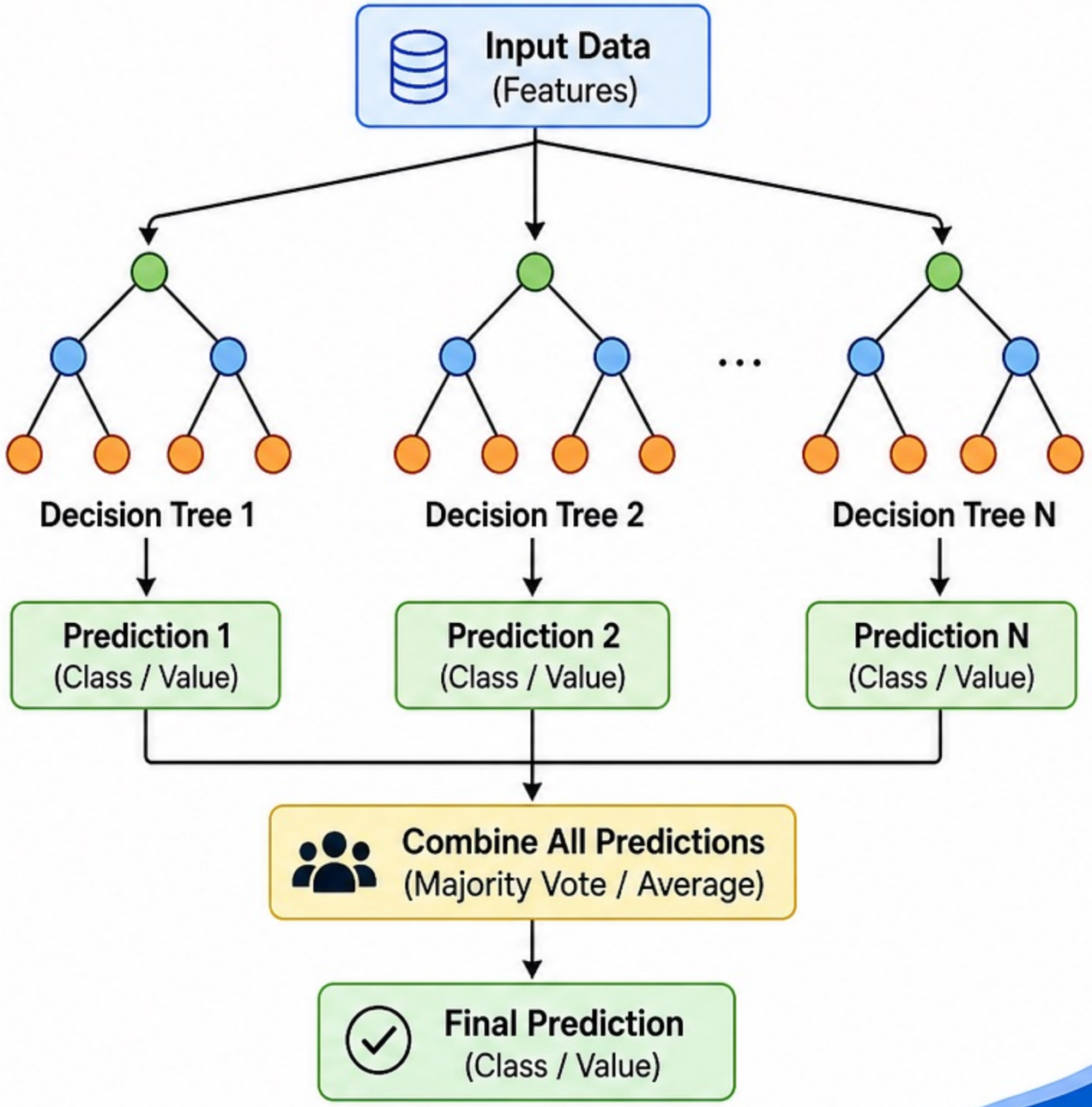
What is Random Forest in Machine Learning?

Random Forest is a powerful machine learning algorithm used for both **classification** and **regression**.

It belongs to **ensemble learning** and is based on **Decision Trees**.

The simple idea:
Use many trees instead of one.

Combine results from all trees for **better predictions**.



www.codeswithpankaj.com
Pankaj Chouhan

What is a Decision Tree in Machine Learning?

