Decision Trees and Random Forests

What are Decision Trees?

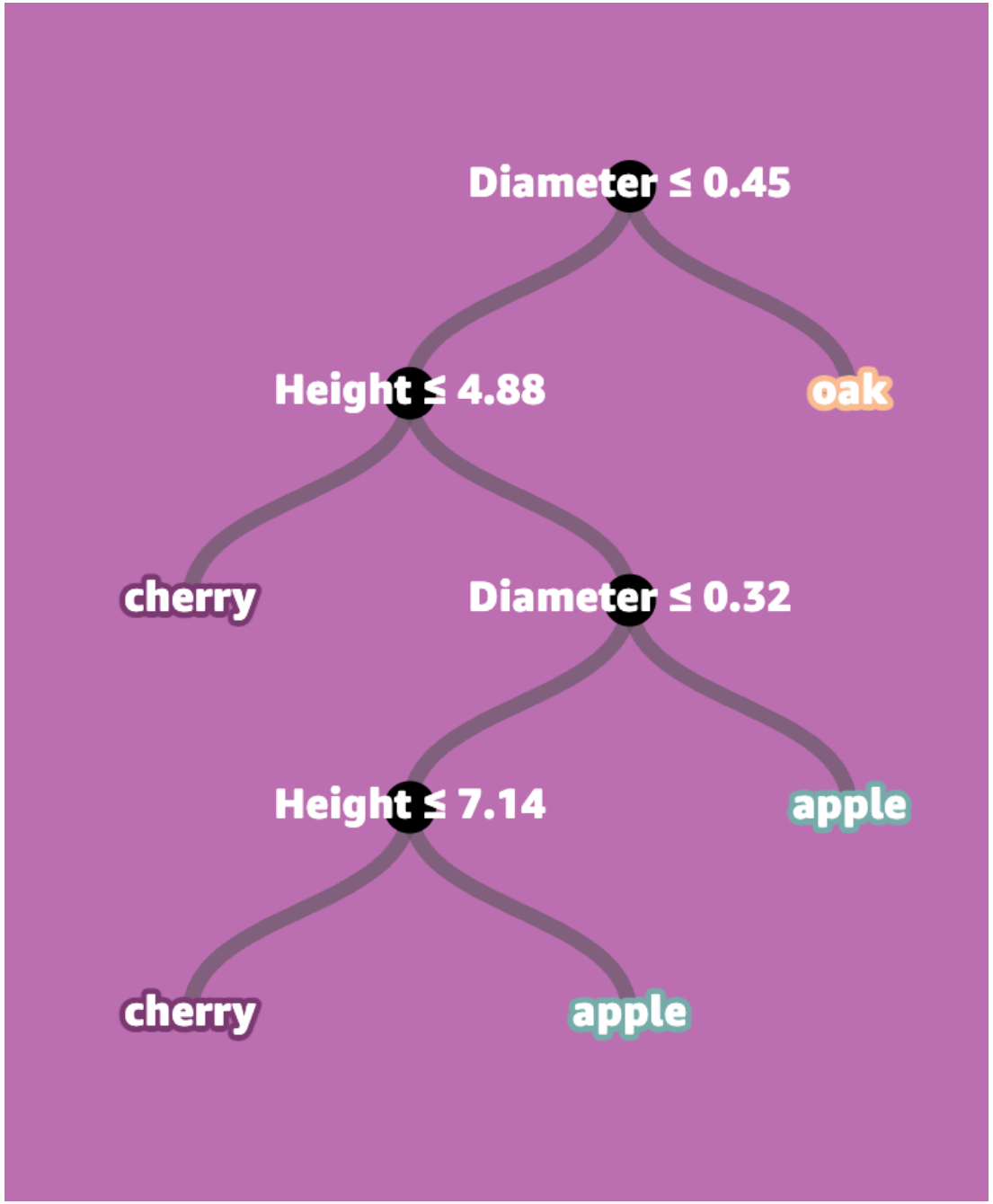
- A Decision Tree is a non-linear predictive model used in machine learning that simulates a series of decision paths, mimicking the human decision-making process. Imagine a tree turned upside down: it starts with a single question or condition (the root), branches out based on possible answers to that question, and ends in decisions or predictions (the leaves).
- Visualize it as a flowchart where each internal node represents a "test" on an attribute (e.g., whether a customer is older than 50 years), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.
- This methodology is not just intuitive but also powerful, allowing complex decision-making processes to be visualized and understood at a glance. It's widely used for both classification and regression tasks in various fields, from customer segmentation in marketing to disease diagnosis in healthcare.

