# **📘 Scikit-Learn Roadmap (From Zero → Hero)**

1. **Introduction & Setup**
   * What is Scikit-Learn?
   * Installing & Importing
   * Basic dataset handling
2. **Core Concepts**
   * Features & Labels
   * Train-Test Split
   * Supervised vs Unsupervised Learning
3. **Supervised Learning**
   * Regression (LinearRegression, Ridge, etc.)
   * Classification (LogisticRegression, KNN, SVM, DecisionTree, RandomForest)
4. **Unsupervised Learning**
   * Clustering (KMeans, DBSCAN)
   * Dimensionality Reduction (PCA)
5. **Model Evaluation**
   * Metrics for Regression & Classification
   * Cross-validation
6. **Feature Engineering**
   * Scaling, Encoding, Pipelines
7. **Advanced Topics**
   * Hyperparameter Tuning (GridSearchCV, RandomizedSearchCV)
   * Ensemble Methods (GradientBoosting, XGBoost, VotingClassifier)
   * Working with text data (CountVectorizer, TF-IDF)
   * Custom Transformers & Pipelines

## **🔹 Part 1: Introduction & Setup**

👉 **What is Scikit-Learn?** Scikit-learn is a Python library that provides:

* Pre-built ML algorithms (regression, classification, clustering, etc.)
* Tools for model evaluation & preprocessing
* Datasets for practice

**✅ Step 1: Install and Import**

### **✅ Step 2: Load a Built-in Dataset**

Scikit-learn comes with sample datasets like iris (flowers), digits (images), etc.

### **✅ Step 3: Train-Test Split**

Before training, we split into training & testing sets.

In scikit-learn's train\_test\_split function, these two parameters control how your data is divided:

## **test\_size**

This determines **how much data goes to the test set**. You can specify it as:

* **Float (0.0 to 1.0)**: Represents the proportion of the dataset
  + test\_size=0.2 means 20% for testing, 80% for training
  + test\_size=0.3 means 30% for testing, 70% for training
* **Integer**: The absolute number of test samples
  + test\_size=100 means exactly 100 samples for testing

**Common practice**: Most people use test\_size=0.2 or 0.3 (20-30% for testing)

## **random\_state**

This controls the **randomness of the split**. It's like a seed for the random number generator.

* **Purpose**: Makes your results **reproducible**
* **How it works**: Using the same random\_state value will always produce the same split
  + random\_state=42 (or any integer) ensures you get identical train/test splits every time you run the code
  + random\_state=None (default) gives you a different random split each time

**Why use it?** When you're experimenting or sharing code, you want others to get the exact same results. Setting random\_state=42 (or any number) ensures consistency.

Without random\_state, you'd get slightly different model performance each time you run your code, making debugging and comparison difficult!

The **R² score** (R-squared, also called the coefficient of determination) measures **how well your model's predictions fit the actual data**.

## **What does it mean?**

**R² tells you what percentage of the variance in your target variable is explained by your model.**

* **R² = 1.0** (perfect): Your model perfectly predicts every data point
* **R² = 0.8**: Your model explains 80% of the variance (pretty good!)
* **R² = 0.5**: Your model explains 50% of the variance (moderate)
* **R² = 0.0**: Your model is no better than just predicting the average value
* **R² < 0** (negative): Your model is worse than just predicting the average (very bad!)

## **Simple intuition**

Think of it as a **percentage score for your model's accuracy** in regression problems:

* Higher is better
* Closer to 1 means better predictions
* Closer to 0 means poor predictions

R² score and **Mean Squared Error (MSE)** are **related but different** metrics!

