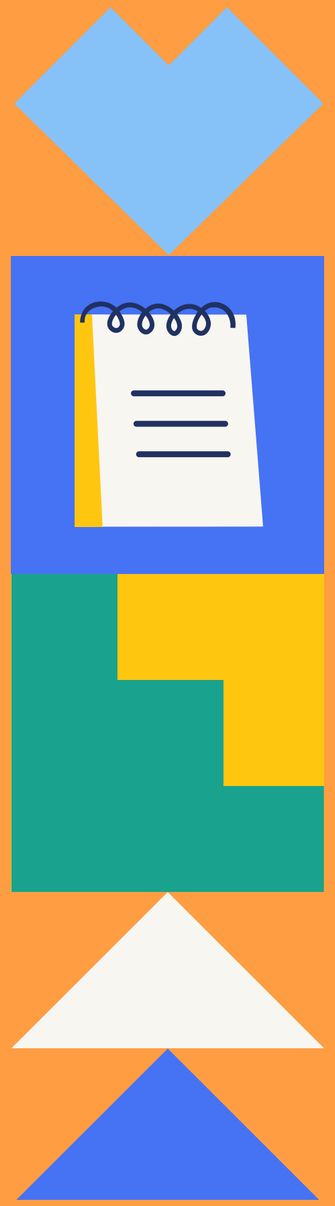
# Random Forest using Python

PRESENTED BY **VARUNN KAUSHIK**



# What We'll Cover Today

1. Laying the Groundwork
Why Random Forests? (Bias-Variance Trade-off)

2. Core Components
Understanding Decision Trees

3. The Ensemble Magic
Randomness in Action: Bagging & Feature Subsampling

4. The Algorithm in Detail
Step-by-Step Construction & Prediction

5. Performance & Validation
Out-of-Bag Error Estimation

# Introduction & Motivation

## The Challenge of Learning from Data

- Supervised Learning Goal: Learn $f : X \to Y$ from $D = \{(x_i, y_i)\}_{i=1}^{n}$
- Two Primary Issues in Modeling:
  - Overfitting (High Variance) - Poor performance on unseen data due to capturing noise.
  - Underfitting (High Bias) - Model too simplistic, fails to capture patterns.

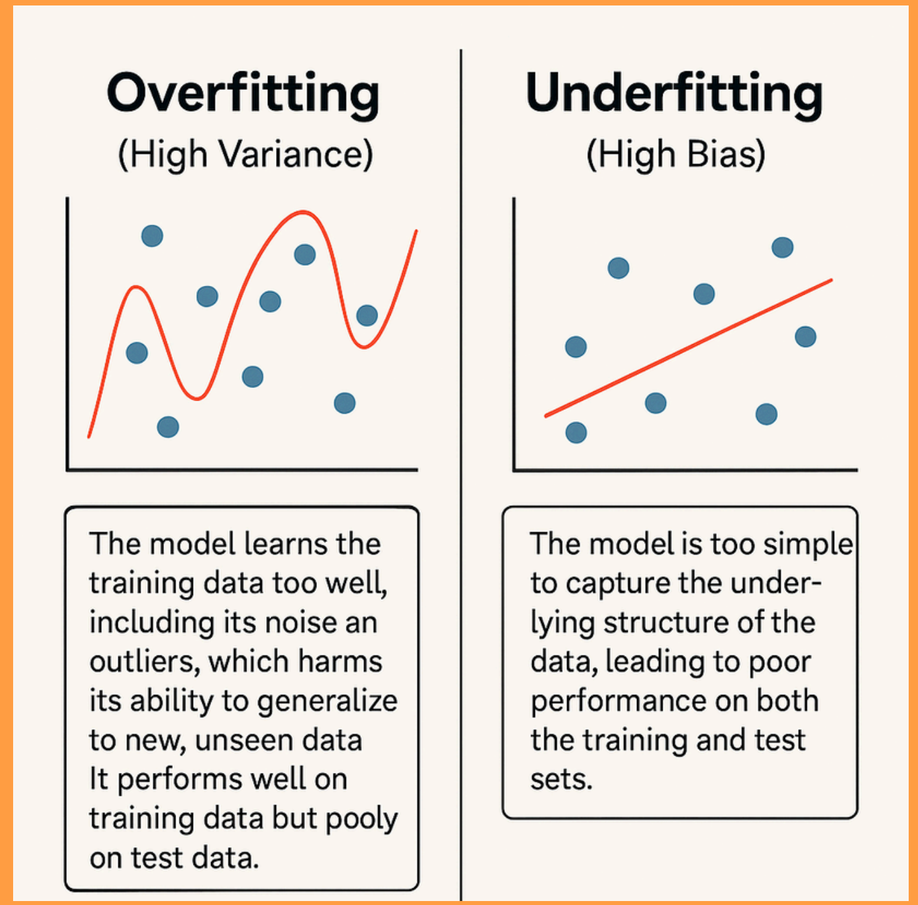
**Bias-Variance Decomposition**

$$E[(y - \hat{f}(x))^2] = \text{Bias}^2[\hat{f}(x)] + \text{Var}[\hat{f}(x)] + \sigma^2$$

- $\hat{f}(x)$: Model's prediction
- $\sigma^2$: Irreducible error (noise)
- Goal: Balance bias and variance to minimize prediction error.
- Ensemble methods (like Random Forests) help by reducing variance.

