**Interview Questions:**

1. What are some common hyperparameters of decision tree models, and how do they affect the model's performance?

Common hyperparameters of decision tree models significantly influence their performance. Here are the key hyperparameters and their effects:

Common Hyperparameters

1. **Criterion**:
   * **Description**: This parameter determines the function used to measure the quality of a split. Common options include:
     + **Gini impurity**: Measures the impurity of a node based on the probability of misclassifying a randomly chosen element.
     + **Entropy (Information Gain)**: Measures the information gain from a split, focusing on reducing uncertainty.
   * **Effect on Performance**: Different criteria can lead to different tree structures and performance; choosing the right one can improve accuracy.
2. **Max Depth**:
   * **Description**: This limits the maximum depth of the tree.
   * **Effect on Performance**: A deeper tree can capture more complex patterns but may lead to overfitting. Limiting depth helps improve generalization on unseen data.
3. **Min Samples Split**:
   * **Description**: The minimum number of samples required to split an internal node.
   * **Effect on Performance**: Higher values prevent splits that may not be statistically significant, reducing overfitting.
4. **Min Samples Leaf**:
   * **Description**: The minimum number of samples that must be present in a leaf node.
   * **Effect on Performance**: Setting this parameter helps ensure that leaf nodes have enough samples to make reliable predictions, which can reduce overfitting.
5. **Max Features**:
   * **Description**: The number of features to consider when looking for the best split.
   * **Effect on Performance**: Limiting features can help reduce overfitting and improve model generalization by introducing randomness into the model.
6. **Min Weight Fraction Leaf**:
   * **Description**: The minimum weighted fraction of the sum total of weights (for weighted classification) required to be at a leaf node.
   * **Effect on Performance**: This is particularly useful for addressing class imbalance, ensuring that minority classes are adequately represented in leaf nodes.
7. **Class Weight**:
   * **Description**: This parameter allows you to assign different weights to classes to handle imbalanced datasets.
   * **Effect on Performance**: Properly adjusting class weights can help improve model performance on underrepresented classes.

Importance of Hyperparameter Tuning

* **Improved Performance**: Untuned hyperparameters may lead to suboptimal models. Tuning helps find settings that better capture underlying data patterns.
* **Reduced Overfitting**: Decision trees are prone to overfitting; tuning parameters like max\_depth and min\_samples\_split controls complexity and prevents memorization of noise in training data.
* **Enhanced Generalization**: A well-tuned tree strikes a balance between complexity and flexibility, improving performance on unseen data.
* **Tailoring to Specific Tasks**: Different tasks may require different behaviors from decision trees, and hyperparameter tuning allows customization for specific needs.

2. What is the difference between the Label encoding and One-hot encoding?

Label encoding and one-hot encoding are two common techniques for converting categorical variables into numerical formats that machine learning algorithms can interpret.

Label Encoding

* **Definition**: Label encoding assigns a unique integer to each category in a categorical variable. For example, if you have a feature "Color" with categories "Red," "Green," and "Blue," label encoding might assign them as follows:
  + Red = 0
  + Green = 1
  + Blue = 2
* **Characteristics**:
  + **Single Column**: It converts categorical values into a single column of integers.
  + **Ordinal Relationships**: It implies an ordinal relationship among categories, which may not be appropriate for nominal data (categories without a natural order).
  + **Computational Efficiency**: Uses less memory and is computationally lighter since it only adds one column.
* **Use Cases**:
  + Suitable for ordinal data where the order matters (e.g., "Low," "Medium," "High").
  + Can introduce bias in models that assume a ranking among categories when none exists.

One-Hot Encoding

* **Definition**: One-hot encoding creates binary columns for each category in a categorical variable. Each category is represented as a separate column, with a value of 1 indicating the presence of that category and 0 indicating absence. For the same "Color" feature, one-hot encoding would create three columns:
  + Red: [1, 0, 0]
  + Green: [0, 1, 0]
  + Blue: [0, 0, 1]
* **Characteristics**:
  + **Multiple Columns**: It generates multiple columns based on the number of unique categories.
  + **No Ordinal Relationships**: It avoids implying any order among categories, making it suitable for nominal data.
  + **Higher Dimensionality**: Increases the dimensionality of the dataset, which can lead to higher computational costs and potential issues with the curse of dimensionality.
* **Use Cases**:
  + Preferred for nominal data where no inherent order exists (e.g., colors, types).
  + Useful in algorithms that do not handle ordinal relationships well (e.g., linear regression).

Key Differences

| **Feature** | **Label Encoding** | **One-Hot Encoding** |
| --- | --- | --- |
| **Output Format** | Single column with integer values | Multiple binary columns |
| **Relationship Implied** | Implies ordinal relationship | No ordinal relationship |
| **Memory Usage** | More efficient (less memory) | Less efficient (more memory) |
| **Dimensionality** | Does not increase dimensionality | Increases dimensionality significantly |
| **Suitable For** | Ordinal data | Nominal data |