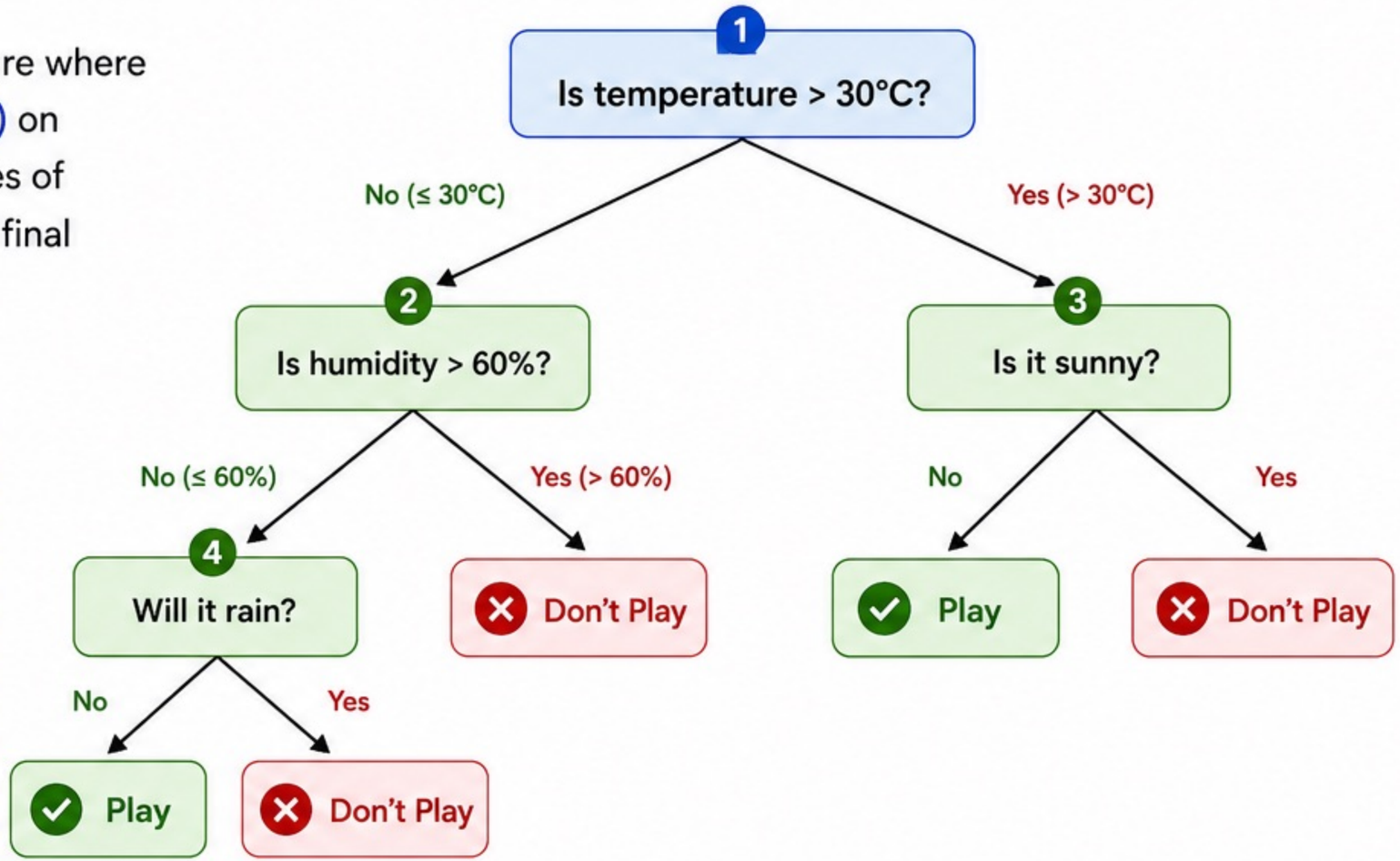
What is a Decision Tree?

A Decision Tree is a flowchart-like structure where internal nodes represent **questions (tests)** on features, **branches** represent the outcomes of those tests, and **leaf nodes** represent the final decisions or predictions.

Key Points

- It is easy to understand and interpret.
- It can be used for both **Classification** and **Regression** problems.
- It splits data based on feature values to make decisions.
- It forms the building block of **Random Forest**.

Decision Tree Structure (Example)



Root Node (Start)

Decision Node (Question)

Leaf Node (Play)

Leaf Node (Don't Play)

The tree starts at the **root node** and moves down based on answers until it reaches a **leaf node (final decision)**.

Random Forest - Basic Intuition

1 Basic Intuition

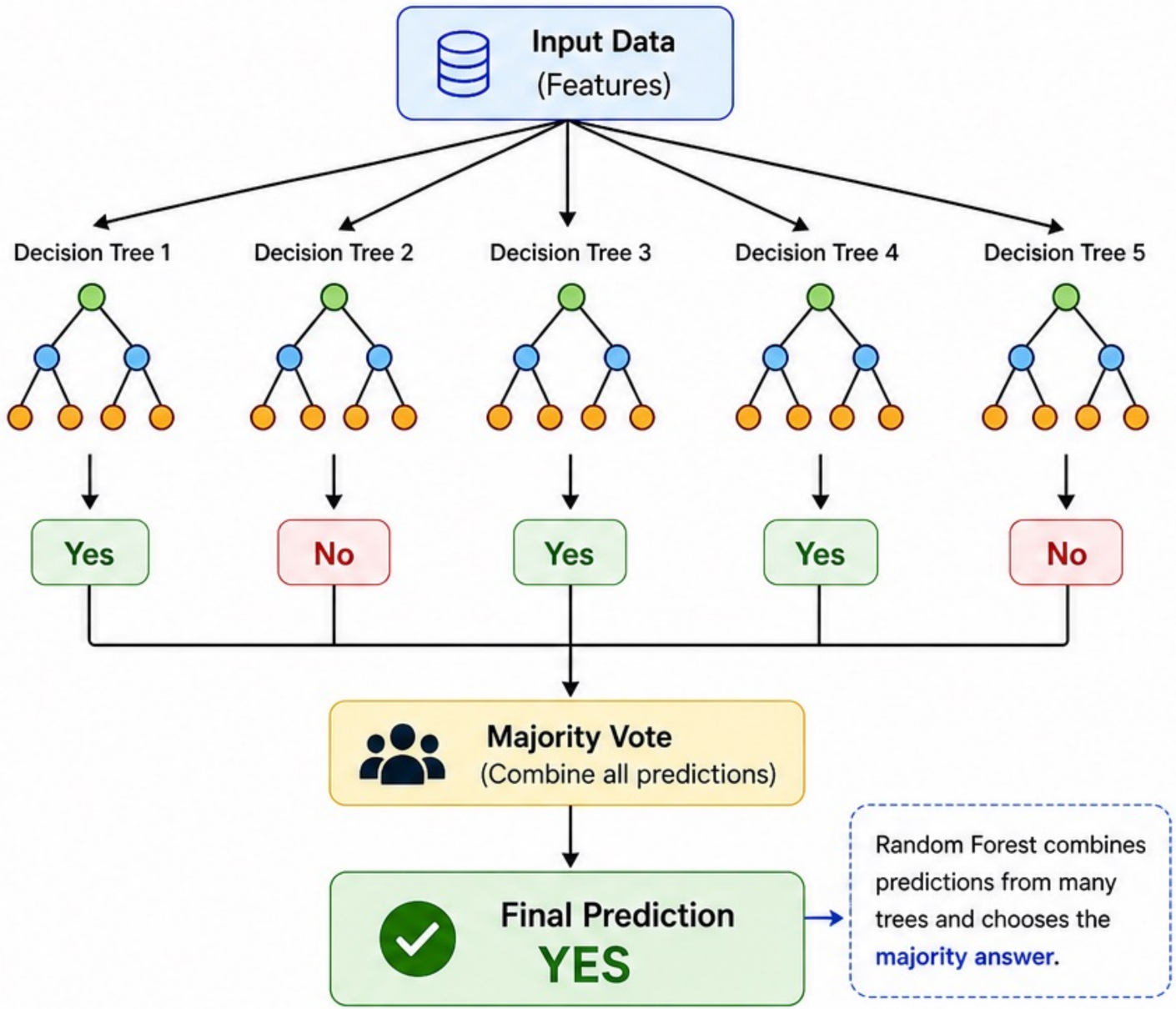
Imagine you ask 5 people:

"Will it rain tomorrow?"

Person 1	→	Yes
Person 2	→	No
Person 3	→	Yes
Person 4	→	Yes
Person 5	→	No

👉 Final decision = Majority Voting → **YES**

2 How Random Forest Works



www.codeswithpankaj.com
Pankaj Chouhan

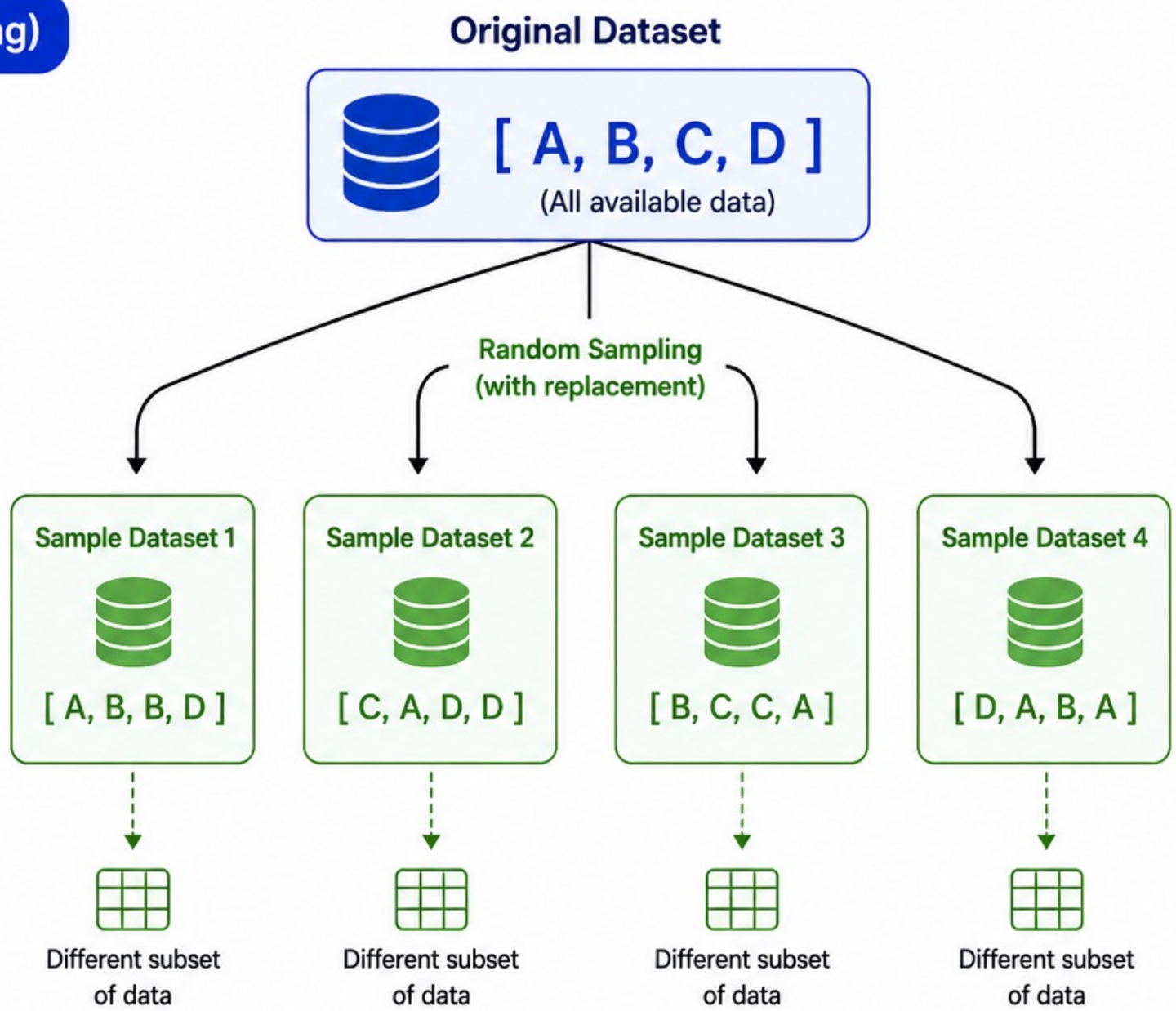
How Random Forest Works (Step-by-Step)

1 Step 1: Bootstrap Sampling (Bagging)

- Create **Multiple Datasets** from original data
- Random selection **WITH replacement**
- Some data points may **repeat**
- Each dataset is slightly **different**
- This process is called **Bagging (Bootstrap Aggregation)**

Example

Original Data:	[A, B, C, D]
Sample 1:	[A, B, B, D]
Sample 2:	[C, A, D, D]
Sample 3:	[B, C, C, A]



Random Forest in Machine Learning

Random Forest is an ensemble learning algorithm that builds multiple decision trees and combines their predictions to get **better accuracy** and **avoid overfitting**.