$\mathbb{E}[(y - \hat{f}(x))^2]$

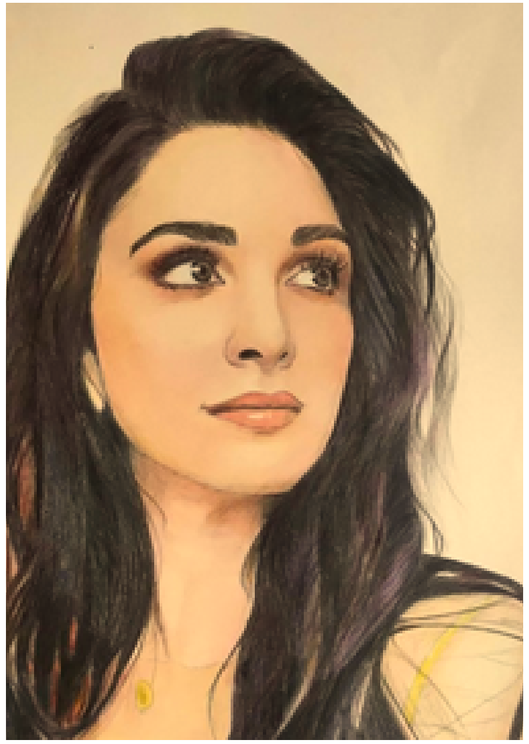
This is the **expected prediction error**.



# Real world problem - Kiara's House Hunt

**Random Forest Algorithm**: A powerful **supervised classification algorithm**
**Concept:** As the name suggests, it creates a "**forest**" comprised of a collection of **individual** "**trees**"
Each tree is a decision-making entity, and together, they form a **robust predictive model**

**Meet Kiara**



### Kiara's Big Decision: Finding Her Dream Home

- **The Challenge:** Kiara needs to buy a house and settle in one of three cities: Mumbai, Delhi, or Bangalore
- **The Goal:** Make a right and optimal choice, considering all perspectives, as this is a very important life decision
- **The Approach:** Kiara decides to seek advice from her best friend

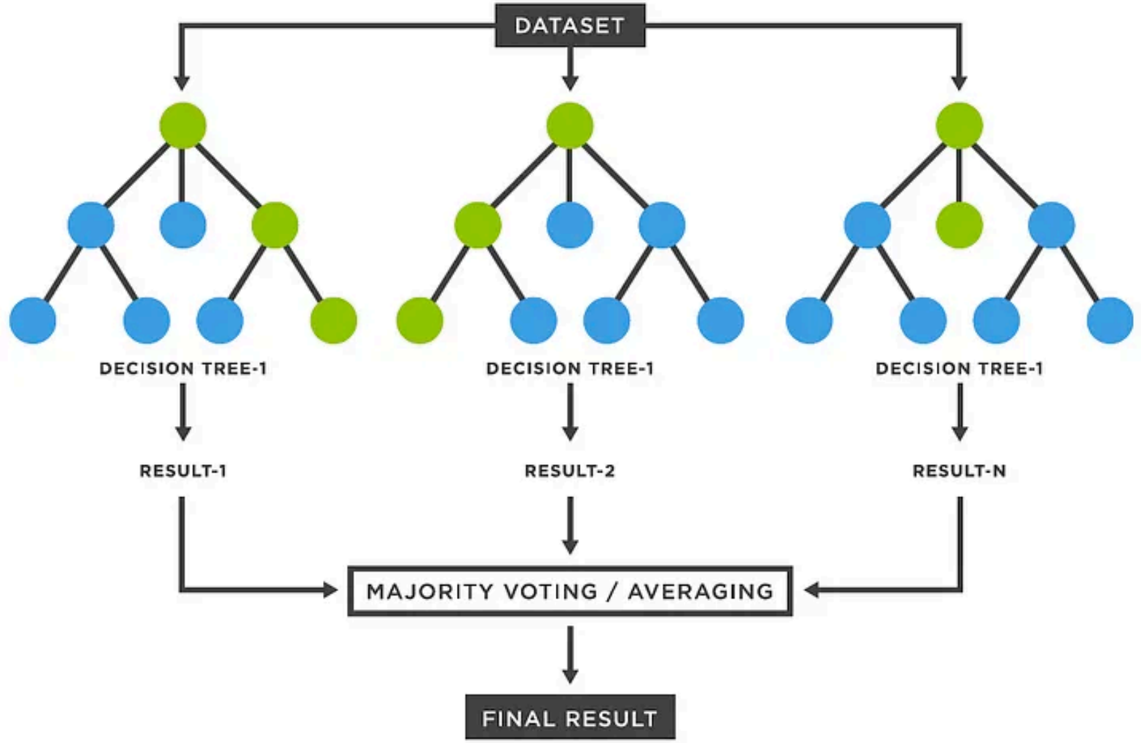
### Friend's Approach: Single Decision Tree in Action

- Kiara's friend begins by asking her specific opinions: "You have visited X city. Did you like it?"
- Based on Kiara's answers, her friend starts forming a recommendation
- This process perfectly mimics a single Decision Tree:Each question is a "**decision node**."Kiara's answer determines the "**branch**" taken.The final recommendation is a "**leaf node**" – a concrete decision

The Power of "Many Friends" (**Random Forest**)
Imagine if Kiara asked **different friends**, **each with their own unique way of asking questions and giving advice**

# The Advantage:
# Why a Forest is Better Than a Single Tree

- **Robustness:** Each tree makes its **own prediction**. The Random Forest then **aggregates these predictions**
- **Reduced Overfitting**: Single decision trees can be prone to overfitting (like one friend giving overly specific or biased advice)
- Random Forest, by combining multiple diverse trees, **averages out individual tree errors** and biases, leading to more generalized and accurate predictions on unseen data
- **Kiara's Benefit:** This ensures a more **reliable and optimal** city **recommendation** for Kiara, similar to getting a consensus from multiple trusted advisors.

