INTRODUCTION TO DECISION TREES

ABHINAV PANDEY

MY PROFILE

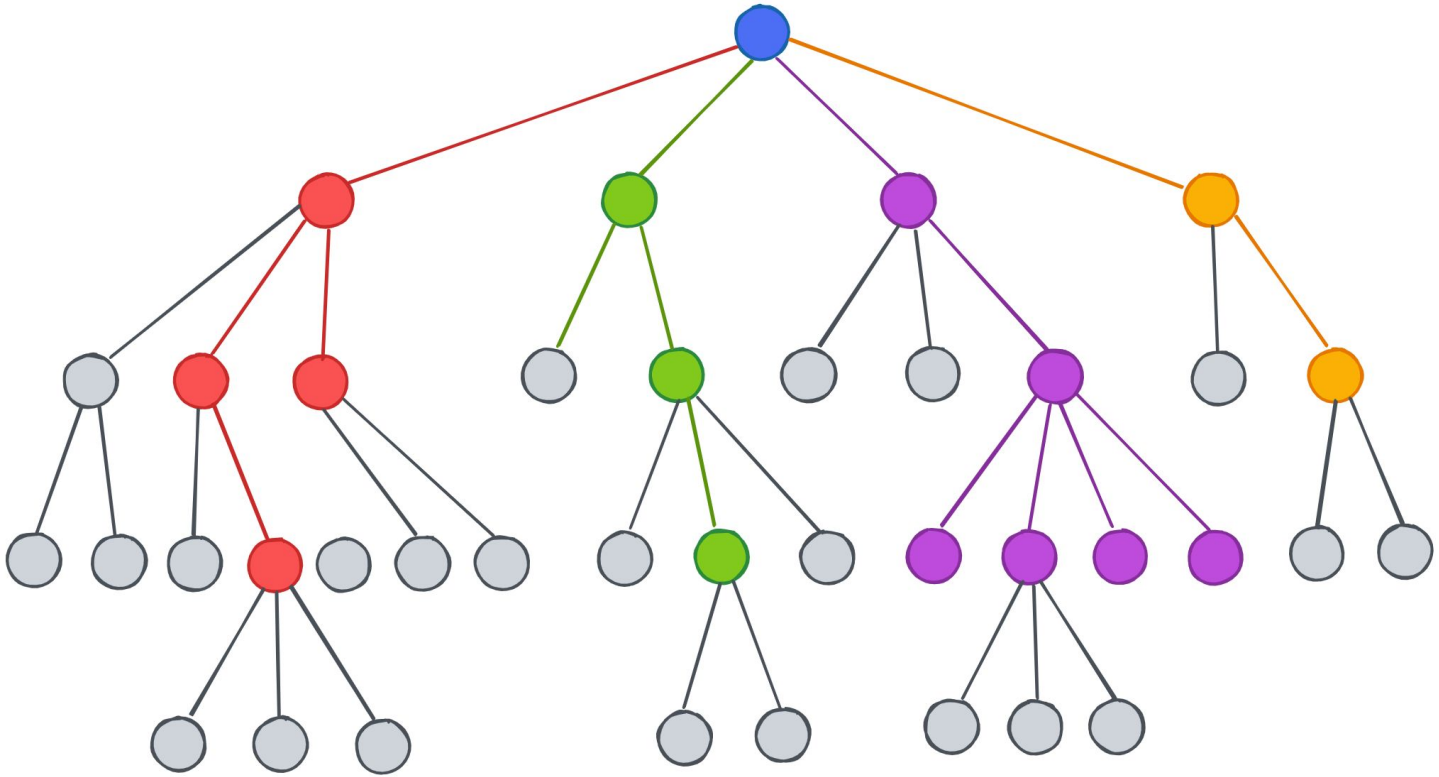
Abhinav Pandey

Data Scientist | Gen AI | LLMs | Author | Speaker | Ex-JP Morgan | Cricketer | Cyclist

AGENDA

1. INTRODUCTION TO DECISION TREES
2. HOW DECISION TREES WORK
3. BUILDING A DECISION TREE
4. PRUNING DECISION TREES
5. ENSEMBLE METHODS WITH DECISION TREES
6. HYPERPARAMETER TUNING
7. ADVANTAGES & DISADVANTAGES OF DECISION TREES
8. REAL-LIFE EXAMPLES & APPLICATIONS
9. DECISION TREE IMPLEMENTATION

INTRODUCTION TO DECISION TREES



WHAT ARE DECISION TREES

A decision tree is a supervised machine learning algorithm that mimics the human decision-making process. It's a tree-like structure where each internal node represents a test on an attribute, each branch represents a possible outcome, and each leaf node represents a decision or prediction.