1 Bootstrap Sampling

From the original dataset, create multiple datasets by **random sampling with replacement**.

Original Dataset

ID	Weather	Temp	Play
1	Sunny	Hot	No
2	Sunny	Mild	Yes
3	Rainy	Cool	Yes
4	Sunny	Hot	No
5	Rainy	Mild	Yes
6	Cloudy	Cool	Yes
7	Sunny	Mild	Yes
8	Rainy	Hot	No
...			

↓

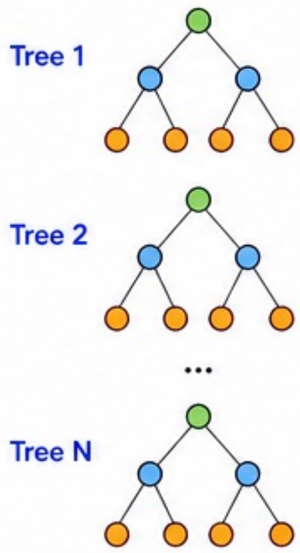
Bootstrap Samples

Sample 1	Sample 2	Sample N
2, 5, 1, 2, 8, 6, 2, 3 ...	4, 7, 4, 1, 6, 3, 7, 2 ...	3, 6, 8, 1, 5, 2, 6, 4 ...

This process is called **Bagging (Bootstrap Aggregation)**.

2 Build Decision Trees (in Parallel)

Train a decision tree on each sample dataset. Each tree will learn different patterns.



Trees are built independently and in parallel.

3 Random Feature Selection

At each split in the tree, a random subset of features is selected instead of using all features.

All Features

Weather | Temp | Humidity | Wind

↓

Randomly Selected Subset (Example)

Weather | Humidity

Why?

It makes trees more diverse and reduces overfitting.

4 Make Predictions

Give a new input to all trees. Each tree makes its own prediction.

New Input

(e.g., Weather = Rainy, Temp = Cool, Humidity = High)

Tree 1 → Yes

Tree 2 → No

...

Tree N → Yes

5 Combine Predictions (Voting / Averaging)

Combine the predictions from all trees to get the final result.

Classification (Voting)

Yes, No, Yes, Yes, No, Yes

↓

Majority Vote

YES

(It will rain)

Regression (Averaging)

Predictions: 2.1, 2.5, 3.0, 1.8, 2.6

↓

Average

2.4

Key Takeaway: Random Forest combines many trees trained on different data and features and uses **voting (classification)** or **averaging (regression)** to make more accurate and reliable predictions.

Benefits of Random Forest

- ✔ High Accuracy
- ✔ Reduces Overfitting
- ✔ Works for both Classification & Regression
- ✔ Handles large datasets well

www.codeswithpankaj.com
Pankaj Chouhan

STEP 1 Create Multiple Datasets (Bootstrap Sampling)

From the original dataset, multiple smaller datasets are created.

Data is selected randomly with replacement.

This process is called Bagging (Bootstrap Aggregation).

What is Bagging?

Bagging means creating many different training sets by random sampling with replacement from the original data.

- Some data points may repeat
- Some data points may not appear in a sample

Example

Original Dataset

ID	Weather	Temperature	Play
1	Sunny	Hot	No
2	Sunny	Mild	Yes
3	Rainy	Cool	Yes
4	Cloudy	Cool	Yes
5	Rainy	Mild	No
6	Sunny	Hot	Yes

Bootstrap Sampling
(Random sampling with replacement)

Sample Dataset 1

ID	Weather	Temp	Play
2	Sunny	Mild	Yes
5	Rainy	Mild	No
2	Sunny	Mild	Yes
6	Sunny	Hot	Yes
3	Rainy	Cool	Yes
2	Sunny	Mild	Yes

Sample Dataset 2

ID	Weather	Temp	Play
5	Rainy	Mild	No
1	Sunny	Hot	No
5	Rainy	Mild	No
3	Rainy	Cool	Yes
6	Sunny	Hot	Yes
1	Sunny	Hot	No

Sample Dataset 3

ID	Weather	Temp	Play
4	Cloudy	Cool	Yes
2	Sunny	Mild	Yes
4	Cloudy	Cool	Yes
3	Rainy	Cool	Yes
6	Sunny	Hot	Yes
2	Sunny	Mild	Yes

...

Sample Dataset N

ID	Weather	Temp	Play
1	Sunny	Hot	No
4	Cloudy	Cool	Yes
5	Rainy	Mild	No
1	Sunny	Hot	No
6	Sunny	Hot	Yes
3	Rainy	Cool	Yes

Key Idea

We create multiple datasets so that each model (decision tree) sees a slightly different view of the data.

Outcome: We now have multiple datasets (Sample Dataset 1, 2, 3, ... N) created using bootstrap sampling.
Next Step: We will use each of these datasets to build a separate decision tree.

STEP 2 Build Multiple Decision Trees

Each dataset is used to train a **separate decision tree**.