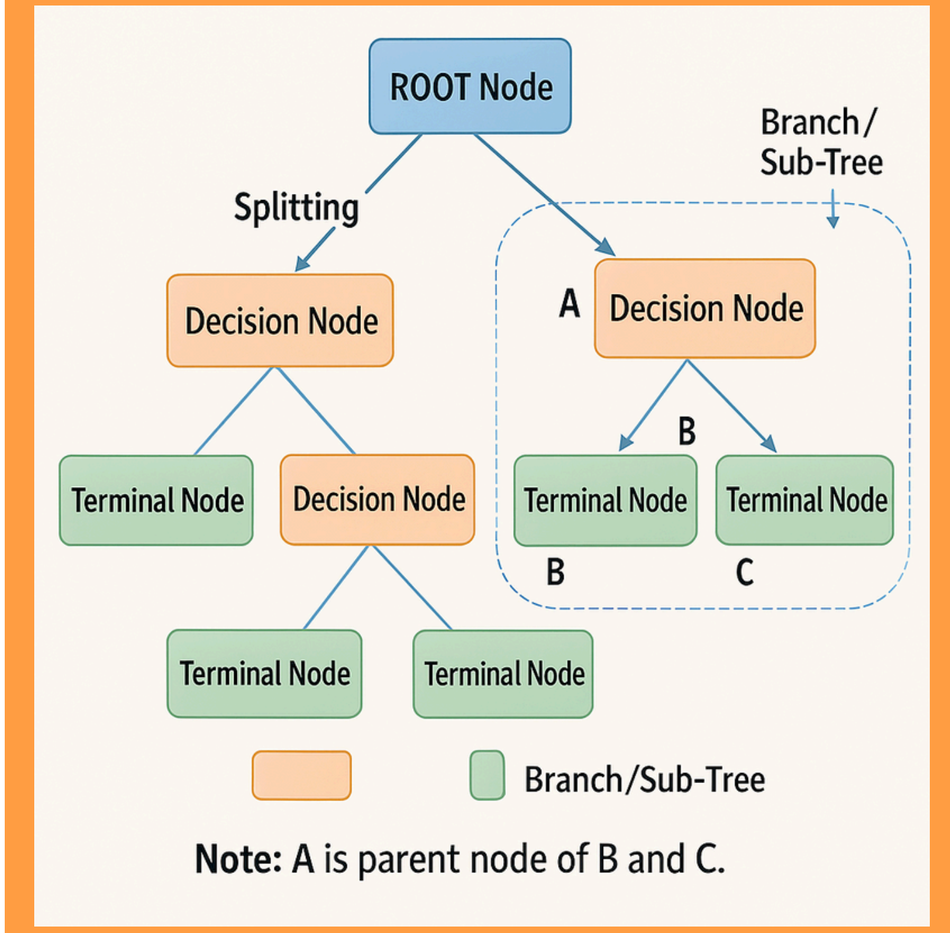


# Deep Dive: Building Blocks of a Decision Tree

Now that we understand how Random Forest uses multiple trees, let's zoom in on the anatomy of a single Decision Tree.

### Key Components We'll Explore:

- **Root Node: The starting point of the tree (Kiara's first question)**
- **Branches: The paths taken based on a decision**
- **Decision Nodes: Intermediate questions/decisions**
- **Splitting Criteria: How the tree decides which question to ask next**
- **Leaf Nodes: The final outcome or prediction (the city recommendation)**




## Types of Decision Trees

**Classification Tree:**

- Target variable is **categorical** (e.g., "Will Employee Leave: Yes/No")
- Prediction based on mode (most frequent class)

**Regression Tree:**

- Target variable is **continuous** (e.g., "Sales Amount")
- Prediction based on mean in a region