Types of Decision Trees

- • **Classification Trees:** Used when the outcome variable is categorical. These trees split the data into subsets based on the value of input variables, aiming to segregate the classes distinctly at each node. For instance, determining whether an email is spam or not spam based on features like the frequency of certain words.
- • **Regression Trees:** Employed when the outcome variable is continuous or quantitative. These trees also divide the dataset but do so in a way that reduces the variability of the target variable within the subsets. An example would be predicting a person's income based on various demographic factors.
- • Decision Trees offer a practical and visually interpretable means for decision- making, accommodating both discrete and continuous input and output variables. Their versatility and simplicity make them a go-to method for exploratory data analysis, predictive modeling, and even complex ensemble methods like Random Forests.

Anatomy of a Decision Tree

- A Decision Tree is composed of nodes ,branches, and leaves, each representing different aspects of the decision-making process. Here’s how these components fit together:
- **Root Node:** This is the starting point of the tree. It represents the entire dataset, which will be split into two or more homogeneous sets. The root node embodies the initial question or condition affecting the most significant variation (based on a specific metric like Gini impurity or information gain in classification trees, or variance reduction in regression trees).
- **Branches:** Emerging from nodes, branches represent the outcome of the tests carried out on the previous nodes. In essence, they are the "roads" that lead to decisions. Each branch corresponds to a possible value of the tested attribute leading to further partitions of the dataset.
- **Internal Nodes:** These nodes represent points where the data is split further, based on a condition or question. Each internal node performs another test on one of the input features, leading to more branches and nodes. The choice of which feature to split at each node is determined by specific algorithms aiming to maximize the homogeneity of the resultant nodes.
- • **Leaf Nodes(Terminal Nodes):** These are the final nodes at the ends of the tree, where nofurther splitting occurs. In a classification tree, each leaf node represents a class label (the decision
- outcome). In a regression tree, it represents a continuous value or average of the values of the instances that reach the leaf.


Diameter ≤ 0.45
Height ≤ 4.88
oak
cherry
Diameter ≤ 0.32
Height ≤ 7.14
apple
cherry
apple

Split Criteria: The Heart of Decision Making

- To determine where and how splits should be made, decision trees use various metrics:
- **Gini Impurity:** A measure of how often a randomly chosen element would be incorrectly
- labeled if it was randomly labeled according to the distribution of labels in the subset.
- **Entropy/Information Gain:** Entropy measures the disorder or uncertainty, and the information gain is the decrease in entropy after a dataset is split on an attribute. It identifies the split that maximizes the information gain (reduces uncertainty).
- **Variance Reduction:** Used in regression trees, this criterion aims to reduce the variance of the target variable within each node.
- The anatomy of a decision tree reveals a structured approach to decision-making, where each component plays a specific role in segregating the dataset into more homogenous subsets. Understanding these elements is crucial for comprehending how decision trees make predictions and how they can be optimized for better performance.

