**Part 4: Models based on trees:**

1.1. A classification tree can return class probabilities by using the following approach: At each leaf node, the class probabilities are calculated as the proportion of training samples belonging to each class in that node.  
For example, if a leaf node contains 40 samples, with 30 samples from Class A and 10 samples from Class B, the class probabilities would be: P(Class A)=30/40=0.75, P(Class B)=10/40=0.25. These probabilities can then be used for predictions or further analysis. Most libraries, such as scikit-learn, provide a “predict\_proba” method for decision tree classifiers that computes these probabilities.

After class probabilities most often, the algorithm does argmax on the highest probability and return the classification that he decides to choose.

1.2. Measures for Classification Trees:

**1. Gini Index**: Measures impurity of a node, merits are computationally efficient and works well in most scenarios. The major drawback is less interpretable than entropy.

**2. Entropy (Information Gain)**: Measures the amount of information or uncertainty in a node. Merits are theoretically grounded in information theory and more interpretable. The major drawback is computationally more expensive due to the logarithmic calculation.

**3. Classification Error**: Measures the misclassification rate. Merits simple and intuitive. The major drawbacks are less sensitive to changes in node purity, making it less effective for tree building.

Measure for Regression Trees is **Mean Squared Error (MSE)**:Measures the variance of the target variable within each node. The merit is directly optimizing for variance reduction in the predictions.

1.3. The Gini index represents the probability of incorrectly classifying a randomly chosen sample if it were labeled according to the distribution of classes in the node. The range is 0 pure node, all samples belong to one class to a maximum value where is the most impure, uniform distribution of classes. The interpretation is a lower Gini index indicates a purer node, meaning it is better at distinguishing between classes.

1.4. The disadvantages are first **overfitting** trees can grow too complex, capturing noise in the training data and performing poorly on unseen data, **instability** small changes in the dataset can result in drastically different trees due to the greedy splitting algorithm, **bias towards features with more levels** categorical features with many unique values tend to dominate splits, even when not truly important and **prone to local optima** the greedy nature of tree-building algorithms may miss globally optimal splits.

1.5. The **aim** of pruning reduces overfitting by simplifying the tree, making it less sensitive to noise in the training data and improving generalization.

**The methods** are **pre-pruning (Early Stopping)** Stop tree growth early based on criteria like maximum depth, minimum number of samples per leaf, or minimum information gain. Second method is **post-pruning** grow the tree fully and then remove branches that contribute the least to predictive power.

Methods include cost complexity pruning which minimize a cost function that balances tree complexity and prediction accuracy and cross-validation which remove nodes and evaluate performance on validation data.

1.6. The **Problem** of categorical features with many levels they can dominate the splitting process they tend to reduce impurity significantly due to the large number of small groups, even if the feature is not meaningful.

The **solution** is first **group Level** which means to combine similar levels into broader categories based on domain knowledge or statistical similarity, second **one-Hot Encoding** that transform the categorical feature into binary columns (though this increases dimensionality) and may **Tree Ensembles** to use ensemble methods like Random Forest or Gradient Boosting, which average splits over multiple trees and reduce bias toward features with many levels.

2.1. **Bagging (Bootstrap Aggregating)** is an ensemble learning technique designed to reduce variance and improve the stability and accuracy of machine learning algorithms.  
The key steps are **Bootstrap Sampling** which means multiple datasets are created by randomly sampling with replacement from the original dataset, **Training Models** a separate model (e.g., decision tree) is trained on each bootstrap sample and **Aggregation** in predictions from all models are combined to produce the final output for classification this is majority voting and for regression is averaging predictions. Bagging reduces overfitting and enhances generalization by combining multiple models trained on slightly different datasets.

2.2. **Small m** encourages diversity among trees, as different features are considered at each split. Helps when there are irrelevant or noisy features but **i**f m is too small, relevant features may be missed, leading to underfitting. **Large m** allows the model to consider more relevant features at each split. Improves individual tree performance but increases correlation between trees, reducing the benefit of aggregation and potentially overfitting. The optimal m balances diversity and strength of individual trees.

2.3. **Out-of-Bag Sampling (OOB)** in bagging, each bootstrap sample excludes some observations. These excluded samples from the OOB set for a given tree. OOB samples are used to evaluate the model's performance without requiring a separate validation dataset.

The **size of OOB Set** for large datasets, approximately 1/e≈0.368 of the data is left out in each bootstrap sample because the probability of a sample being excluded in one draw is 1−1/n and over n draws, this approaches 1/e.

2.4. Feature importance in Random Forest can be measured in two main ways. First way, **Gini Importance (Impurity Reduction)**, measures the total reduction in impurity (e.g., Gini index or entropy) brought by a feature across all trees. Features contributing more to node splits have higher importance scores. The second way is **Permutation Importance**, randomly permute the values of a feature and observe the decrease in model performance (e.g., accuracy or R^2). A significant drop in performance indicates the feature's importance. Both methods are implemented in popular libraries like scikit-learn.