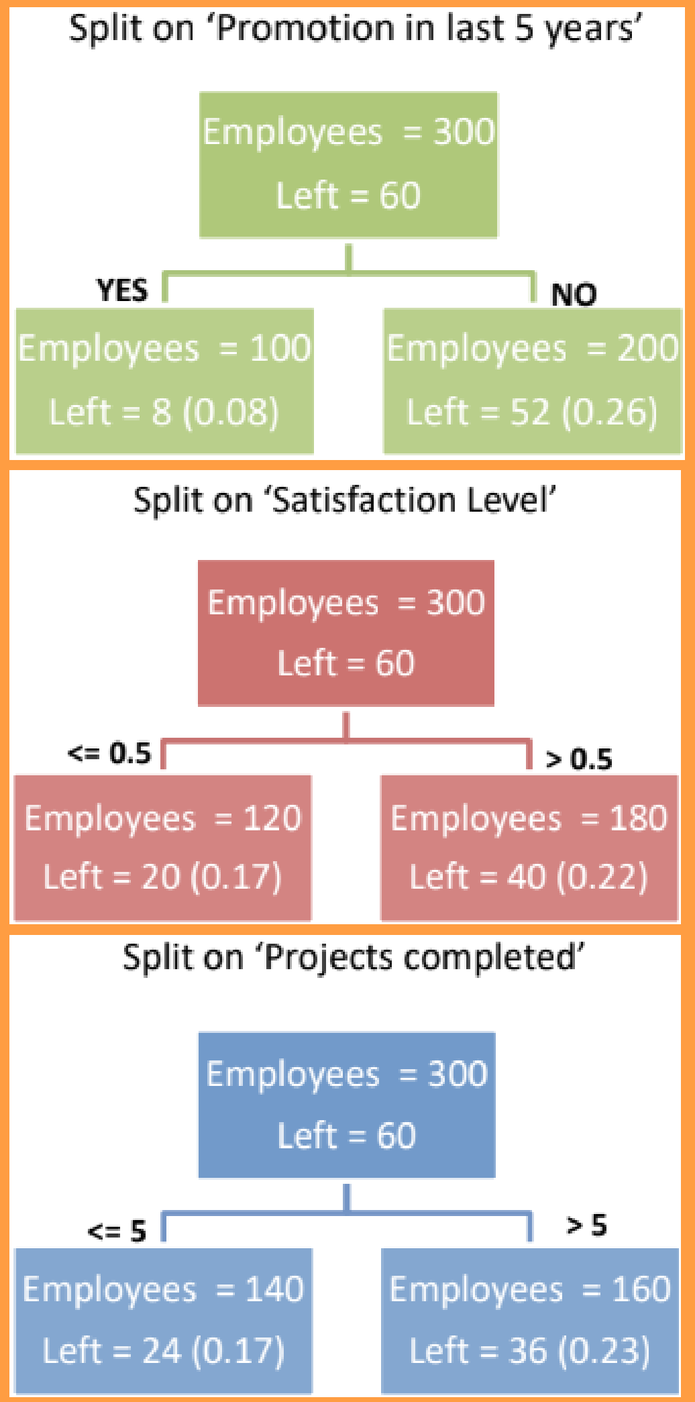
# HR Example Use Case

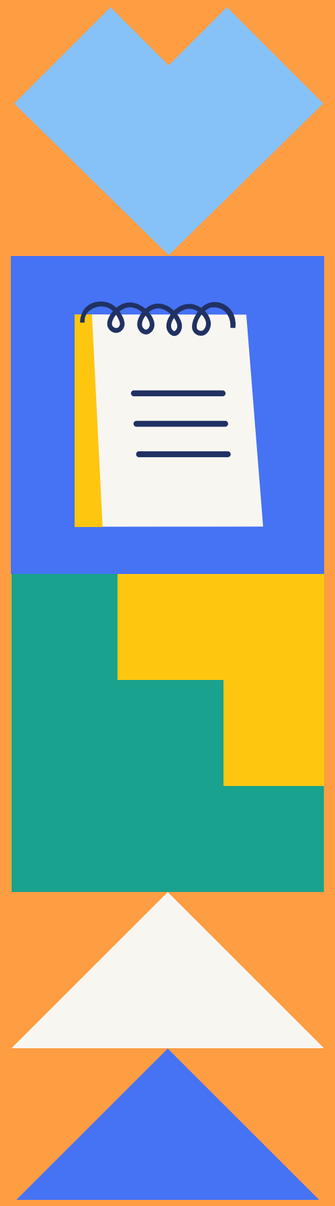
- **Dataset:** 300 employees with:
  - Promotion in last 5 years (Yes/No)
  - Projects completed (<=5 / >5)
  - Satisfaction Level (<=0.5 / >0.5)

- **Goal:** Predict who will leave the company

- **Decision Tree Solution:**
  - Helps segregate data by splitting on significant variables
  - Finds the most informative feature to split the population
  - Creates branches that lead to purest groups of "left" and "stay"

- How does the tree decide which feature and split to choose? Let's explore that using:
  - **Gini Index, Entropy, Chi-Square for Classification Trees**
  - **Reduction in Variance for Regression Trees**





# Splitting Criteria for Classification Trees

VARUNN KAUSHIK

# Gini Index

Gini Index measures how "**mixed up**" the items are.

- **0 = Perfectly pure** (all same)
- **Higher value = More mixed** (more confusion)

Imagine you're separating red and blue balls into two boxes

- If a box has only red balls, it's pure
- If a box has a mix of red and blue balls, it's impure

$$\text{Gini} = 1 - (p^2 + q^2)$$

- $p$ is the proportion of class 1 (e.g., "Stayed")
- $q$ is the proportion of class 2 (e.g., "Left"), and $q = 1 - p$

A decision tree tries to split data in a way that creates the **purest groups possible.impure**

In the HR Case Study, find the best variable to split 300 employees. We want to split based on:

- Whether they got a promotion in last 5 years
- How many projects they completed

🔷 **Step 1: Pick a variable to split (e.g., Promotion)**

- Split the 300 employees into two groups:
  - Group 1: **Yes** (100 employees, 8 left)
  - Group 2: **No** (200 employees, 52 left)

🔷 **Step 2: For each group, calculate the chance of "Left" and "Stayed"**

- Group 1: 8 left → p = 8/100 = 0.08, q = 92/100 = 0.92
  → Gini = $0.08^2 + 0.92^2$ = **0.8528**
- Group 2: 52 left → p = 52/200 = 0.26, q = 0.74
  → Gini = $0.26^2 + 0.74^2$ = **0.6152**

🔷 **Step 3: Combine both groups using Weighted Gini**

Think of it like giving each group a weight depending on how many people it has.

$$\text{Weighted Gini} = \left(\frac{100}{300} \times 0.8528\right) + \left(\frac{200}{300} \times 0.6152\right)$$

$$= 0.6944$$

🟦 **Now Try the Same for "Projects Completed"**

- Group 1: <= 5 projects → 140 employees, 24 left → Gini = 0.7159
- Group 2: > 5 projects → 160 employees, 36 left → Gini = 0.6513

$$\text{Weighted Gini} = \left(\frac{140}{300} \times 0.7159\right) + \left(\frac{160}{300} \times 0.6513\right) = 0.6814$$

Since **0.6814 is lower**, splitting based on **Projects Completed** creates cleaner (purer) groups — so the tree chooses that

# Chi-Square

- Chi-Square checks how **different two groups are from what we would expect**
- It asks: "Is this difference just random, or is it statistically meaningful?"
- Works with **categorical target variables** like “Left” or “Stayed”

Why Use It in a Decision Tree?

- Helps the tree choose the **best variable to split on**
- The bigger the **Chi-Square value, the more significant the split**

$$\text{Chi-square} = \sum \frac{(\text{Actual} - \text{Expected})^2}{\text{Expected}}$$

## 1. Promotion in Last 5 Years

| Node | Left | Stay | Total | Exp Left | Exp Stay | (O−E)² / E |
|---|---|---|---|---|---|---|
| Yes | 8 | 92 | 100 | 20 | 80 | 7.2 + 1.8 |
| No | 52 | 148 | 200 | 40 | 160 | 3.6 + 0.9 |
| **Total** | 60 | 240 | 300 | | | **13.5** |

## 2. Projects Completed

| Node | Left | Stay | Total | Exp Left | Exp Stay | (O−E)² / E |
|---|---|---|---|---|---|---|
| <=5 | 24 | 116 | 140 | 28 | 112 | $\frac{16}{28} \approx 0.571$, $\frac{16}{112} \approx 0.143$ |
| >5 | 36 | 124 | 160 | 32 | 128 | $\frac{16}{32} = 0.5$, $\frac{16}{128} = 0.125$ |
| **Total** | 60 | 240 | 300 | | | **1.339** |