How Decision Trees Work

- • Decision Trees operate on the principle of recursive partitioning: the process starts at the root node and splits the data on the feature that results in the most homogeneous sub-nodes according to a chosen metric. This section delves into how features are selected for splitting and the criteria used to measure the effectiveness of these splits.
- • **Feature Selection: The Quest for the Optimal Split**
- • Feature selection in the context of decision trees is about identifying the attribute that best divides the dataset into subsets that are as pure as possible. The goal is to simplify the complexity of the problem at each step, moving closer to a decision with each split.
- • **Criteria for Feature Selection:**
  • The reduction in heterogeneity post-split.
  • Maximizing information gain or minimizing impurity in classification tasks. • Minimizing variance in regression tasks.

Gini Impurity

- Gini Impurity is a measure used in decision trees to determine the probability of a randomly chosen element being incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. It is a key metric for the CART (Classification and Regression Trees) algorithm for deciding on splits.
- **Mathematical Formulation**
- The Gini Impurity of a dataset can be calculated using the formula: $$Gini(D) = 1 - \sum_{i=1}^{n} p_i^2$$
- Where:
- D represents the dataset for a node,
- N is the number of classes,
- P_i is the proportion of the samples that belong to class i in the dataset.
- **Lower Gini Impurity Indicates Higher Purity**
- A Gini score of 0 indicates perfect purity; all samples in the node belong to the same class.
- As the score approaches 1, the distribution of class labels becomes more uniform, indicating higher impurity.

Post-Split Decision Making in Decision Trees

- • After a split based on the optimal feature is made at a node, the decision tree algorithm proceeds to further grow the tree by evaluating the subsets (child nodes) created by the split. Here’s what happens next:
- **1.Recalculate Gini Impurity for Each Child Node:** For every subset created by the split, we recalculate the Gini impurity to understand the homogeneity of the new child nodes. This recalculated impurity serves as a baseline for further splits.
- **2.Consider All Features Again for the Next Split:** The algorithm then evaluates all available features (including the one used in the previous split) to determine the next best feature to split on for each child node. The decision is based on which feature further reduces the Gini impurity the most from this new baseline.
- **3.Iterative Feature Selection:** This process of evaluating features and splitting based on the one that offers the most significant reduction in impurity continues at each child node. The same feature can be used for subsequent splits if it continues to offer the best division at the next level of the tree, although it's evaluated in the context of the current subset of data.

Stopping Criteria: When Does a Node Become a Leaf?

- • Theprocessofsplittingandcreatingnewnodescontinuesuntiloneofthefollowingstoppingcriteriaismet, at which point a node is considered a leaf and does not split further:
- **1. All Instances Belong to the Same Class:** If after a split, all instances in a node belong to the same class, there's no further impurity to reduce, and the node becomes a leaf.
- **2. No Remaining Features Offer Improvement:** If there are no remaining features that would reduce Gini impurity (i.e., no split can improve homogeneity), the node becomes a leaf.
- **3. Minimum Node Size Reached:** Many algorithms allow setting a minimum node size. If splitting a node would result in a child node smaller than this minimum size, the split is not performed, and the node becomes a leaf.
- **4. Maximum Depth Reached:** If the tree has reached the maximum depth specified as a parameter, the node becomes a leaf to prevent the tree from growing too deep, which might lead to overfitting.
- **5. Pre-defined Minimum Improvement Threshold:** If the improvement in impurity reduction from a potential split is below a pre-defined threshold, the node becomes a leaf, indicating marginal gains are not worth further splitting.
- • Thedecisiontree'sdevelopmentisameticulousprocessinvolvingrecalculatingGiniimpurityandreassessing all features at each new node. It balances the pursuit of purity with practical limits to ensure robust, generalizable models. Stopping criteria are essential to finalize the tree structure, preventing over- complexity and overfitting, and ensuring the model remains interpretable and efficient.