Think of it as a flowchart where you ask a series of questions to narrow down the possibilities. For example, to identify an animal, you might first ask if it has fur, then if it's carnivorous, and so on.

ADVANTAGES OF DECISION TREES

1. **Interpretability**: Decision trees are easy to understand and visualize, making them a popular choice for explaining models to non-technical stakeholders.

2. **Non-parametric**: They don't make assumptions about the underlying data distribution, making them robust to outliers.

3. **Handle both categorical and numerical data**: Decision trees can handle mixed data types.

4. **Efficient**: They can be built relatively quickly, especially for smaller datasets.

TYPES OF DECISION TREES

1. **Classification Trees**: Used for predicting categorical outcomes (e.g., spam or not spam, customer churn or not churn).

2. **Regression Trees**: Used for predicting numerical outcomes (e.g., house price, sales revenue).

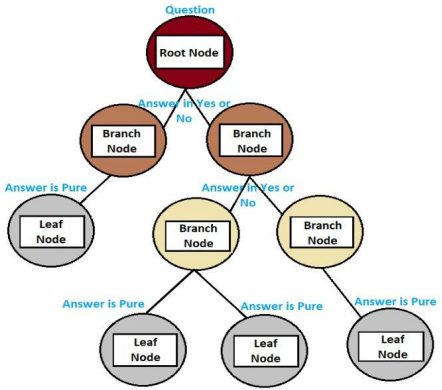
*Note: While decision trees are intuitive, they can be prone to overfitting, especially with complex decision trees. Techniques like pruning and ensemble methods (e.g., random forests) can help mitigate this issue.

HOW DECISION TREES WORK

TREE STRUCTURE & TERMINOLOGY

A decision tree consists of:

1. **Root Node**: The starting point of the tree.
2. **Internal Nodes**: Nodes that represent tests on attributes.
3. **Leaf Nodes**: Nodes that represent the final decisions or predictions.



DECISION RULES AND FEATURE SPLITS

At each internal node, a decision rule is applied to the data. This rule is based on a feature (attribute). The data is then split into subsets based on the possible outcomes of the test. This process is known as feature splitting.

RECURSIVE PARTITIONING OF DATA

Decision trees are constructed recursively. Starting from the root node, the data is split into subsets based on the best feature and its corresponding values. This process is repeated for each subset until a stopping criterion is met (e.g., all data points in a subset belong to the same class).

ENTROPY AND INFORMATION GAIN

To determine the best feature for splitting, decision trees often use entropy and information gain.

1. **Entropy**: A measure of the impurity or randomness in a dataset. A dataset with equal proportions of different classes has maximum entropy. It is used to determine the best attribute to split the data at each node. A lower entropy indicates a more homogeneous dataset.

2. **Information Gain**: It is a metric used in decision trees to measure the amount of information that an attribute provides about the target variable.The reduction in entropy achieved by splitting the data on a particular feature. The feature with the highest information gain is typically chosen for the split.

The goal of decision trees is to create a tree structure that minimizes the entropy of the leaf nodes, effectively partitioning the data into pure subsets.

ENTROPY WITH AN EXAMPLE

Consider the following dataset of 10 customers, with two attributes: Age and Credit Rating:

Age	Credit Rating
25	Good
30	Bad
35	Good
40	Bad
45	Good
50	Bad
55	Good
60	Bad
65	Good
70	Bad

ENTROPY WITH AN EXAMPLE

If we use the attribute Age to split the data, we would get the following two subsets:

- **Age <= 45**:
  - Good: 5
  - Bad: 3
- **Age > 45**:
  - Good: 4
  - Bad: 5

The entropy of the first subset is: entropy = - (5/8) * log2(5/8) - (3/8) * log2(3/8) = **0.954**

The entropy of the second subset is: entropy = - (4/9) * log2(4/9) - (5/9) * log2(5/9) = **0.991**

The weighted average of the entropy of the two subsets is: weighted_entropy = (8/10) * 0.954 + (2/10) * 0.991 = **0.963**

ENTROPY WITH AN EXAMPLE

If we use the attribute Credit Rating to split the data, we would get the following two subsets:

- **Credit Rating = Good**:
  - Age <= 45: 4
  - Age > 45: 3
- **Credit Rating = Bad**:
  - Age <= 45: 2
  - Age > 45: 4

The entropy of the first subset is: entropy = - (4/7) * log2(4/7) - (3/7) * log2(3/7) = **0.985**

The entropy of the second subset is: entropy = - (2/6) * log2(2/6) - (4/6) * log2(4/6) = **0.918**

The weighted average of the entropy of the two subsets is: weighted_entropy = (7/10) * 0.985 + (3/10) * 0.918 = **0.969**

ENTROPY WITH EXAMPLE

The attribute with the lowest weighted entropy is the best attribute to split the data at the root node. In this case, **Age** is the best attribute.

