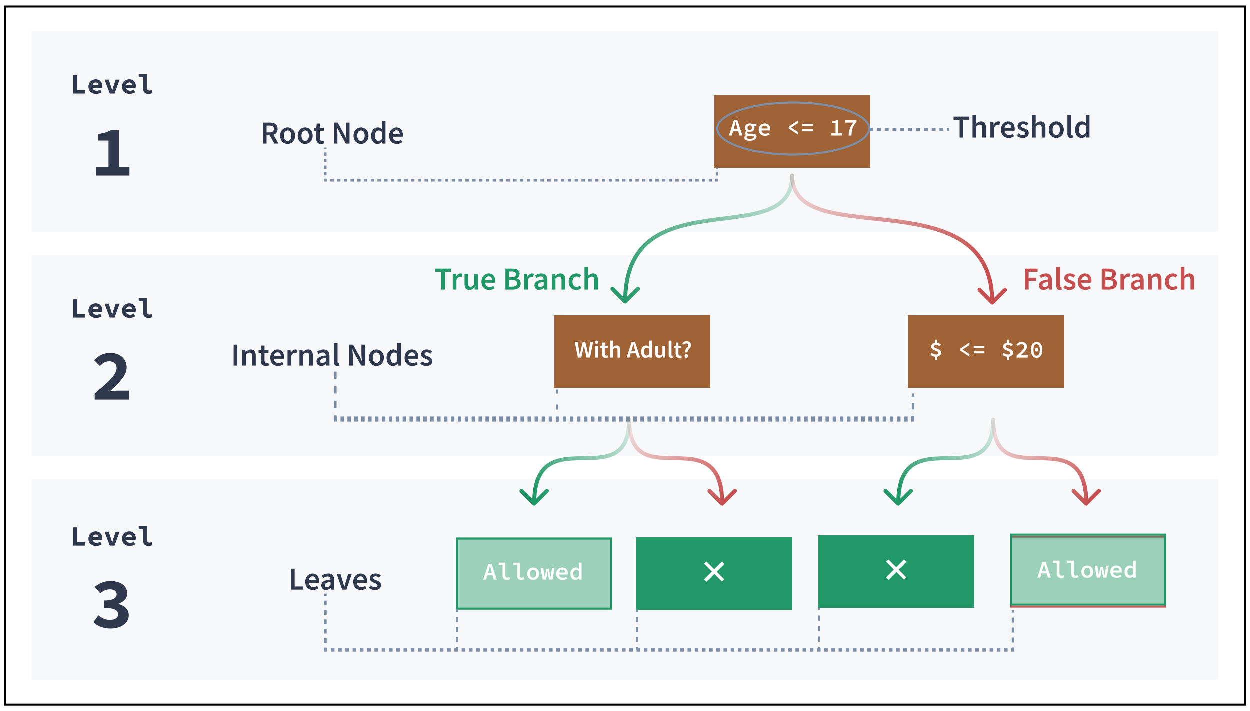
Foundations of Decision Trees: Takeaways

CONCEPTS

- Decision trees are supervised algorithms.
- There are different algorithms to create decision trees (ID3, C4.5, C5.0 and CART). The `scikit-learn` library uses the **classification and regression trees (CART)** algorithm.
- The ultimate goal of decision trees is to split our datasets into homogeneous groups of observations (for example, observations that belong to the same target column class).
- Decision trees work by recursively splitting our datasets based on binary conditions, called **thresholds**.
- By using `scikit-learn`, it's possible to visualize these splits using diagrams. As a result, decision trees are relatively easier to understand compared to other machine learning models.
- Decision trees are comprised of the following elements:



- **Nodes**: contain the thresholds. We have different types of nodes:
  - **Root Node**: is the node at the top, where the tree begins (level 1). It has the most important and predictive threshold from the dataset.
  - **Internal Nodes**: start at level 2, and if the decision tree has different levels, they will continue to appear until the second last level. Other than that, they are exactly the same as the root node, in the sense that they feature additional thresholds.
  - **Leaves** (also called **terminal nodes**): make a prediction. In other words, if we use a decision tree to solve a categorical or regression problem, we need to take the observation

    and filter it through the different thresholds until we reach a leaf. Leaves will always be present at the bottom level, but in some cases, they can also appear on upper levels along with internal nodes. If a leaf only features homogeneous data, it will be considered a **pure leaf**; otherwise, it's an **impure leaf**.

- Additional classification for nodes: **parent nodes** are the nodes that get divided into two subsequent nodes, which are the **child nodes**. Therefore:
    - The root node is exclusively a parent node, as it doesn't originate from any other node.
    - The internal nodes are both parent and child Nodes. They can be parents of either two nodes, two leaves, or both a node and a leaf.
    - The leaves are always exclusively child nodes.
- **Branches**: the arrows that connect a parent node with its two child nodes. They follow a universal rule: based on the filtering from the threshold, `True` observations will always follow the left arrow, and `False` observations will continue through the right one. So, **True = Left; False = Right**.
- All these elements are required to create a decision tree. A tree simply cannot exists if any of the elements are missing.
- If we have a binary column with two possible classifications, we would need to assign one of them to the value "0" and the other one to be "1".
- In order to be computationally efficient, decision trees sort the observations in all the feature columns and then get the middle value between every consecutive pair of observations. This operation then generates a list with a specific number of thresholds, and every single one will be tried to discover the best split.
- For every split, absolutely all the thresholds from all feature columns must be tested every time.
- In order to select the best threshold from the list, depending on the type of tree we're using, we can use different criteria:
    - For **regression trees**, we can use **mean squared error (MSE)** and **mean absolute error (MAE)**.
    - For **classification trees**, we can use **Gini impurity** (or simply **Gini**) and **entropy and information gain**.
- While classification trees aim to generate leaves for every class present in the target column, regression trees create a general metric from all the unique observations from the target column, which is then used to generate a standardized value on the different leaves.