Example

-
- Dataset
- We're predicting whether a day is suitable for playing outside based on temperature. Our dataset, sorted by temperature, is as follows:
-
- Step 1: Sort the Data
- Our dataset is already sorted by temperature:

Temperature (°C)	Play
18	No
20	No
26	Yes
28	Yes
30	Yes

- Step 2: Identify Potential Split Points
- We calculate the potential split points as the midpoints between consecutive temperatures:
-
- - Between 18 and 20: (18 + 20) / 2 = 19
- - Between 20 and 26: (20 + 26) / 2 = 23
- - Between 26 and 28: (26 + 28) / 2 = 27
- - Between 28 and 30: (28 + 30) / 2 = 29
-
- Step 3: Calculate Gini Impurity for Each Split
- We calculate the Gini impurity for each potential split point:
-
- $Gini(S) = 1 - \sum_{i=1}^{k} p_i^2$
- Where p is the proportion of the class i elements in set S.
- K is the number of different classes

$$\text{Weighted Gini} = \sum_{j=1}^{m} \left(\frac{|S_j|}{|S|} \right) \times \text{Gini}(S_j)$$

- Where $|S_j|$ is the number of samples in subset J after the split.
- $|S|$ is the total number of samples before the split.
-
- $\text{Gini}(S_j)$ is the Gini impurity of subset j
- m is the number of subsets created by the split
- Split at 19°C:
- - Below 19°C: [No]
- - Gini =1 - (1^2) = 0
- - Above 19°C: [No, Yes, Yes, Yes]
- - Gini = 1 - $((1/4)^2 + (3/4)^2)$ = 0.375
- - Weighted Gini:
- $\frac{1}{5}(0) + \frac{4}{5}(0.375) = 0.3$
-
-

- Split at 23°C:
- - Below 23°C: [No, No]
- - Gini = 1 - (1^2) = 0
- - Above 23°C: [Yes, Yes, Yes]
- - Gini = 1 - (1^2) = 0
- - Weighted Gini:
- $\frac{2}{5}(0) + \frac{3}{5}(0) = 0$
-
- Split at 27°C:
- - Below 27°C: [No, No, Yes]
- - Gini = 1 - $((2/3)^2 + (1/3)^2)$ = 0.444
- - Above 27°C: [Yes, Yes]
- - Gini = 1 - (1^2) = 0
- - Weighted Gini:
- $\frac{3}{5}(0.444) + \frac{2}{5}(0) = 0.267$
-
- Split at 29°C:
- - Below 29°C: [No, No, Yes, Yes]
- - Gini = 1 - $((2/4)^2 + (2/4)^2)$ = 0.5
- - Above 29°C: [Yes]
- - Gini = 1 - (1^2) = 0
- - Weighted Gini:
- $\frac{4}{5}(0.5) + \frac{1}{5}(0) = 0.4$

Step 4: Select the Best Split
From these calculations:
- The split at 23°C has a Weighted Gini of 0, indicating perfect segregation between "No" and "Yes". This is the optimal threshold to use for this decision tree node because it perfectly classifies the data with zero impurity.

Information Gain

- Information Gain measures the reduction in entropy or uncertainty about the target variable after partitioning the dataset based on a particular feature. In the context of decision trees, it helps to identify the feature that best separates the classes, thus making the dataset more orderly or "pure" after the split.
- **Mathematical Formulation of Information Gain**
- The Information Gain (IG) can be calculated using the formula:

$$IG(D, A) = Entropy(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} \times Entropy(D_v)$$

- Where:
- - IG(D, A) is the information gain of dataset D after splitting on attribute A.
- - Entropy(D) is the entropy of the whole dataset before the split.
- - Values(A) are the different values attribute A can take.
- - $|D_v|$ is the number of instances in D that have value v for attribute A.
- - Entropy(D_v) is the entropy of the subset of D for which attribute A has value v.

- **Entropy: The Starting Point**
- Entropy for a dataset is defined as:

$$Entropy(D) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- *pi* is the proportion of the instances in *D* that belong to class *i*.

