Decision Trees

Saturday, July 27, 2024 2:02 PM

What is a decision tree?

- A tree like model that splits that data into subsets based on features
- A supervised learning algorithm that can handle both regression and classification tasks

How does a decision tree work?

- The tree starts with a root node
    - The root node represents the entire dataset
- The algorithm chooses the best feature it should use to split the data into subsets
    - Typically done using a measure such as Gini impurity or entropy
- The data is then split into subsets based on the selected feature and threshold value
- Two child nodes are created
    - One for each subset of data
- The feature selection, splitting, and child nodes are created until a stopping criterion has been met
    - Ex all instances in a node belong to the same class

Key Concepts?

- Entropy
    - A measure of the uncertainty of randomness in the data
        - A lower entropy indicates a better split
- Gini impurity
    - A measure of how well a feature splits the data
        - A lower Gini Impurity indicates a better split
- Information Gain
    - The difference in entropy before and after splitting the data
        - A higher information gain means a better split
- Overfitting
    - When a decision tree is too complex and fits the noise in the training data rather than the underlying patterns

What is a class in the context of decision trees and machine learning?

- A class refers to a label or category that an instance (data point) should belong to
- When instances belong to the same class, it means the decision tree has successfully separated the data into distinct groups
- This is desirable because
    - When instances are grouped by class, the model can make more accurate predictions for new, unseen data
    - When instances are evenly distributed among classes, the model is less confident in its prediction, and will struggle with new, unseen data

Classification Problem Classes:

1. Win/Loss: Will the home team win (Class 1) or lose (Class 0) the game?

2. Cover/Not Cover: Will the favorite team cover the point spread (Class 1) or not cover (Class 0)?
3. Over/Under: Will the total points scored in the game be over (Class 1) or under (Class 0) the predicted total?
4. Player Prop: Will a specific player (e.g., quarterback) throw for over (Class 1) or under (Class 0) a certain number of yards?

Regression Problem Classes:

1. Point Spread Margin: Predict the exact margin of victory for the favorite team (e.g., 7.5 points).
2. Total Points Scored: Predict the exact total points scored in the game (e.g., 45.5 points).
3. Player Performance: Predict the exact number of yards a player will throw for (e.g., 275.5 yards) or rush for (e.g., 92.5 yards).
4. Game Outcome Probability: Predict the probability of a specific game outcome (e.g., 75% chance of the home team winning).

Decision Tree Example

Let's consider a simple example to illustrate how a Decision Tree works. Suppose we want to predict whether a player will score a touchdown based on two features: `yards_gained` and `red_zone_attempts`.

yards_gained	red_zone_attempts	touchdown
10	2	1
20	1	0
30	3	1
40	2	1
50	1	0

The Decision Tree might look like this:

- Root Node: `yards_gained` < 30
    - Left Child Node: `red_zone_attempts` < 2
        - Leaf Node: 0 (no touchdown)
    - Right Child Node: `red_zone_attempts` ≥ 2
        - Leaf Node: 1 (touchdown)
- Right Child Node: `yards_gained` ≥ 30
    - Leaf Node: 1 (touchdown)

How to tell if your model is overfitting?

- Training and test scores
    - By running the model on both training and test data, the results can be compared
        - If there is a significant difference between training data output and test data output,

this could be an indicator of overfitting

- Validation Curves
    - Plot validation curves to help visualize the models performance on the training and validation datasets over different hyperparameters
        - This can help you determine if model is over or underfitting
- Learning Curves
    - Show the models performance on the training and validation datasets over different sample sizes
        - This can help you determine if model is over or underfitting

Tools and Libraries

Popular libraries like scikit-learn, TensorFlow, and PyTorch provide tools and functions to help you evaluate and visualize your model's performance. For example, scikit-learn's GridSearchCV and RandomizedSearchCV can help you perform hyperparameter tuning and provide insights into your model's performance.

Gini Impurity

- A measure of the probability of of misclassifying a randomly chosen instance from a dataset
- Calculates the sum of squared probabilities of each class in the dataset
- Ranges from 0 to 1.
    - 0 being a pure node, all instances belong to the same class
    - 1 being an impure node, instances are evenly distributed among classes

Entropy

- A measure of the probability of the uncertainty of randomness in the dataset
- Calculated as the sum of the probabilities of each class in the dataset multiplied by the logarithm of those probabilities.
- Ranges from 0 to 1.
    - 0 being a pure node, all instances belong to the same class
    - 1 being an impure node, instances are evenly distributed among classes

Why use Gini Impurity and Entropy?

- Easy to compute
- Provide a clear interpretation of the quality of the split
    - This makes it easier to determine which feature is used for the split
- Both are less sensitive to outliers and noisy data

    - Makes them more reliable for decision tree construction

Example

Here's an example of a Decision Tree for predicting whether a player will score a touchdown:

Root Node: Red Zone Attempts (Feature: Number of times the player's team has attempted a play in the opponent's red zone)

- Threshold: 3 attempts
- Split: If the player's team has attempted 3 or more plays in the red zone, go to the left child node. Otherwise, go to the right child node.

  Left Child Node: Target Share (Feature: Percentage of targets the player has received in the game)

- Threshold: 20%
- Split: If the player has received 20% or more of the team's targets, go to the left grandchild node. Otherwise, go to the right grandchild node.

  Left Grandchild Node: Yards Per Route Run (Feature: Average yards gained per route run by the player)

- Threshold: 2.5 yards
- Leaf Node: If the player has gained 2.5 or more yards per route run, predict Touchdown (Class 1). Otherwise, predict No Touchdown (Class 0).

  Right Child Node: Rushing Attempts (Feature: Number of rushing attempts by the player)

- Threshold: 5 attempts
- Split: If the player has 5 or more rushing attempts, go to the left grandchild node. Otherwise, go to the right grandchild node.

  Right Grandchild Node: Goal-Line Carries (Feature: Number of carries the player has had on the goal line)

- Threshold: 2 carries
- Leaf Node: If the player has had 2 or more goal-line carries, predict Touchdown (Class 1). Otherwise, predict No Touchdown (Class 0).