## 3. Satisfaction Level

| Node | Left | Stay | Total | Exp Left | Exp Stay | (O−E)² / E |
|---|---|---|---|---|---|---|
| <=0.5 | 20 | 100 | 120 | 24 | 96 | $\frac{16}{24} = 0.667$, $\frac{16}{96} \approx 0.167$ |
| >0.5 | 40 | 140 | 180 | 36 | 144 | $\frac{16}{36} \approx 0.444$, $\frac{16}{144} \approx 0.111$ |
| **Total** | 60 | 240 | 300 | | | **1.389** |

**Interpretation:**

- A **higher** $\chi^2$ value indicates a **stronger deviation** from the expected distribution — i.e., the split better separates the target classes.
- A **lower** $\chi^2$ value suggests little difference between groups — a weaker split.

# Entropy and Information Gain

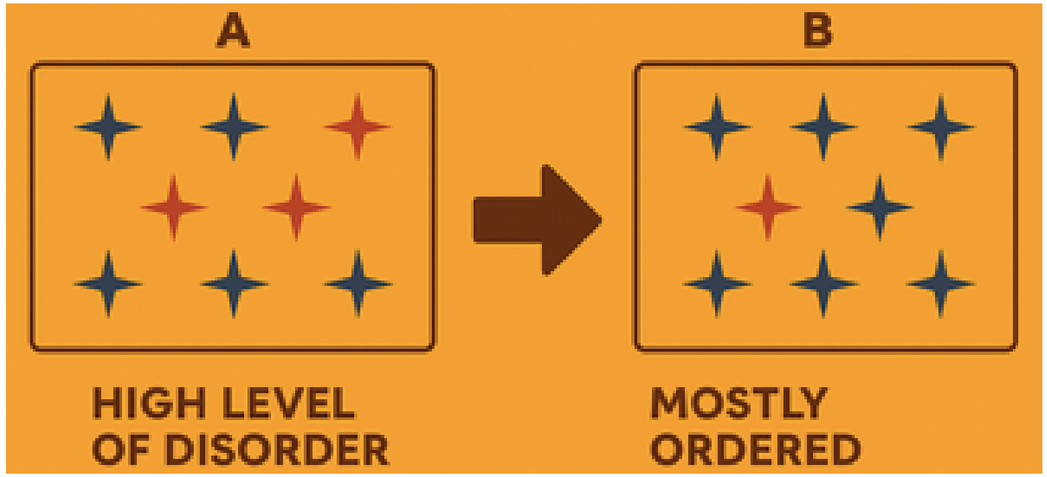
**Entropy**

- Entropy tells us **how mixed or uncertain** a group is
- A group with only blue stars is pure → Entropy = 0
- A group with equal red and blue stars is most mixed → Entropy = 1
- **Lower entropy = better for making decisions** (more predictable/pure group)

**Information Gain**

- Information Gain = What we learned by splitting
- The **higher** the i**nformation gain**, the **better the split.**

$$\text{Information Gain} = \text{Entropy of Parent Node} - \text{Weighted Entropy of Children Nodes}$$




$$\text{Entropy} = -p\log_2(p) - q\log_2(q)$$

Where:

- `p` = % of "Left"
- `q` = % of "Stayed"

Entropy is calculated for each group formed by a split.

Steps to Calculate Entropy for a Split:

- Calculate p and q for each node in the split
- Use the formula above to get each node's entropy
- Multiply each entropy by its group size (weight)
- Add weighted entropies of all groups = Total Entropy for Split
- Lower total entropy means better split

**Cont..**

# Entropy and Information Gain

## 🔷 1. Split on Promotion in Last 5 Years

We split employees into:

- **YES**: 100 employees, 8 left → p = 8/100 = 0.08, q = 92/100 = 0.92
- **NO**: 200 employees, 52 left → p = 52/200 = 0.26, q = 0.74

### 🔷 Step A: Calculate entropy for each group

✅ **For "YES":**

$$\text{Entropy} = -0.08 \log_2(0.08) - 0.92 \log_2(0.92) \approx 0.4022$$

✅ **For "NO":**

$$\text{Entropy} = -0.26 \log_2(0.26) - 0.74 \log_2(0.74) \approx 0.8267$$

### 🔷 Step B: Calculate weighted entropy

$$\text{Weighted Entropy} = \left(\frac{100}{300} \times 0.4022\right) + \left(\frac{200}{300} \times 0.8267\right) = 0.6852$$

## 🔷 2. Split on Projects Completed

- **<= 5**: 140 employees, 24 left → p = 24/140 = 0.17, q = 0.83
- **> 5**: 160 employees, 36 left → p = 36/160 = 0.23, q = 0.77

### 🔷 Step A: Entropies

✅ **For "<=5":**

$$\text{Entropy} = -0.17 \log_2(0.17) - 0.83 \log_2(0.83) \approx 0.6610$$

✅ **For ">5":**

$$\text{Entropy} = -0.23 \log_2(0.23) - 0.77 \log_2(0.77) \approx 0.7692$$

### 🔷 Step B: Weighted Entropy

$$\left(\frac{140}{300} \times 0.6610\right) + \left(\frac{160}{300} \times 0.7692\right) \approx 0.7187$$

## 🔷 3. Split on Satisfaction Level

- **<= 0.5**: 120 employees, 20 left → p = 20/120 = 0.17
- **> 0.5**: 180 employees, 40 left → p = 40/180 = 0.22