Example

- Dataset
- Consider a small dataset for predicting whether someone will go hiking based on weather conditions:

Temperature (°C)	Humidity (%)	Wind (km/h)	Hike
18	80	10	No
20	70	15	No
26	60	20	Yes
28	55	5	Yes
30	50	10	Yes

- Step 1: Calculate Overall Entropy
- First, we calculate the entropy for the entire dataset (before any splits). The target variable "Hike" has two classes: "Yes" and "No".
-
- - Count of "Yes": 3
- - Count of "No": 2
- - Total: 5
-
- Entropy is calculated using the formula: $\text{Entropy}(S) = -\sum_{i=1}^{k} p_i \log_2(p_i)$
- Where p_i is the proportion of class i in the dataset. For our dataset:

$$\text{Entropy}(S) = -\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right) \approx -(0.6 \times -0.737 + 0.4 \times -1.322) = 0.971$$

- Step 2: Calculate Entropy After Each Potential Split
- We'll evaluate splits based on the "Temperature" feature. Assume potential split points at 23°C and 27°C:
-
- Split at 23°C:
- - Below 23°C ("No", "No"):
- - Entropy = -(1 $\log_2$ 1 + 0 $\log_2$ 0) = 0
- - Above 23°C ("Yes", "Yes", "Yes"):
- - Entropy = -(1 $\log_2$ 1 + 0 $\log_2$ 0) = 0
-
- - Weighted Entropy:
- $\frac{2}{5}$ (0) + $\frac{3}{5}$ (0) = 0
-
- Split at 27°C:
- - Below 27°C ("No", "No", "Yes"):
- - Entropy = -(2/3 $\log_2$ 2/3 + 1/3 $\log_2$ 1/3) = 0.918
- - Above 27°C ("Yes", "Yes"):
- - Entropy = -(1 $\log_2$ 1 + 0 $\log_2$ 0) = 0
- - Weighted Entropy:
- $\frac{3}{5}$ (0.918) + $\frac{2}{5}$ (0) = 0.551
-

- Step 3: Calculate Information Gain for Each Split
- Information Gain is calculated by subtracting the weighted entropy of the split from the original entropy of the dataset

$$\text{Information Gain} = \text{Entropy}(S) - \text{Weighted Entropy}$$

-
- - Split at 23°C:
- - Information Gain = 0.971 - 0 = 0.971
- - Split at 27°C:
- - Information Gain = 0.971 - 0.551 = 0.420
-
- Step 4: Select the Best Split
- The best split is the one with the highest Information Gain. In this case, splitting at 23°C yields the highest Information Gain, thus it is the optimal point for the first split.
-
-

- Let’s continue with the example by calculating the Information Gain for splitting based on the "Humidity" feature from the dataset. We'll select potential split points based on the humidity values.
-
-
- Step 1: Identify Potential Split Points for Humidity
- We consider potential split points at the midpoints between consecutive humidity values:
-
- - Between 80 and 70: (80 + 70) / 2 = 75
- - Between 70 and 60: (70 + 60) / 2 = 65
- - Between 60 and 55: (60 + 55) / 2 = 57.5
- - Between 55 and 50: (55 + 50) / 2 = 52.5

- Step 2: Calculate Entropy After Each Potential Split
- Let's calculate the entropy for each segment formed by these splits, using the overall entropy Entropy(S) = 0.971 as previously calculated.
-
- Split at 75% Humidity:
- - Below 75%: [No, No]
- - Entropy = -(1 \log$_2$ 1 + 0 log$_2$ 0) = 0
- - Above 75%: [Yes, Yes, Yes]
- - Entropy = -(1 \log$_2$ 1 + 0 log$_2$ 0) = 0
- - Weighted Entropy:
- - $\frac{2}{5}$ (0)+ $\frac{3}{5}$ (0) = 0
-
- Split at 65% Humidity:
- - Below 65%: [No, No, Yes]
- - Entropy = -(2/3 log$_2$ 2/3 + 1/3 log$_2$ 1/3) = 0.918
- - Above 65%: [Yes, Yes]
- - Entropy = -(1 log$_2$ 1 + 0 log$_2$ 0) = 0
- - Weighted Entropy:
- $\frac{3}{5}$ (0.918) + $\frac{2}{5}$ (0) = 0.551