- The **mean squared error (MSE)** gets the **mean** from the target column, and then computes the **error** for each observation, which is the square difference between the value of the observation and the mean. Then it divides the sum of all those **errors** by the number of observations.
- Since it's referring to errors, the threshold with the lowest MSE will then be the optimal one.
- The formula for calculating the MSE is as follows:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Where:

- n is the number of observations
- x_i are the individual values in the target column
- $\bar{x}$ is the mean value of the target column

- We can also express the MSE formula as follows:

$$\text{MSE} = \frac{(\text{observation}_1 - \text{mean})^2 + (\text{observation}_2 - \text{mean})^2 + \ldots + (\text{observation}_n - \text{mean})^2}{\text{total number of observations}}$$

- The **mean absolute error (MAE)** is pretty similar to MSE, but instead of using the mean, it gets the absolute difference between the value of the observation and the **median**. Absolute means that the resulting number will be forced to be positive. Then it divides the sum of all those **errors** by the number of observations.
- Since it's referring to errors, the threshold with the lowest MAE will then be the optimal one.
- The formula for calculating the MAE is as follows:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |x_i - \tilde{x}|$$

Where:

- n is the number of observations
- x_i are the individual values in the target column
- $\tilde{x}$ is the median value of the target column

- We can also express the MAE formula as follows:

$$\text{MAE} = \frac{|\text{observation}_1 - \text{median}| + |\text{observation}_2 - \text{median}| + \ldots + |\text{observation}_n - \text{median}|}{\text{total number of observations}}$$

- **Gini Impurity** tells us the probability of misclassifying the label in the target column of a certain observation if we randomly select that observation from the dataset. It varies from 0 (optimal) to values close to 1 (but it never reaches 1, as there would need to be an infinite number of observations). Therefore, the threshold with the lowest Gini will be the optimal one.
- The formula for calculating Gini impurity is as follows:

$$\text{Gini} = 1 - \sum_{i=1}^{k} (p_i)^2$$

Where:

- k is the number of distinct classes in the target column
- p_i is the probability an observation belongs to class i

- We can also express the Gini impurity formula as follows:

$$\text{Gini} = 1 - \left[\left(\frac{\text{number of observations of class } 1}{\text{total number of observations}}\right)^2 + \ldots + \left(\frac{\text{number of observations of class } k}{\text{total number of observations}}\right)^2\right]$$

- When we want to determine the optimal threshold for a split, we need to apply weights for every `True` and `False` subset, like this:

$$\text{Weighted Gini} = \left(\frac{\text{true\_split.shape}[0]}{\text{parent.shape}[0]} \times \text{Gini}_{\text{true child}}\right) + \left(\frac{\text{false\_split.shape}[0]}{\text{parent.shape}[0]} \times \text{Gini}_{\text{false child}}\right)$$

Notice that $\frac{\text{child\_split.shape}[0]}{\text{parent.shape}[0]}$ is how we weight the Gini impurities due to every split potentially having different numbers of observations.

- **Entropy** is a term from information theory that tells us about the level of disorder/ unpredictability in a target column; in other words, it's about how heterogeneous the column is. Depending on the number of classes in the column, entropy can vary from a maximum value of 1 for binary values, 2 for a column with four different classes, etc. If our column is completely homogeneous, then the entropy will be 0, and since that situation is where decision trees aim to be, they will split the data to reduce entropy accordingly.
- The formula for calculating the Entropy is as follows:

$$\text{Entropy} = -\sum_{i=1}^{k}(p_i) \times \log_2(p_i)$$

$$\text{Entropy} = -\left[\frac{\text{number of observations of class 1}}{\text{total number of observations}} \times \log_2\left(\frac{\text{number of observations of class 1}}{\text{total number of observations}}\right) + \ldots + \frac{\text{number of observa}}{\text{total number of}}\right.$$

- However, instead of entropy, decision trees use **information gain** to decide on the best split. It is the difference between the entropy of the parent node and the entropy of the child node. The higher the information gain, the better. Note that when we want to apply information gain to determine the optimal threshold for a split, we need to apply weights to the `True` and `False` splits.
- The formula for calculating information gain is as follows:

$$\text{IG} = \text{Entropy}_{\text{parent}} - \left(\frac{\text{true\_split.shape}[0]}{\text{parent.shape}[0]} \times \text{Entropy}_{\text{true child}} + \frac{\text{false\_split.shape}[0]}{\text{parent.shape}[0]} \times \text{Entropy}_{\text{false child}}\right)$$

Remember that $\frac{\text{child\_split.shape}[0]}{\text{parent.shape}[0]}$ is how we weight the entropies due to every split potentially having different numbers of observations.

RESOURCES

- Decision Tree learning
- Decision Tree algorithms
- Mean Squared Error
- Mean Absolute Error
- Gini Impurity
- Entropy
- Logarithms
- Information Gain in Decision Trees