Trees are built **independently** and in **parallel**.

Example From Step 1, we have multiple datasets

Sample Dataset 1

ID	Weather	Temp	Play
2	Sunny	Mild	Yes
5	Rainy	Mild	No
2	Sunny	Mild	Yes
6	Sunny	Hot	Yes
3	Rainy	Cool	Yes

Sample Dataset 2

ID	Weather	Temp	Play
5	Rainy	Mild	No
1	Sunny	Hot	No
5	Rainy	Mild	No
3	Rainy	Cool	Yes
6	Sunny	Hot	Yes

Sample Dataset 3

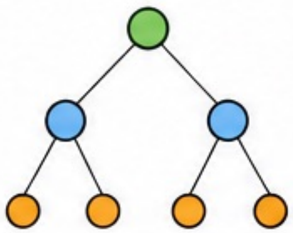
ID	Weather	Temp	Play
4	Cloudy	Cool	Yes
2	Sunny	Mild	Yes
4	Cloudy	Cool	Yes
3	Rainy	Cool	Yes
6	Sunny	Hot	Yes

...

Sample Dataset N

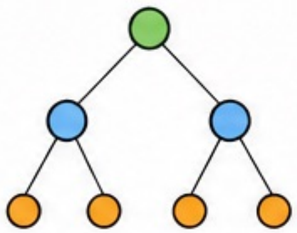
ID	Weather	Temp	Play
1	Sunny	Hot	No
4	Cloudy	Cool	Yes
5	Rainy	Mild	No
1	Sunny	Hot	No
6	Sunny	Hot	Yes

Decision Tree 1



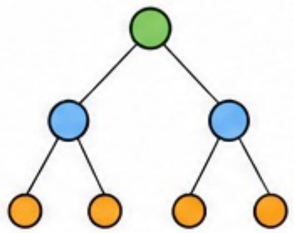
Trained on Sample Dataset 1

Decision Tree 2



Trained on Sample Dataset 2

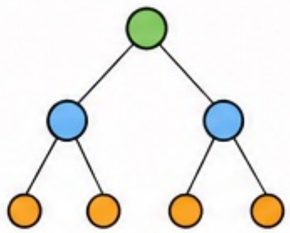
Decision Tree 3



Trained on Sample Dataset 3

...

Decision Tree N



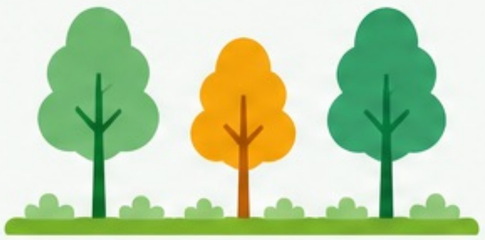
Trained on Sample Dataset N

Each decision tree is built independently using its own dataset.
All trees are built in parallel.

Key Idea

Each tree learns patterns from its own dataset.

Because datasets are different, trees will also be different.



STEP 3 Random Feature Selection

At each split in a tree, only a random subset of features is considered.

This makes trees different from each other.

This is why it's called Random Forest.

Key Idea

Trees see different features at each split, so they learn different patterns from the data.

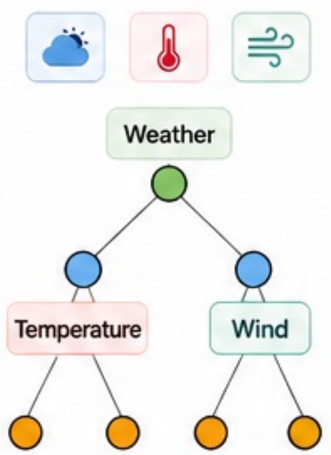
Example Assume our dataset has 5 features:

Weather | Temperature | Humidity | Wind | Play (Target)

At each split, only a random subset of these features is considered.

Decision Tree 1

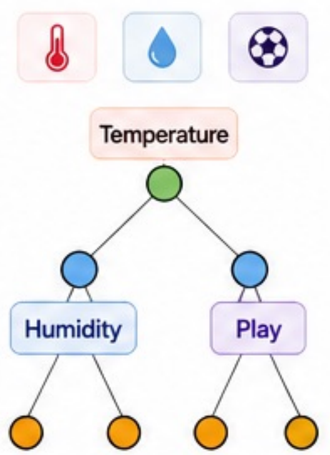
Randomly chosen features at each split



Uses a random subset of features at each split

Decision Tree 2

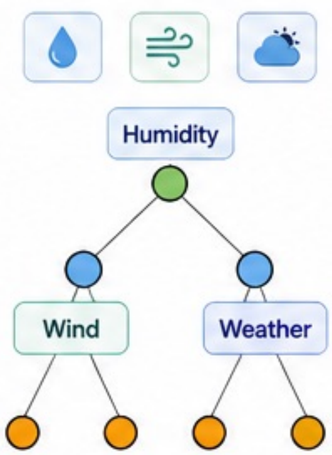
Randomly chosen features at each split



Uses a different random subset of features

Decision Tree 3

Randomly chosen features at each split

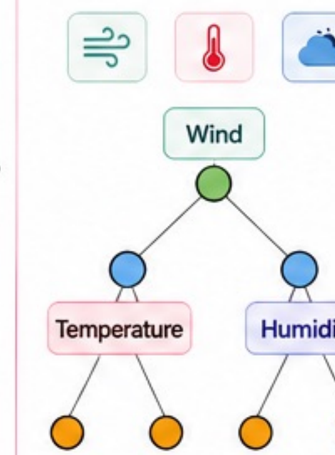


Uses a different random subset of features

...