- Split at 57.5% Humidity:
- - Below 57.5%: [No, No, Yes, Yes]
- - Entropy = -(2/4 $\log_2$ 2/4 + 2/4 $\log_2$ 2/4) = 1.0
- - Above 57.5%: [Yes]
- - Entropy = -(1 $\log_2$ 1 + 0 $\log_2$ 0) = 0
- - Weighted Entropy:
- $\frac{4}{5}$ (1) + $\frac{1}{5}$ (0) = 0.8
-
- Split at 52.5% Humidity:
- - Below 52.5%: [No, No, Yes, Yes, Yes]
- - Entropy = -(2/5 $\log_2$ 2/5 + 3/5 $\log_2$ 3/5) = 0.971
- - Above 52.5%: []
- - Entropy = 0 (no data)
- - Weighted Entropy:
- $\frac{5}{5}$ (0.971) + $\frac{0}{5}$ (0) = 0.971 (but practically, it should be ignored due to no data on one side)

- Step 3: Calculate Information Gain for Each Split
- Using the entropy values and weighted entropies calculated, the Information Gain for each split is:
-
- - Split at 75%:
-
- - Information Gain = 0.971 - 0 = 0.971
- - Split at 65%:
- - Information Gain = 0.971 - 0.551 = 0.420
- - Split at 57.5%:
- - Information Gain = 0.971 - 0.8 = 0.171
- - Split at 52.5%:
- - Information Gain = 0.971 - 0.971 = 0.0 (considered irrelevant due to lack of data split on one side)
-
- Step 4: Select the Best Split
- From the calculations, the best split for "Humidity" occurs at 75% with an Information Gain of 0.971, the same as the optimal split for "Temperature" at 23°C.
-

Regression Example

-
- Let's consider a detailed example of building a decision tree for a regression task using a simple dataset. In regression trees, instead of predicting a class label, we predict a continuous value. The decision tree will try to minimize the variance within each node as a measure of impurity.
-
- Dataset
- Suppose we have a small dataset that shows the relationship between the number of study hours and the scores on a test:

Study Hours	Score
1	58
1	60
3	65
4	70
5	72
6	76
7	80
8	83
9	88
10	90

- Step 1: Calculate Overall Variance
- First, we calculate the variance for the entire dataset, which is our target to minimize with splits. Variance is calculated as:

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

- Where x_i is each score, $\bar{x}$ is the mean score, and n is the number of scores.
-
- Calculating the mean score: $\bar{x} = \frac{58 + 60 + 65 + 70 + 72 + 76 + 80 + 83 + 88 + 90}{10} = 74.2$
- Calculating the variance: $\text{Variance}(S) = \frac{(58 - 74.2)^2 + (60 - 74.2)^2 + \ldots + (90 - 74.2)^2}{10} = 96.96$

- Step 2: Evaluate Possible Splits
- We consider splitting the dataset at each midpoint between consecutive study hours:
-
- Split at 2 Hours:
- - Below or equal to 2 Hours: Scores = [58, 60]
- - Mean = (58 + 60) / 2 = 59
- - Variance = $((58-59)^2 + (60-59)^2)/2 = 0.5$
- - Above 2 Hours: Scores = [65, 70, 72, 76, 80, 83, 88, 90]
- - Mean = (65 + 70 + 72 + 76 + 80 + 83 + 88 + 90) / 8 = 78
- - Variance = $((65-78)^2 + \ldots + (90-78)^2)/8 = 86.75$
- - Weighted Variance:
- $\frac{2}{10}(0.5) + \frac{8}{10}\ 86.75 = 69.4$
-
- Similar calculations are done for each potential split point:
-
- - At 2.5 Hours
- - At 3.5 Hours
- - At 4.5 Hours
- - At 5.5 Hours
- - At 6.5 Hours
- - At 7.5 Hours
- - At 8.5 Hours

