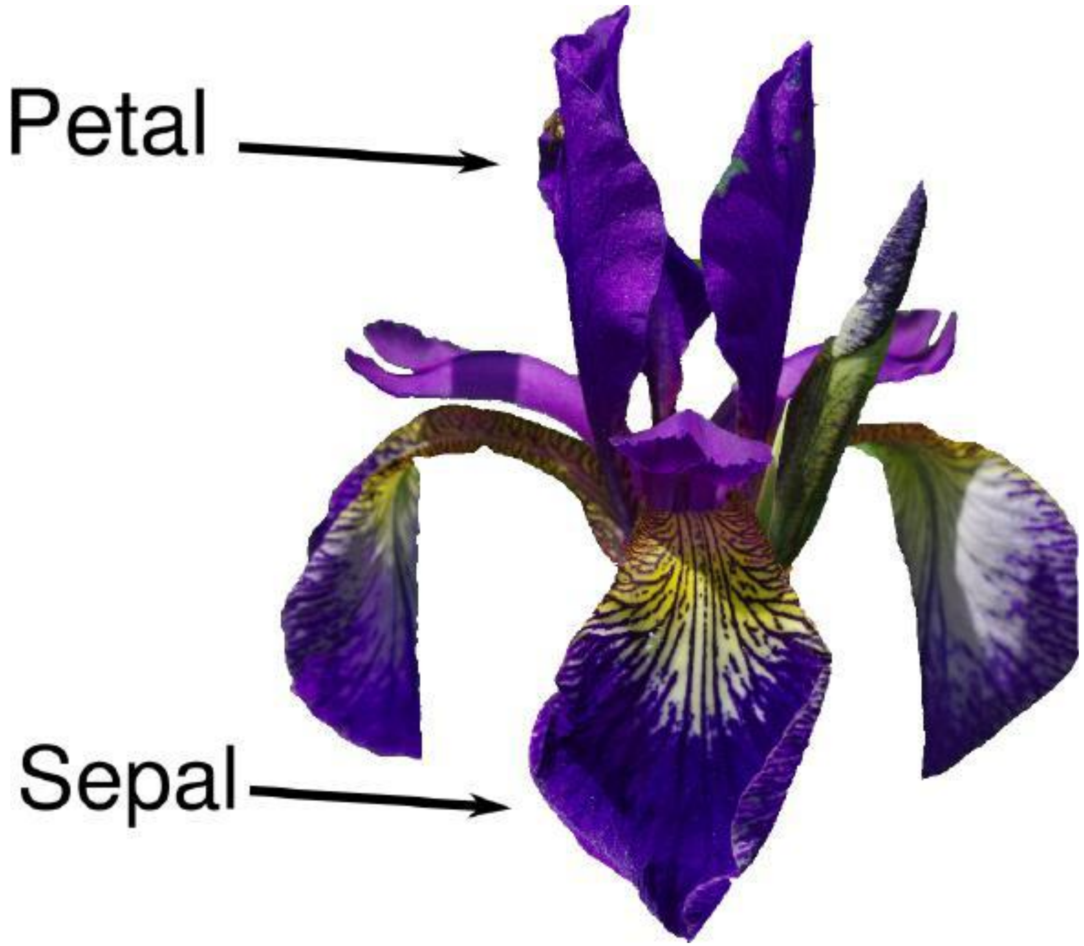
Supervised Learning Cases

Example	Task Type	Description
1️⃣ Spam Email Detection	Classification	Predict if an email is **spam or not** based on text features.
2️⃣ House Price Prediction	Regression	Predict the **price** of a house based on size, location, etc.
3️⃣ Medical Diagnosis	Classification	Classify whether a tumor is **benign or malignant** from imaging features.
4️⃣ Credit Scoring	Classification	Assess if a person is likely to **default on a loan** using credit history.
5️⃣ Stock Price Forecasting	Regression	Predict the **next day's stock price** using historical data.

Unsupervised Learning Cases

Example	Task Type	Description
1️⃣ Customer Segmentation	Clustering	Group customers by purchasing behavior (e.g., into **segments**).
2️⃣ Anomaly Detection	Outlier Detection	Detect **fraudulent transactions** without explicit labels.
3️⃣ Topic Modeling	Dimensionality Reduction	Discover **latent topics** in a collection of documents (e.g., LDA).
4️⃣ Image Compression	Feature Extraction	Reduce image size using **PCA** while preserving key information.
5️⃣ Gene Expression Analysis	Clustering	Cluster genes with similar expression patterns across samples.

A First Application: Classifying Iris Species



- **Classification**
  - Setosa
  - Versicolor
  - Virginica