By calculating the entropy of each attribute at each node, decision trees can find the optimal way to split the data and make accurate predictions.

INFORMATION GAIN WITH AN EXAMPLE

Formula: Information Gain(A) = Entropy(Parent) - ∑((|Si|/|S|) * Entropy(Si))

where:

A: Attribute

S: Parent node

Si: Child node i

|Si|: Number of instances in child node i

|S|: Number of instances in parent node

INFORMATION GAIN WITH AN EXAMPLE

Interpretation:

- **Higher Information Gain:** An attribute with a higher Information Gain provides more information about the target variable and is a better choice for splitting the data.
- **Lower Information Gain:** An attribute with a lower Information Gain provides less information about the target variable and is a less suitable choice for splitting the data.

Let's use the same dataset from the previous example.

INFORMATION GAIN WITH AN EXAMPLE

We already calculated the weighted entropy for both Age and Credit Rating:

- Weighted entropy for Age: **0.963**
- Weighted entropy for Credit Rating: **0.969**

The entropy of the parent node (containing all 10 instances) is:

Entropy(Parent) = - (5/10) * log2(5/10) - (5/10) * log2(5/10) = **1**

Now, we can calculate the Information Gain for both attributes:

- **Information Gain for Age:** 1 - 0.963 = **0.037**
- **Information Gain for Credit Rating:** 1 - 0.969 = **0.031**

Conclusion: In this example, **Age** has a slightly higher Information Gain than Credit Rating, suggesting that Age provides slightly more information about the target variable (Credit Rating). Therefore, **Age** would be a better choice for splitting the data at the root node.

BUILDING A DECISION TREE

ALGORITHMS FOR CONSTRUCTING DECISION TREES

Several algorithms can be used to build decision trees:

1. **ID3 (Iterative Dichotomiser 3)**: A top-down greedy algorithm that uses information gain as the splitting criterion.

2. **C4.5**: An extension of ID3 that handles missing values and continuous attributes. It also uses gain ratio as the splitting criterion to address the bias of information gain towards features with many possible values.

3. **CART (Classification and Regression Trees)**: A popular algorithm that can handle both classification and regression tasks. It uses the Gini index as the splitting criterion.

SPLITTING CRITERIA

1. **Gini Index**: A measure of impurity based on the probability of randomly selecting two different samples from the same node and having them belong to different classes. A Gini index of 0 indicates pure nodes.

2. **Information Gain**:As discussed earlier, the reduction in entropy achieved by splitting the data on a particular feature.

3. **Gain Ratio**: The information gain normalized by the intrinsic information of the split, which helps to address the bias of information gain towards features with many possible values.

GINI INDEX WITH AN AN EXAMPLE

Gini Index is another metric used in decision trees to measure the impurity of a dataset. It is a measure of how evenly distributed the classes are within a node. A Gini Index of 0 indicates that all instances belong to the same class, while a Gini Index of 0.5 indicates a perfectly balanced distribution of classes.

Formula:

Gini Index = 1 - ∑(pi^2)

where:

- pi: Proportion of instances belonging to class i

GINI INDEX WITH AN EXAMPLE

Let's use the same dataset from the previous examples:

For the parent node:

- Gini Index = 1 - (5/10)^2 - (5/10)^2 = **0.5**

For the child nodes after splitting on Age:

- Left child (Age <= 45): Gini Index = 1 - (5/8)^2 - (3/8)^2 = **0.375**
- Right child (Age > 45): Gini Index = 1 - (4/9)^2 - (5/9)^2 = **0.494**

Weighted Gini Index:

- Weighted Gini Index = (8/10) * 0.375 + (2/10) * 0.494 = **0.409**

GINI INDEX WITH AN EXAMPLE

For the child nodes after splitting on Credit Rating:

- Left child (Credit Rating = Good): Gini Index = 1 - (4/7)^2 - (3/7)^2 = **0.476**
- Right child (Credit Rating = Bad): Gini Index = 1 - (2/6)^2 - (4/6)^2 = **0.444**

Weighted Gini Index:

- Weighted Gini Index = (7/10) * 0.476 + (3/10) * 0.444 = **0.464**

GINI INDEX WITH AN EXAMPLE

Comparison:

- The weighted Gini Index for **Age** (0.409) is lower than the weighted Gini Index for **Credit Rating** (0.464).
- This indicates that splitting on Age results in a more homogeneous distribution of classes in the child nodes.