### 🔷 Step A: Entropies

✅ **For "<= 0.5":**

$$\text{Entropy} \approx 0.6500$$

✅ **For "> 0.5":**

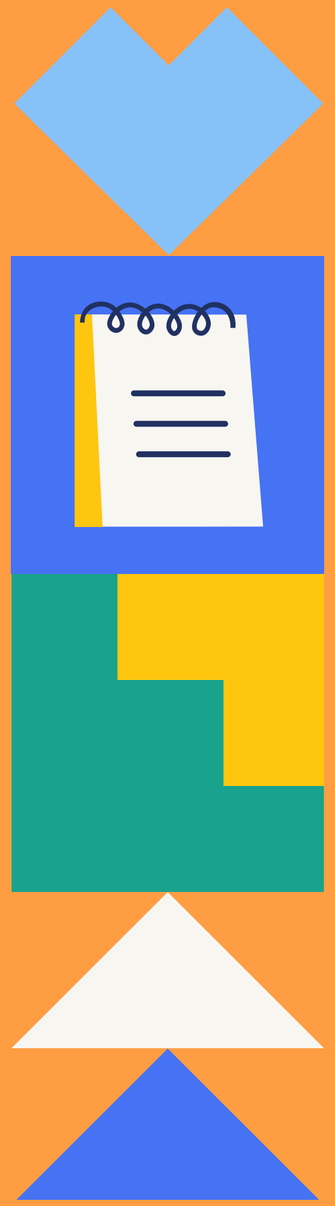
$$\text{Entropy} \approx 0.7642$$

### 🔷 Step B: Weighted Entropy

$$\left(\frac{120}{300} \times 0.6500\right) + \left(\frac{180}{300} \times 0.7642\right) \approx 0.7185$$

Information Gain = **1 - Weighted Entropy of Split**

- **Promotion**: 1 - 0.6852 = **0.3148**
- **Projects**: 1 - 0.7187 = 0.2813
- **Satisfaction**: 1 - 0.7185 = 0.2815
  - **Promotion in Last 5 Years** has the **lowest entropy** (most pure split)
  - Hence, it gives the **highest information gain (0.3148)**
  - So the decision tree will **split on this variable**



# Splitting Criteria for Regression Trees

VARUNN KAUSHIK

# Reduction in Variance

## Nature of Target Variable

- **Gini, Entropy, and Chi-Square** are designed for **categorical targets**
- They measure impurity or class distribution, e.g., what percentage is "Yes" vs. "No".
  - These metrics focus on **probabilities and frequencies of classes** — which don't exist in a numeric (continuous) target
- **Regression trees deal with continuous target variables** (e.g., price, income, age), not discrete categories

1. **Convert the target into numeric values:**
   - In this case:

     "Left" = 1, "Stayed" = 0
2. **Calculate the mean** of the target in each group (after the split)
3. **Use the variance formula** for each group:

$$\text{Variance} = \frac{1}{n}\sum(x_i - \bar{x})^2$$

4. **Combine the variances** using weighted average:

$$\text{Split Variance} = \frac{n_1}{N}Var_1 + \frac{n_2}{N}Var_2$$

## What Are We Optimizing?

| Classification Trees | Regression Trees |
|---|---|
| Minimize **classification error** | Minimize **prediction error (variance)** |
| Measure **impurity** of class labels | Measure **spread** of numerical values |
| Use **Gini / Entropy / Chi-square** | Use **Reduction in Variance (or MSE)** |

**Group 1: Yes**

- 100 employees: 8 left, 92 stayed
- Mean = $\frac{8}{100} = 0.08$
- Variance =

$$\frac{8(1-0.08)^2 + 92(0-0.08)^2}{100} = 0.0736$$

**Group 2: No**

- 200 employees: 52 left, 148 stayed
- Mean = $\frac{52}{200} = 0.26$
- Variance =

$$\frac{52(1-0.26)^2 + 148(0-0.26)^2}{200} = 0.1924$$

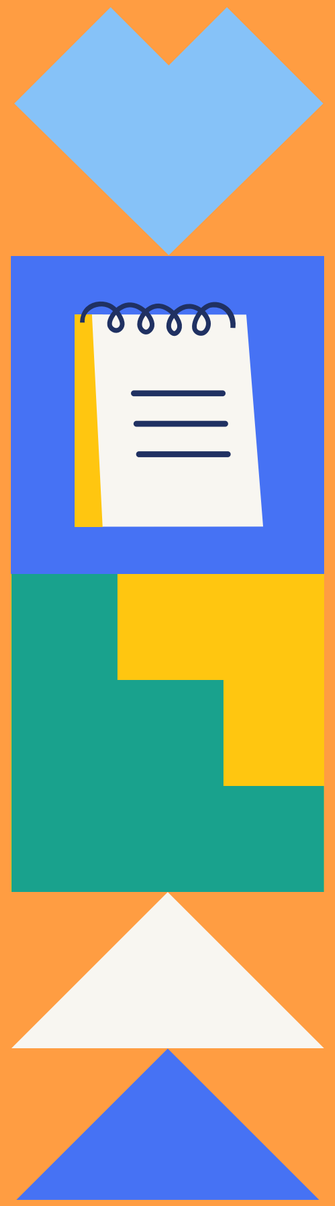
**Final Step: Calculate Weighted Variance of Split**

$$\text{Split Variance} = \left(\frac{100}{300} \times 0.0736\right) + \left(\frac{200}{300} \times 0.1924\right) = \mathbf{0.1528}$$

### Interpretation

This value (0.1528) tells us how **scattered** the data is **after** the split.