Decision Tree N

Randomly chosen features at each split



Uses a different random subset of features

Because each tree uses different features at splits, the trees become diverse.
This diversity reduces overfitting and improves generalization!

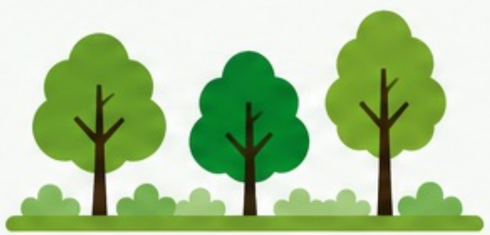
STEP 4 Combine Predictions (Voting / Averaging)

Each decision tree makes its own prediction.

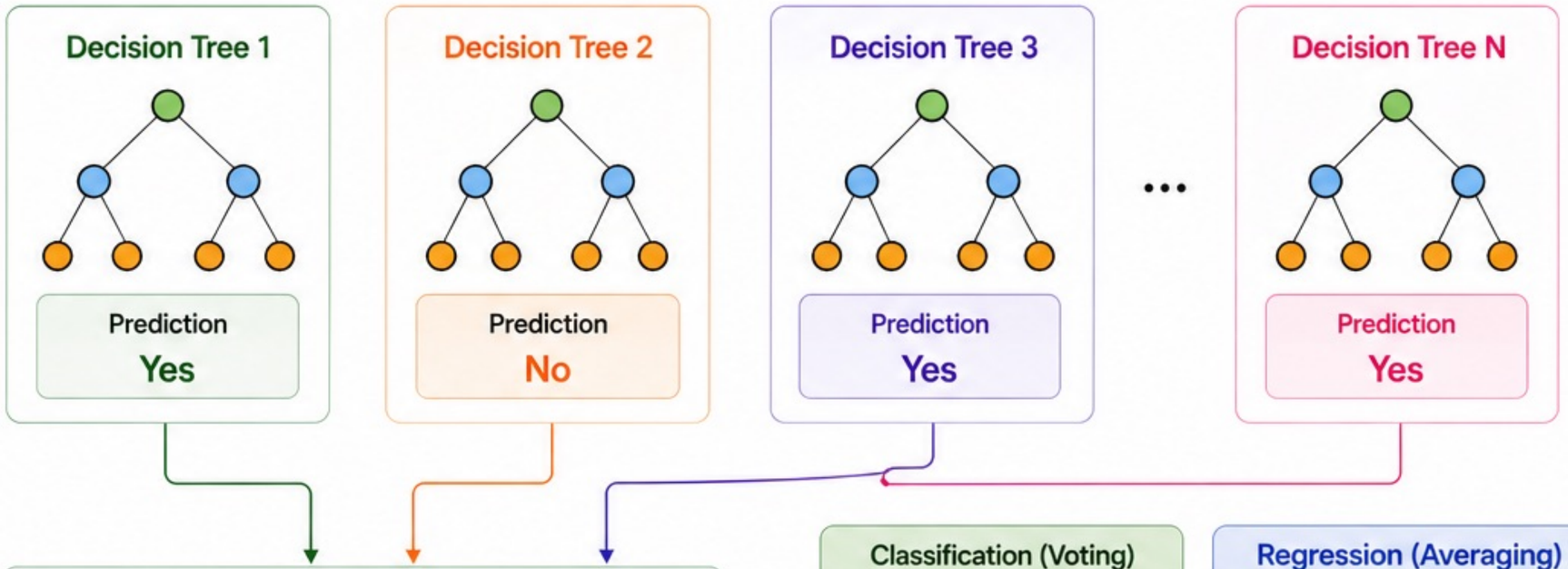
We combine all the predictions to get the final output.

Key Idea

Instead of relying on one tree, we take help from all trees. This reduces errors and makes predictions more accurate and stable.



Example Let's predict: Will it play tomorrow? (Yes / No)



Combine All Predictions

Predictions from all trees:

Yes | No | Yes | Yes | ... | Yes

Classification (Voting)

Majority Voting

Yes = 4

No = 1

Final Prediction

Yes

Regression (Averaging)

Predictions from trees:

2.1, 3.4, 2,8, 3.0, 2.6

Average

2.78

(Final Prediction)

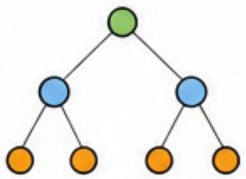
In Classification → We use Majority Voting.
In Regression → We use Average of predictions.

This way, Random Forest makes a final prediction by combining the power of all decision trees.

STEP 5 Final Prediction (The Power of Many Trees)

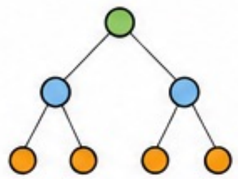
Random Forest makes the final prediction by **combining the results of all decision trees.**
This leads to more accurate, stable, and reliable predictions.

Decision Tree 1



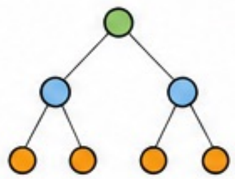
Prediction
Yes

Decision Tree 2



Prediction
No

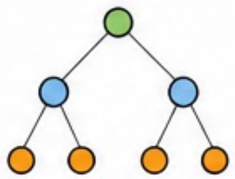
Decision Tree 3



Prediction
Yes

...

Decision Tree N



Prediction
Yes

Combine All Predictions

Predictions from all trees:

Yes | No | Yes | Yes | No | Yes | Yes | ...

Classification (Voting)

Count the votes:
Yes = 5 | No = 2

Final Prediction:
Yes

Regression (Averaging)

Predictions from trees:
2.1, 3.4, 2.8, 3.0, 2.6, 3.1, 2.9, ...