Conclusion: Similar to Information Gain, the Gini Index is used to select the best attribute for splitting the data. A lower Gini Index suggests a more homogeneous distribution of classes in the child nodes, which is generally desirable in decision trees.

In summary, both Information Gain and Gini Index are effective metrics for selecting attributes in decision trees. The choice between these metrics often depends on the specific problem and dataset.

GAIN RATIO WITH AN EXAMPLE

Gain Ratio is a modification of Information Gain that takes into account the number of branches created by a split. This is useful because Information Gain can be biased towards attributes with many possible values.

Formula:

Gain Ratio(A) = Information Gain(A) / Split Information(A)

where:

- Split Information(A) = - ∑((|Si|/|S|) * log2(|Si|/|S|))

GAIN RATIO WITH AN EXAMPLE

Interpretation:

- **Higher Gain Ratio:** An attribute with a higher Gain Ratio provides more information about the target variable relative to the number of branches it creates.
- **Lower Gain Ratio:** An attribute with a lower Gain Ratio provides less information about the target variable relative to the number of branches it creates.

Let's use the same dataset from the previous examples:

GAIN RATIO WITH AN EXAMPLE

We already calculated the Information Gain for Age and Credit Rating:

- Information Gain for Age: **0.037**
- Information Gain for Credit Rating: **0.031**

Now, we need to calculate the Split Information for both attributes:

- Split Information for Age: - (8/10) * log2(8/10) - (2/10) * log2(2/10) = **0.721**
- Split Information for Credit Rating: - (7/10) * log2(7/10) - (3/10) * log2(3/10) = **0.886**

Finally, we can calculate the Gain Ratio for both attributes:

- Gain Ratio for Age: 0.037 / 0.721 = **0.051**
- Gain Ratio for Credit Rating: 0.031 / 0.886 = **0.035**

GAIN RATIO WITH AN EXAMPLE

Conclusion: In this example, **Age** has a higher Gain Ratio than Credit Rating, suggesting that Age provides more information about the target variable relative to the number of branches it creates. Therefore, Age would be a better choice for splitting the data at the root node.

In summary, Gain Ratio is a useful metric for selecting attributes in decision trees, especially when dealing with attributes with many possible values. By considering both the information gain and the number of branches, Gain Ratio can help prevent bias towards attributes with many levels.

STOPPING CRITERIA

To prevent overfitting, decision trees typically use stopping criteria:

1. **Maximum Depth**: The maximum number of levels allowed in the tree.
2. **Minimum Samples Split**: The minimum number of samples required to split an internal node.
3. **Minimum Samples Leaf**: The minimum number of samples required to form a leaf node.

By setting appropriate stopping criteria, decision trees can be made more generalizable and less prone to overfitting.

PRUNING DECISION TREES

OVERFITTING & UNDERFITTING

1. **Overfitting**: When a model is too complex and fits the training data too closely, leading to poor performance on new, unseen data.

2. **Underfitting**: When a model is too simple and cannot capture the underlying patterns in the data, also leading to poor performance.

Pruning is a technique that involves chopping off parts of a tree once it is fully grown. It is a bottom-up approach used to prevent overfitting in decision trees.

PRE-PRUNING AND POST PRUNING TECHNIQUES

1. **Pre-pruning**: Stopping the tree construction process early, based on predefined criteria (e.g., maximum depth, minimum samples per leaf).

2. **Post-pruning**: Building the full tree and then removing branches that do not significantly improve performance.

COST COMPLEXITY PRUNING

A popular post-pruning technique that involves:

1. Building the full decision tree.
2. Calculating the cost of each node, which is a weighted sum of its training error and the number of leaves below it.
3. Iteratively pruning the subtree that results in the smallest increase in the overall cost.
4. Stopping when pruning further leads to a significant increase in cost.

Cost complexity pruning helps to find the optimal balance between a tree's complexity and its predictive accuracy.

ENSEMBLE METHODS WITH DECISION TREES

BAGGING (BOOTSTRAP AGGREGATION)

1. **Idea**: Create multiple decision trees by sampling the training data with replacement (bootstrapping).
2. **Aggregation**: Combine the predictions of all trees, typically by majority voting for classification or averaging for regression.
3. **Benefits**: Reduces variance and helps prevent overfitting.