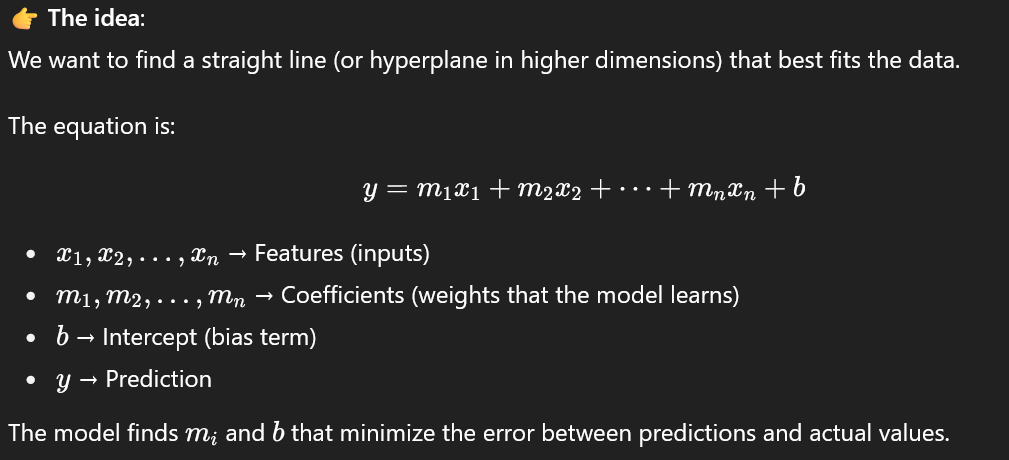
## **Key Differences**

### **Mean Squared Error (MSE)**

* **Absolute error measure** - tells you the average of squared differences between predictions and actual values
* **Lower is better** (0 is perfect)
* **Has units** (squared units of your target variable)
* **Example**: MSE = 25 means on average, your predictions are off by √25 = 5 units

### **R² Score**

* **Relative measure** - tells you how much variance your model explains (as a proportion)
* **Higher is better** (1 is perfect)
* **No units** - it's always between -∞ and 1
* **Example**: R² = 0.85 means your model explains 85% of the variance



If we want to predict **house price** just from **area (sq ft)**:

Price=m⋅Area+b\text{Price} = m \cdot \text{Area} + bPrice=m⋅Area+b

The algorithm will try to draw the **best straight line** through the data points.

* If m is large → steeper line
* If b is large → line shifts upward

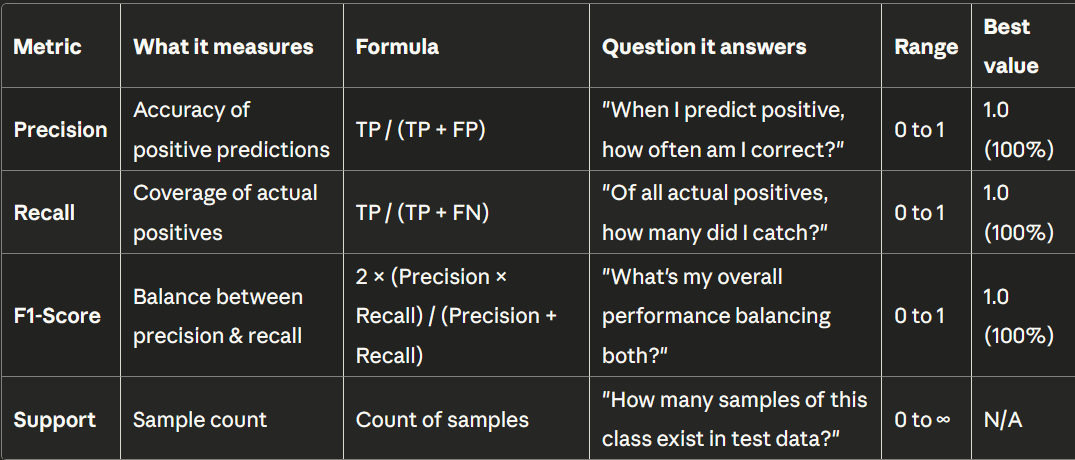
# **🔹 Part 3: Logistic Regression (Classification)**