The **lower the split variance**, the **better the split.**



# Coming back to Random Forest

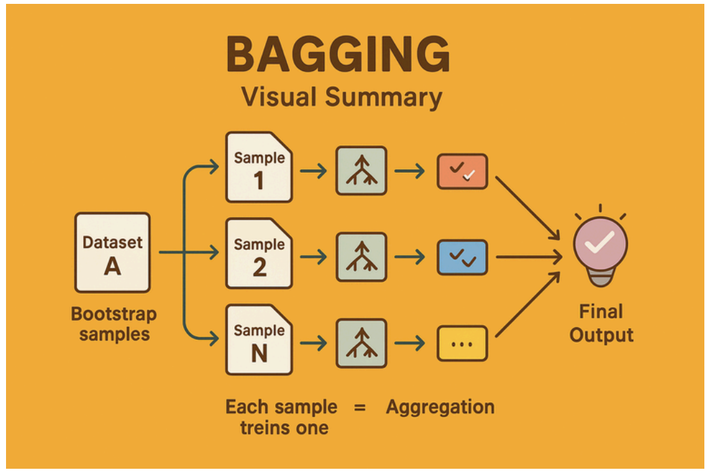
# The "Random" in Random Forest

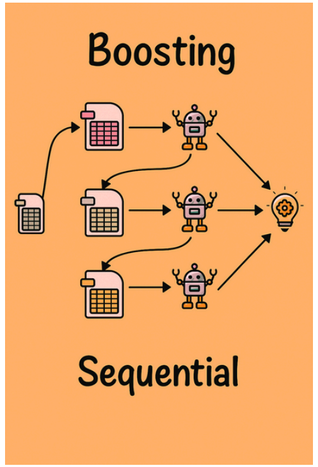
- Decision Trees are intuitive and powerful
- But they can **overfit** on noisy data
- **Random Forests = Ensemble of Decision Trees + Randomness**

### Two Sources of Randomness

- Bootstrap Aggregating (**Bagging**)
- Feature Randomness (**Random Subspace Method**)

| Randomness Type | Purpose | Benefit |
|---|---|---|
| Bagging | Vary data rows | Reduces variance |
| Feature Randomness | Vary columns | Reduces tree correlation |




**Bootstrap Aggregating (Bagging)**

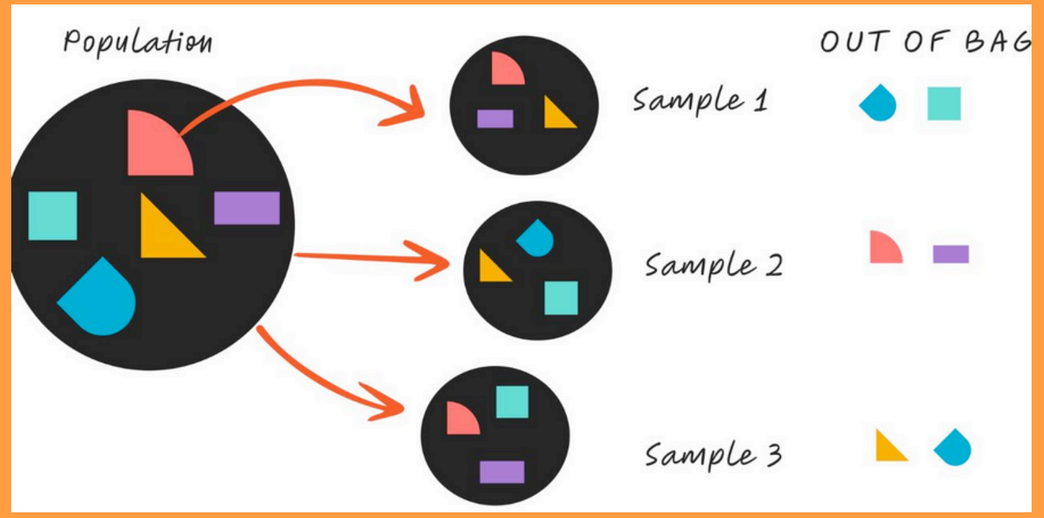
- **Multiple training datasets** are generated by sampling with replacement
- Each Decision Tree is trained on a different sample
- **Classification: Majority vote**
- **Regression: Average prediction**
- **Benefit:** Reduces variance by averaging out individual tree noise

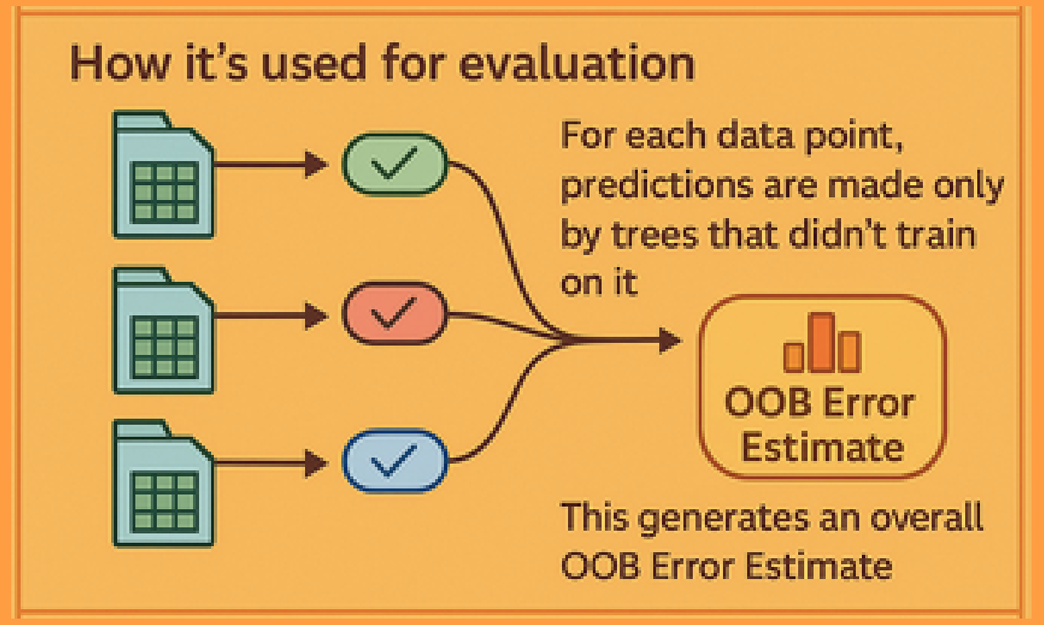
**Feature Randomness (Random Subspace Method)**