RANDOM FORESTS

1. **Extension of Bagging**: Randomly select a subset of features at each node during tree construction.
2. **Benefits**: Further reduces correlation between trees, improving generalization performance.

BOOSTING

1. **Idea**: Build trees sequentially, focusing on the examples that were misclassified by previous trees.
2. **Types**:
    - **AdaBoost (Adaptive Boosting)**: Assigns weights to examples based on their classification accuracy.
    - **Gradient Boosting**: Fits each tree to the residuals of the previous trees, minimizing a loss function.
3. **Benefits**: Can achieve high accuracy, especially for complex tasks.

HYPERPARAMETER TUNING

Hyperparameters are parameters that are not learned from the data but are set before training. Tuning hyperparameters is crucial for optimizing the performance of a decision tree model.

IMPORTANT HYPERPARAMETERS

max_depth: The maximum depth of the tree. A deeper tree can capture more complex patterns but is more prone to overfitting.

min_samples_split: The minimum number of samples required to split an internal node. A larger value can prevent overfitting but may result in underfitting.

min_samples_leaf: The minimum number of samples required to form a leaf node. A larger value can prevent overfitting but may result in underfitting.

max_features: The maximum number of features to consider at each split. A smaller value can reduce computational cost but may also reduce accuracy.

GRID SEARCH AND RANDOM SEARCH FOR HYPERPARAMETER OPTIMIZATION

1. **Grid Search**: Exhaustively searches over a predefined grid of hyperparameter values.
2. **Random Search**: Randomly samples hyperparameter values from a specified distribution.
3. **Advantages of Random Search**: Often more efficient than grid search for large hyperparameter spaces.

BIAS-VARIANCE TRADEOFF

1. **Bias**: The error due to the model's inability to capture the true underlying relationship. A complex model typically has lower bias but higher variance.

2. **Variance**: The error due to the model's sensitivity to small changes in the training data. A simple model typically has higher bias but lower variance.

Hyperparameter tuning aims to find the optimal balance between bias and variance. A well-tuned decision tree model can achieve high accuracy without overfitting.

ADVANTAGES AND DISADVANTAGES OF DECISION TREES

ADVANTAGES

1. **Interpretability and Feature Importance**: Decision trees are easy to understand and visualize, making them a popular choice for explaining models to non-technical stakeholders. The importance of each feature can be assessed based on its position in the tree and the number of times it is used for splitting.

2. **Handling Missing Data and Categorical Features**: Decision trees can naturally handle missing data by creating branches for missing values. They can also handle both categorical and numerical features without requiring preprocessing.

DISADVANTAGES

1. **Instability and Sensitivity to Small Changes in Data**: Decision trees can be sensitive to small changes in the data, leading to different tree structures and predictions. This can make them less robust to noise and outliers.

2. **Bias Towards Features with Many Levels**: Decision trees tend to favor features with many levels, as they have more potential to split the data. This can lead to biased models if features with fewer levels are important.

Despite these limitations, decision trees remain a valuable tool in machine learning, especially when interpretability is a priority. Techniques like ensemble methods and careful hyperparameter tuning can help mitigate their drawbacks.

REAL-LIFE EXAMPLES AND APPLICATIONS OF DECISION TREES

CREDIT RISK ANALYSIS

1. **Goal**: Predict the likelihood of a borrower defaulting on a loan.

2. **Approach**: Use decision trees to analyze factors like income, credit history, and debt-to-income ratio to determine creditworthiness.

CHURN PREDICTION

1. **Goal**: Predict whether a customer is likely to stop using a product or service.

2. **Approach**: Analyze customer behavior, demographics, and other relevant factors to identify patterns associated with churn.

COMPUTER VISION AND IMAGE RECOGNITION

1. **Goal**: Classify images into different categories.

2. **Approach**: Decision trees can be used to create hierarchical classifiers that break down the image classification task into smaller, more manageable subtasks.

FRAUD DETECTION

1. **Goal**: Identify fraudulent transactions or activities.

2. **Approach**: Analyze patterns in transaction data, user behavior, and other relevant factors to detect anomalies that may indicate fraudulent activity.

MEDICAL DIAGNOSIS

1. **Goal**: Assist in diagnosing diseases based on patient symptoms and test results.

2. **Approach**: Decision trees can be used to create diagnostic models that guide healthcare professionals in making informed decisions.

DECISION TREE IMPLEMENTATION

EXAMPLES

1. **BASIC DECISION TREE**
2. **CART**

MY PROFILE

Abhinav Pandey

Data Scientist | Gen AI | LLMs | Author | Speaker | Ex-JP Morgan | Cricketer | Cyclist

THANK YOU