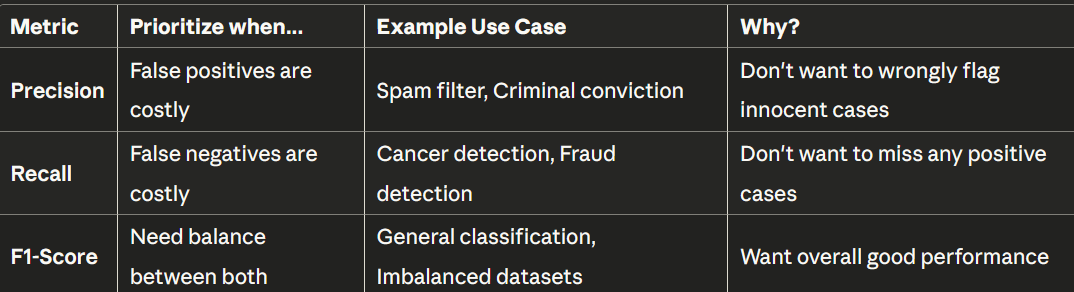
👉 **What is Logistic Regression?** Despite its name, it’s not for regression — it’s used for **binary or multi-class classification**.

* Example: Spam (1) vs Not Spam (0)
* Example: Iris flower species (Setosa, Versicolor, Virginica)

The core idea:

* Uses a **sigmoid function** to squash predictions between 0 and 1 → interpreted as probability.
* If probability > 0.5 → Class 1, else → Class 0.





# **🔹 Part 4: K-Nearest Neighbors (KNN)**

👉 **What is KNN?**

* A **non-parametric** algorithm (doesn’t assume any function).
* It predicts the class of a sample based on the **majority vote of its K nearest neighbors** in feature space.
* Distance is usually measured using **Euclidean distance**.

Example:  
 If we want to classify a fruit as apple 🍎 or orange 🍊, KNN looks at its closest fruits in the dataset and assigns the majority label.

📊 **Output you’ll see**:

* Accuracy changes as **k** increases.
* Small k → more sensitive to noise.
* Large k → smoother, but may misclassify minority classes.

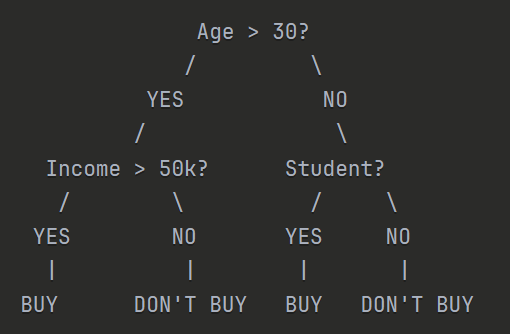
# **🔹 Part 5: Decision Trees & Random Forests 🌳🌲**

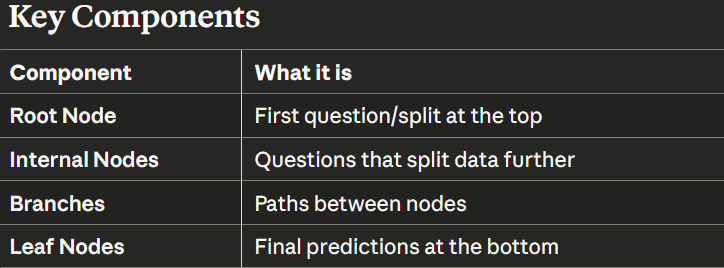
## **✅ Decision Tree Classifier**

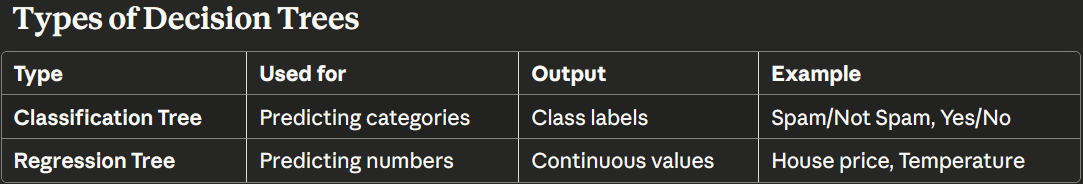
👉 **What is it?**

* A tree-like model where data is split based on feature conditions.
* Each split tries to maximize "purity" (all samples in a node belong to the same class).
* Easy to interpret, but can **overfit** if not controlled.

A **decision tree** is a machine learning algorithm that makes predictions by asking a series of **yes/no questions** about your data, like playing "20 Questions"!