- Step 3: Calculate Reduction in Variance for Each Split
- For each split, the Reduction in Variance is calculated as the difference between the variance of the entire dataset and the weighted variance after the split.
-
- For the split at 2 Hours: $\text{Reduction in Variance} = 96.96 - 69.4 = 27.56$

- Step 4: Select the Best Split
- We calculate the reduction in variance for all potential split points and select the one with the highest reduction in variance as the optimal split point. This will continue recursively for each subset until a stopping criterion is met, such as a minimum number of samples in a node or a minimum reduction in variance achieved.
-

- After a decision tree is constructed for a regression task, making predictions with it involves traversing the tree based on the input features of the data point for which a prediction is needed. Here's how this process typically works:
-
- 1. Traversal of the Tree
- The decision tree consists of nodes and decision points starting from the root. Each node in the tree represents a decision point where the tree splits based on the value of a certain feature relative to a threshold. The traversal starts at the root and moves down the tree:
- - At each node, the value of the feature that the node splits on is compared to the threshold value for that node.
- - If the value of the feature is less than or equal to the threshold, the traversal moves to the left child of that node.
- - If the value is greater than the threshold, the traversal moves to the right child.

- 2. Reaching a Leaf Node
- The process continues until a leaf node is reached. A leaf node does not have any children and does not split further. It represents the final decision or prediction of the tree for input features that lead to this node.
-
- 3. Prediction Output
- In a regression tree, each leaf node contains a prediction value which typically is the mean or median of the target values of the training samples that fall into that leaf. This value is used as the predicted output for any input that reaches this leaf:
- - Mean Value Prediction: The average of all the target values in the training data that end up in the leaf node after the tree has been trained. This is the most common method.
- - Median Value Prediction: Sometimes the median is used instead of the mean, especially if the data in the leaf is skewed or has outliers, as the median is more robust to such anomalies.
-
- Example Walkthrough
- Let’s say you have a regression tree trained to predict house prices based on features like size (sq ft) and age (years). Here is how a prediction would be made:
- - Input: A house with 2000 sq ft and 10 years old.
- - Root Node: Checks if size > 1500 sq ft. Since 2000 > 1500, move to the right child.
- - Next Node: Checks if age > 15 years. Since 10 < 15, move to the left child.
- - Leaf Node Reached: The leaf node might have a predicted house price value based on the historical data of similar houses (e.g., the average price of houses with sizes around 2000 sq ft and ages around 10 years that were used to train the tree).
-

Random Forests

- At its core, a Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Developed by Leo Breiman and Adele Cutler, Random Forests build upon the simplicity of decision trees with the added sophistication of ensemble techniques to mitigate their limitations, notably overfitting.

The Ensemble Learning Paradigm:

- **Conceptual Foundation:** Ensemble learning is a machine learning paradigm where multiple models (often of the same type) are strategically combined to solve a particular computational intelligence problem. The central thesis is that a group of "weak learners" can come together to form a "strong learner."
- • **Strength in Numbers:** The underlying principle of ensemble methods is that a diverse collection of models, when aggregated, can achieve better predictive performance and robustness than any single constituent model alone. This approach leverages the diversity among the models to reduce errors from variance (overfitting) and bias (underfitting).

- **How Random Forests Work:**
- **1.Bootstrap Aggregating (Bagging):** Random Forests apply the bagging technique, where multiple decision trees are trained on different subsets of the training data, sampled with replacement. This process introduces diversity among the trees through varied training samples.
- **2.Feature Randomness:** When splitting a node during the construction of the tree, the choice of the feature to split on is made from a random subset of the total features. This "feature bagging" further increases the diversity among the trees and contributes to the reduction of variance without significantly increasing bias.
- **3.Aggregation of Predictions:** For classification tasks, the final prediction is typically made by majority voting among the decision trees. For regression tasks, the average outcome of all trees is considered.

