Lecture1b : Classification Trees

⚠️ **There might be errors, so please read with caution.**

Main Slide Link

Examples Slide Link

A **Classification Tree** is a machine learning model used for making decisions based on a set of input data. It works by asking questions that split the data into smaller subsets (nodes) until each subset (leaf) contains data points that belong to a single category. The tree structure is intuitive and easy to interpret.

▼ Steps to Build a Classification Tree

1. **Start with the Entire Dataset**:
    - Begin with your full dataset and the goal is to classify data into distinct classes based on certain features (independent variables).
2. **Choose the Best Feature for the Root**:
    - Select the feature that best splits the data into groups based on the target variable (dependent variable).
    - Use metrics like **Gini Impurity** or **Entropy** to determine the best feature. The goal is to find the feature that best separates the classes.

```
Gini Impurity for a Leaf
= 1 - (the probability of “yes”)² - (the probability of “no”)²
= 1 - (the number for yes/the total for the leaf)²
   - (the number for no/the total for the leaf)²

Total Gini Impurity = weighted average of Gini Impurities for the Leaves

for numeric values,
first sort the rows, from lowest to highest.
```

then, calculate the average for all adjacent rows.
lastly, calculate the Gini Impurity values for each average.

3. **Split the Data**:
   - Based on the best feature (e.g., “Loves Soda”), split the data into two branches:
     - One branch for cases where the feature is true (e.g., loves soda).
     - Another branch for cases where the feature is false (e.g., doesn’t love soda).
4. **Repeat the Splitting Process**:
   - For each new branch, apply the same procedure to further split the data based on the remaining features.
   - This continues recursively until one of the stopping criteria is met (e.g., no further improvement, pure nodes, or the tree reaches a maximum depth).
5. **Stopping Criteria**:
   - **Pure Nodes**: Stop if all data points in a node belong to the same class (pure node).
   - **Max Depth**: Stop if the tree reaches a pre-defined maximum depth.
   - **Minimum Samples per Leaf**: Stop if a node has fewer than a minimum number of data points.
   - **Maximum Leaf Nodes**: Stop if there are too many leaves in the tree.
6. **Assign Class Labels to the Leaves**:
   - Once you’ve reached the leaf nodes, assign a **class label** to each leaf based on the majority class of the data in that leaf.
   - For example, if a leaf contains 3 people who love *Troll 2* and 1 person who doesn’t, the label for that leaf is **"Loves Troll 2"**.

▼ Pruning

Pruning is the process of **removing parts of the tree** that are overly specific and do not improve the model's ability to generalize to new data. Pruning helps **avoid overfitting**, which occurs when a tree learns the details of the training data too well and performs poorly on unseen data.

Why Prune a Tree?

- **Overfitting**: A very deep tree might classify every point in the training data perfectly but fail to generalize well to new data.
- **Tree Complexity**: The more complex the tree, the harder it is to interpret. A simpler tree is easier to understand and usually performs better on unseen data.
- **Improve generalization**

Types of Pruning:

1. **Pre-Pruning (Early Stopping)**:
    - This involves stopping the tree from growing once it reaches a certain complexity.
    - You can set limits such as:
        - The minimum number of samples required at each leaf.
        - The maximum depth of the tree.
    - Pre-pruning helps **limit** the growth of the tree, but it doesn’t necessarily give you the optimal tree.
2. **Post-Pruning (Cost Complexity Pruning)**:
    - After the tree is fully grown, you cut off (prune) branches that do not provide a significant improvement.
    - The idea is to remove branches that **do not add value** to the predictive power of the model. This is usually done by evaluating how much each split improves accuracy on a validation set.

    Post-Pruning Process:

    - Start with a fully grown tree.

- Remove one node at a time, then check how removing it affects model performance on a validation dataset.
- If the accuracy improves or stays the same, the branch is pruned.
- The process continues until pruning further would worsen the performance.

Pruning Example:

Imagine you have a tree that splits based on the feature **Loves Soda** and goes further into **Age**.

- If splitting based on **Age** results in a very specific subset (e.g., only one person) and that split doesn't improve classification performance, you might prune that part of the tree.
- The goal is to keep the tree simple but still effective at classifying new data.

▼ Overfitting

Definition: Overfitting occurs when a machine learning model learns **too much** from the training data, capturing not just the underlying patterns but also the **noise** or random fluctuations in the data. As a result, the model performs very well on the training set but **fails to generalize** to new, unseen data.

- **Example**: Imagine a decision tree that perfectly classifies every point in the training data but is extremely complex (deep tree with many branches). It might classify the training data perfectly but struggle with new data because it has memorized the details of the training set, not the general trend.

Signs of Overfitting:

- High accuracy on the training set but low accuracy on the test set.
- The model is too complex or has too many parameters for the amount of data available.

▼ Underfitting

Definition: Underfitting occurs when a model is **too simple** to capture the underlying patterns of the data. This can happen if the model is not complex enough or if it is overly constrained. The model performs poorly on both the training set and the test set because it has not learned enough from the data.

- **Example**: A decision tree that only has one or two branches might be too simplistic to learn any meaningful patterns from the data, leading to poor performance on both the training and test data.

Signs of Underfitting:

- Low accuracy on both the training set and test set.
- The model is too simple (e.g., a linear model for complex data).

▼ Cross Validation

Definition: **Cross Validation** is a technique used to evaluate the performance of a machine learning model by splitting the data into multiple subsets (folds). The model is trained on some of these folds and tested on the remaining fold(s) to assess how well it generalizes to unseen data. This helps ensure that the model's performance is robust and not overly dependent on any particular training set.

- **K-Fold Cross Validation**: One of the most common types, where the data is split into **K** subsets (folds). The model is trained on **K-1 folds** and tested on the remaining fold. This process is repeated K times, each time with a different fold as the test set.

  Example: If you use 5-fold cross validation, the dataset is divided into 5 equal parts. The model is trained on 4 parts and tested on the 1 remaining part, repeated 5 times so that each part is used as the test set once.

Purpose of Cross Validation:

- **Estimate model performance**: Helps assess how well the model will perform on new, unseen data.
- **Reduce variance**: Ensures the model is evaluated over multiple data splits, giving a more reliable estimate of performance.
- **Hyperparameter tuning**: Often used to select the best parameters for a model by evaluating different configurations over cross-validation folds.

Summary Table

Term	Definition	Key Sign
Overfitting	Model learns too much from training data, capturing noise along with patterns.	High accuracy on training set, low accuracy on test set.
Underfitting	Model is too simple and cannot capture the underlying patterns of the data.	Low accuracy on both training and test sets.
Pruning	Process of simplifying a decision tree by removing branches that don't improve the model's performance.	Reduces complexity, improves generalization, prevents overfitting.
Cross Validation	Technique to evaluate model performance by splitting the data into multiple subsets (folds) and testing iteratively.	Model performance is evaluated on multiple splits of the data.