- At every split, only a r**andom subset** of features is considered
- Ensures **trees do not always split on the same dominant feature**
- **Benefit:** Decorrelates trees, encourages model diversity

# Out-of-Bag (OOB) Evaluation: Your Model's Internal Report Card

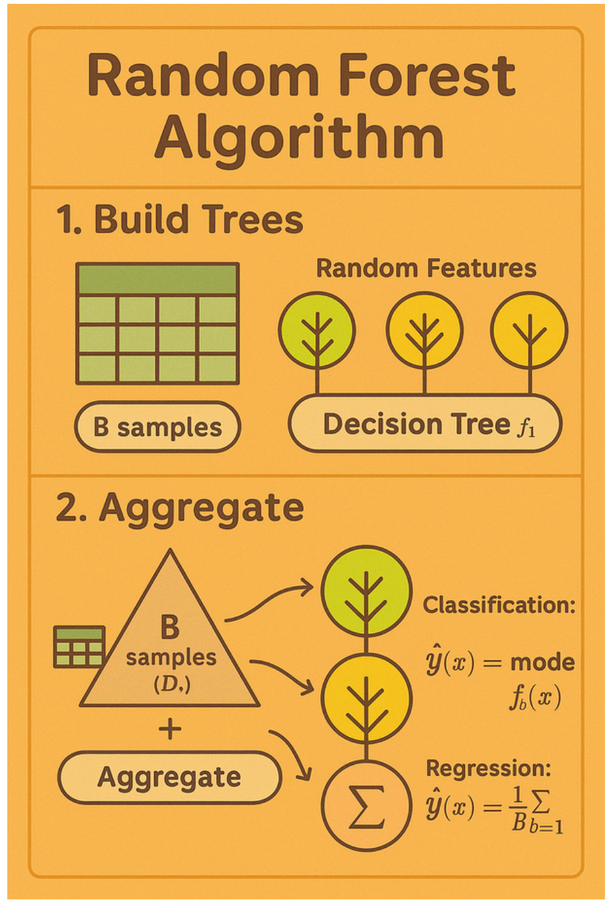
- Random Forests' **built-in validation method**
- **The "Bag":** Each tree trains on a random bootstrap sample (approx. 66% of data).
- **"Out-of-Bag" (OOB):** The ~33% of data not used for a tree's training.
- **How it's used for evaluation:**
  - For each data point, predictions are made only by trees that didn't train on it.
  - These predictions are aggregated (voted/ averaged).
  - This generates an overall OOB Error Estimate.
- **Key Advantages:**
  - Unbiased: Validates on unseen data for each tree.
  - Data-Efficient: No need for a separate validation split.
  - Built-in Efficiency: Like cross-validation, but faster.



# Random Forest Algorithm: Putting It Together

1. **For** $b = 1$ to $B$ (number of trees):
   - Sample, with replacement, $n$ training examples to create a bootstrap sample $D_b$.
   - Train a Decision Tree $f_b$ on $D_b$, considering a random subset of features at each split.
2. **Aggregate Predictions:**
   - **Classification:** $\hat{f}(x) = \text{mode}\{f_b(x)\}_{b=1}^{B}$
   - **Regression:** $\hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x)$

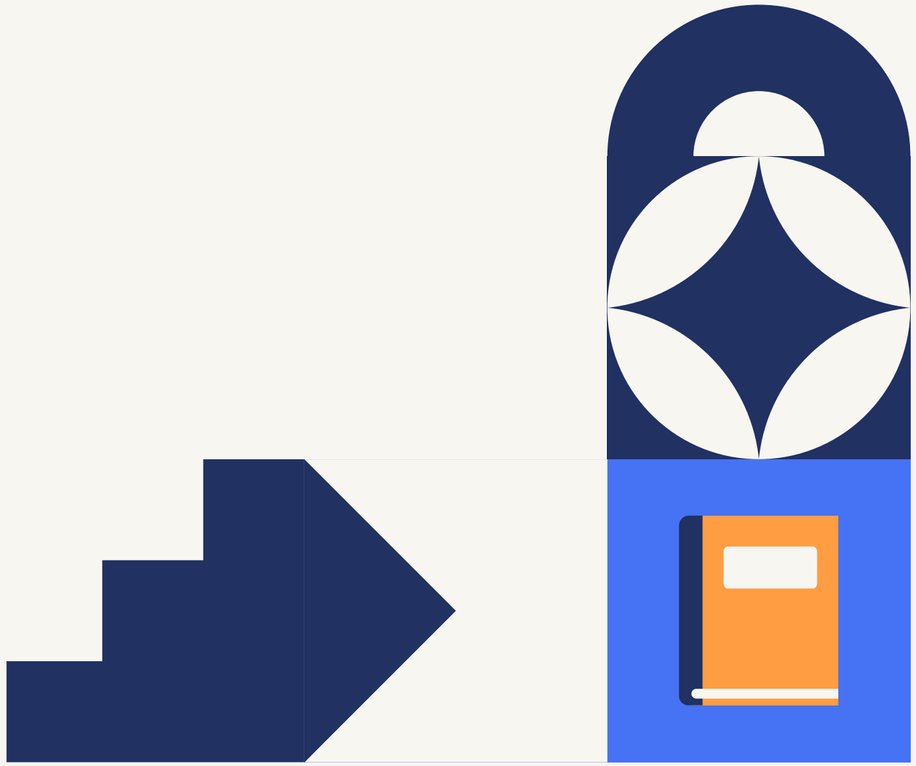



**Advantages:**

- Handles high-dimensional data well.
- Provides estimates of feature importance.
- Robust to outliers and noise.

**Disadvantages:**

- Less interpretable than a single Decision Tree.
- Can be computationally intensive with many trees."



# Let’s move to Python

# Thank You

Varunn Kaushik
LinkedIn | Email: varunnkaushik7@gmail.com | M: 9873975354