Average (Final Prediction):
2.84

Real World Example

Problem: Predict whether it will rain tomorrow (Yes / No)
Using Random Forest:

- Many trees look at the data from different angles.
- Each tree gives its own answer.
- We combine all answers to get the final, best prediction.

Key Takeaway

By combining multiple trees, Random Forest reduces overfitting, handles complex data well, and gives more accurate and robust predictions.

Why Random Forest Works So Well?

- Reduces Overfitting (High Bias)
- Handles High Dimensional Data
- More Accurate and Stable
- Works Well for Both Classification & Regression
- Can Handle Large Datasets Efficiently

Why Use Random Forest?

Why Use Random Forest?

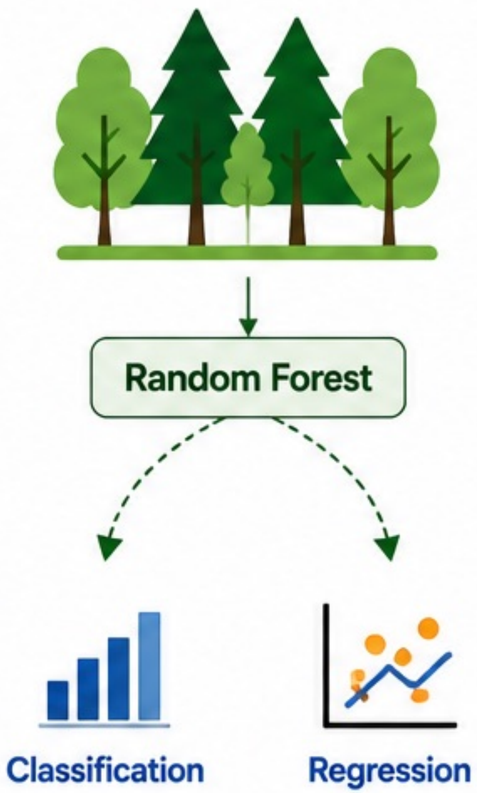
Handles large datasets
Can efficiently work with thousands to millions of records.

Works well with missing values
Can handle missing data without much preprocessing.

Reduces overfitting (better than single tree)
By averaging multiple trees, it generalizes better to unseen data.

High accuracy
Combining many trees gives more stable and accurate results.

Works for both classification & regression
Suitable for a wide range of machine learning problems.



Disadvantages

Slower than a single decision tree
Training multiple trees takes more time.

Harder to interpret (less explainable)
Many trees make it difficult to understand the decision process.

Uses more memory
Storing multiple trees requires more memory compared to a single tree.

Random Forest Step by Step Example Codes

Understand Random Forest with Diagram + Python Codes (Scikit-learn)

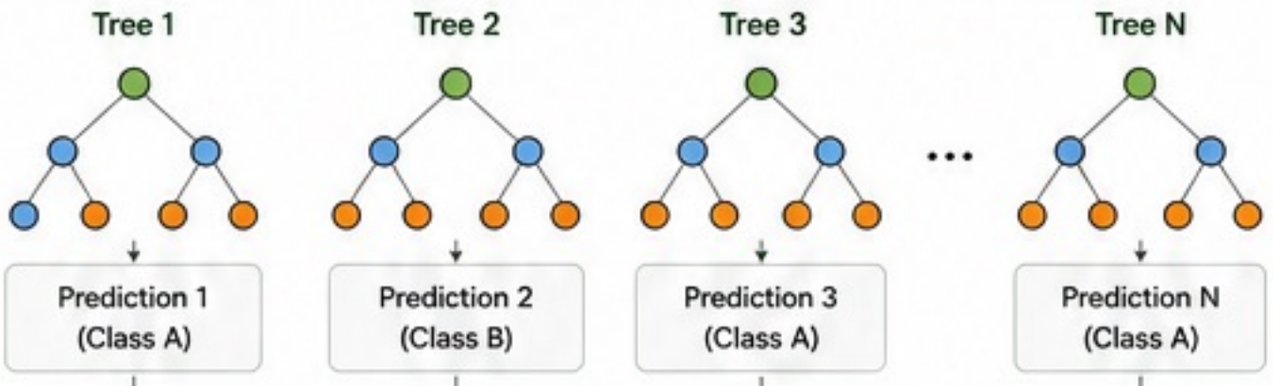
HOW RANDOM FOREST WORKS

1. Original Dataset

Rows = Samples
Columns = Features

2. Random Sampling + Feature Selection

Random Forest creates multiple trees



3. Majority Voting (Classification) / Averaging (Regression)

Final Prediction
(Class A)

- Each tree sees:
  - Random subset of rows
  - Random subset of features
- Benefits:
  - Reduces overfitting
  - Increases accuracy
  - Works well on large datasets
  - Handles missing values

1 Import Libraries

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import
     train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,
     classification_report, confusion_matrix
```

2 Load the Dataset

```
# Load Iris dataset
iris = load_iris()

X = iris.data
y = iris.target

print("Features:", iris.feature_names)
print("Target Classes:", iris.target_names)
```

Target Classes

- 0 → setosa
- 1 → versicolor
- 2 → virginica

3 Split the Data