How Random Forests Work

- **Multiple Decision Trees: A Forest from the Trees**
- **Foundation in Diversity:** The strength of a Random Forest lies in its diversity; each tree in the forest is trained on a slightly different data sample and uses a different subset of features for splitting at each node. This diversity results in a collection of moderately correlated trees whose errors can cancel out when averaged.
- **Independent Growth:** Each tree is grown to its maximum extent without pruning, allowing the trees to fully capture the patterns in their respective subsets of the data. Although this might lead to individual trees overfitting their samples, the ensemble approach mitigates this risk.
- **2.Bagging (Bootstrap Aggregating): Sampling with Replacement**
- **Creating Unique Datasets:** Random Forests employ bagging by creating bootstrapped datasets. Foreach tree, a new dataset is generated by randomly sampling instances from the original dataset with replacement, ensuring each tree has a unique training dataset.
- **Reduction of Variance:** This process helps reduce variance without increasing bias, as each tree gets to see different facets of the data, making the ensemble less likely to overfit than any single tree.
- •

- **3.Feature Randomness : Sowing Seeds of Uniqueness**
- **Random Feature Selection:** When constructing each tree, at every split, a randomly selected subset of features is considered. This method ensures that the trees are not identical and each tree has a chance to discover and utilize different predictors (features) that might be predictive of the outcome.
- **Balance Between Prediction and Diversity:** By limiting the number of features available for each split, Random Forests force each tree to be different and to not overly rely on any single powerful predictor, promoting model diversity and reducing the correlation between trees.
- **The Aggregation Mechanism: Wisdom of the Crowd**
- **Majority Voting(Classification):** For classification, the prediction of the ensemble is the mode of the predictions made by individual trees. This "majority votes" approach can significantly reduce the classification error rate compared to individual trees.
- **Averaging (Regression):** In regression tasks, the final prediction is the average of the predictions from all trees in the forest. This averaging process smooths out predictions, reducing the risk of extreme values and hence lowering the variance.

Feature	Decision Trees	Random Forests
Basic Concept	A single tree structure that models decisions and their possible consequences.	An ensemble of decision trees, combining multiple tree predictions to improve accuracy.
Complexity	Simple and easy to understand and interpret.	More complex due to the ensemble of multiple trees. Less interpretable than a single decision tree.
Tendency to Overfit	High, especially if the tree is deep with many branches.	Lower, as averaging or majority voting across trees reduces overfitting.
Bias-Variance Tradeoff	Can have either high bias or high variance, depending on the depth of the tree.	Generally achieves a good balance by reducing variance without significantly increasing bias.
Handling of Variance	Prone to high variance without pruning or limiting tree depth.	Naturally handles variance well by averaging out the biases and errors of individual trees.
Performance on Large Datasets	Can struggle with very large datasets due to computational inefficiencies.	Excels with large datasets, as the complexity of individual trees is not a limiting factor.
Predictive Power	Good, but can be limited by overfitting or underfitting.	Generally higher due to aggregation reducing overfitting and leveraging the strength of multiple learners.
Training Speed	Fast, as it involves building a single tree.	Slower than decision trees due to the need to train multiple trees. However, it can be parallelized.
Feature Importance	Provides insights on feature importance based on the splits in the tree.	Aggregates feature importance across all trees, offering a more robust insight into feature relevance.
Handling of Missing Values	Requires imputation or special handling of missing values.	Robust to missing values, as different trees in the forest can learn from different subsets of the data.
Versatility	Can be used for both classification and regression tasks.	Highly versatile, can be used for classification, regression, and even for unsupervised tasks like clustering.

- https://mlu-explain.github.io/decision-tree/

- https://mlu-explain.github.io/random-forest/

- See Lesson 4.pynb
- • Code Cell
  - `4.1 Dicision Tree with Gini Impurity`
- Open Rapid Miner