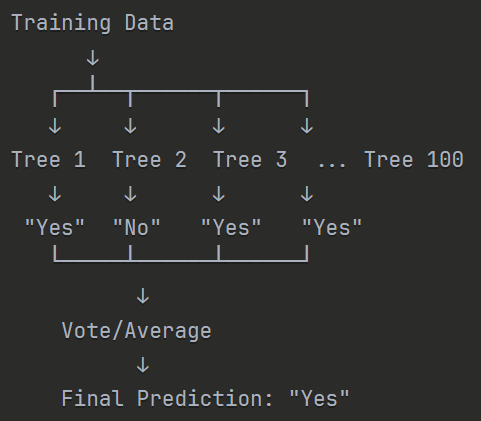




## **✅ Random Forest Classifier**

👉 **What is it?**

* An **ensemble** of decision trees.
* Each tree is trained on a random subset of data & features.
* Predictions are made by **majority vote** across trees.
* Much more robust than a single decision tree.



Many Decision trees make Random Forest

# **🔹 Part 6: Unsupervised Learning**

Unlike supervised learning, here we don’t have labels (y).  
 The algorithm just looks at **patterns/structure** in the data.

Main types:

1. **Clustering** → Group similar points (e.g., customers with similar buying habits).
2. **Dimensionality Reduction** → Compress features (e.g., PCA for visualization).

## **✅ Step 1: K-Means Clustering**

👉 **Idea**:

* Pick k clusters (groups).
* Randomly assign cluster centers.
* Iteratively update until convergence.

## **✅ Step 2: PCA (Dimensionality Reduction)**

👉 Problem: Data often has **many features** → hard to visualize.  
 👉 PCA = Principal Component Analysis → projects high-dimensional data into fewer dimensions while keeping **max variance**.

Variance measures **how spread out your data is**.

* High variance → data points are far from the mean
* Low variance → data points are close to the mean

**============================================================**

**FEATURE ENGINEERING PIPELINE**

**============================================================**

**+----------------------------------------------------------+**

**| STEP 1: DATA COLLECTION & UNDERSTANDING |**

**+----------------------------------------------------------+**

**| - Gather raw data from sources |**

**| - Exploratory Data Analysis (EDA) |**

**| - Understand data types & distributions |**

**+----------------------------------------------------------+**

**|**

**v**

**+----------------------------------------------------------+**

**| STEP 2: DATA CLEANING |**

**+----------------------------------------------------------+**

**| - Handle missing values |**

**| - Detect and treat outliers |**

**| - Remove duplicates |**

**+----------------------------------------------------------+**

**|**

**v**

**+----------------------------------------------------------+**

**| STEP 3: FEATURE TRANSFORMATION |**

**+----------------------------------------------------------+**

**| - Scaling/Normalization |**

**| - Log/Power transformations |**

**| - Binning/Discretization |**

**+----------------------------------------------------------+**

**|**

**v**

**+----------------------------------------------------------+**

**| STEP 4: FEATURE ENCODING |**

**+----------------------------------------------------------+**

**| - One-Hot Encoding |**

**| - Label Encoding |**

**| - Target/Frequency Encoding |**

**+----------------------------------------------------------+**

**|**

**v**

**+----------------------------------------------------------+**

**| STEP 5: FEATURE CREATION |**

**+----------------------------------------------------------+**

**| - Interaction features |**

**| - Polynomial features |**

**| - Date/Time & Text features |**

**+----------------------------------------------------------+**

**|**

**v**

**+----------------------------------------------------------+**

**| STEP 6: FEATURE SELECTION |**

**+----------------------------------------------------------+**

**| - Correlation analysis |**

**| - Feature importance |**

**| - Recursive Feature Elimination |**

**+----------------------------------------------------------+**

**|**

**v**

**+----------------------------------------------------------+**

**| STEP 7: VALIDATION & ITERATION |**

**+----------------------------------------------------------+**

**| - Cross-validation |**

**| - Performance metrics |**

**| - Iterate and refine |**

**+----------------------------------------------------------+**

**|**

**v**

**+-----------------+**

**| FINAL FEATURES |**

**+-----------------+**