```
# Split into training and testing sets
X_train, X_test, y_train, y_test =
  train_test_split(
      X, y, test_size=0.2,
       random_state=42, stratify=y)

print("Training samples:", X_train.shape[0])
print("Testing samples:", X_test.shape[0])
```

4 Create Random Forest Model

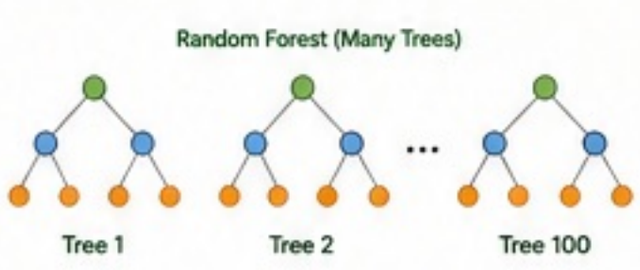
```
# Create Random Forest Classifier
rf = RandomForestClassifier(
  n_estimators=100,    # Number of trees
  criterion='gini',    # Split criterion
  random_state=42,     # For reproducibility
  n_jobs=-1            # Use all CPUs
)
print("Random Forest model created!")
```

Important Parameters

- n_estimators : Number of trees
- criterion : gini or entropy
- random_state : For reproducibility
- n_jobs : Use all CPUs (-1 means all cores)

5 Train the Model

```
# Train Random Forest
rf.fit(X_train, y_train)
print("Model training completed!")
```



6 Make Predictions

```
# Predict on test data
y_pred = rf.predict(X_test)

print("Predictions:", y_pred[:10])
print("Actual:     ", y_test[:10])
```

Example Output (First 10)									
Predicted	0	1	2	1	2	0	2	2	1
Actual	0	1	2	1	2	0	2	2	0

7 Evaluate the Model

```
# Model Evaluation
acc = accuracy_score(y_test, y_pred)
print(f"Accuracy: {acc:.2f}")

print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:\n")
print(confusion_matrix(y_test, y_pred))
```

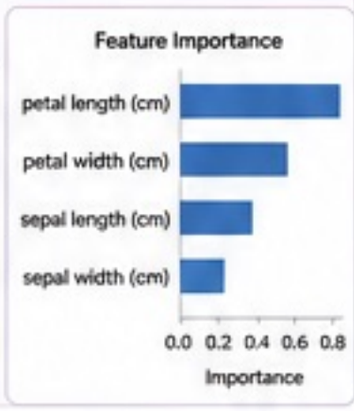
Accuracy
0.97

Confusion Matrix

50	0	0
0	47	3
0	2	48

8 Feature Importance

```
# Feature Importance
importances = rf.feature_importances_
feature_names = iris.feature_names
feat_imp = pd.DataFrame({
   'Feature': feature_names,
   'Importance': importances
}).sort_values(by='Importance',
                ascending=False)
print(feat_imp)
```



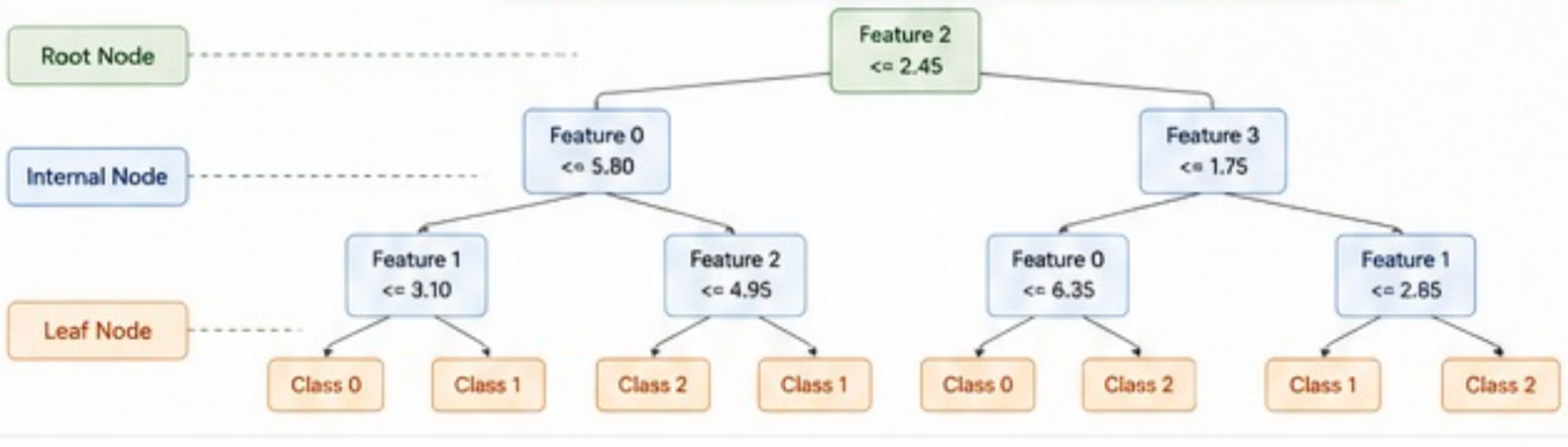
9 Save the Model (Optional)

```
import joblib

# Save the model
joblib.dump(rf, 'random_forest_model.pkl')
print("Model saved successfully!")

# Load the model
loaded_model = joblib.load('random_forest_model.pkl')
print("Model loaded successfully!")
```

RANDOM FOREST TREE STRUCTURE (How a single tree branches)



- **Root Node** — First split on the most important feature.
- **Internal Node** — Further splits based on feature conditions
- **Leaf Node** — Final prediction (Class Label)

COMPLETE WORKFLOW

1 Import Libraries → 2 Load Dataset → 3 Split Data → 4 Create Model → 5 Train Model → 6 Make Predictions → 7 Evaluate Model → 8 Feature Importance → 9 Save / Load Model

SUMMARY

- Random Forest builds many decision trees.
- Each tree sees random rows and features.
- Predictions are combined (voting/averaging).
- It gives high accuracy and reduces overfitting.
- Works great for classification & regression.