Digital Skills Training for Student
Central Computer Center, KUET

ICT DIVISION
FUTURE IS HERE

Machine Learning with Python

Session 22: Decision Tree

Shah Muhammad Azmat Ullah
Dept. of ECE
Khulna University of Engineering & Technology

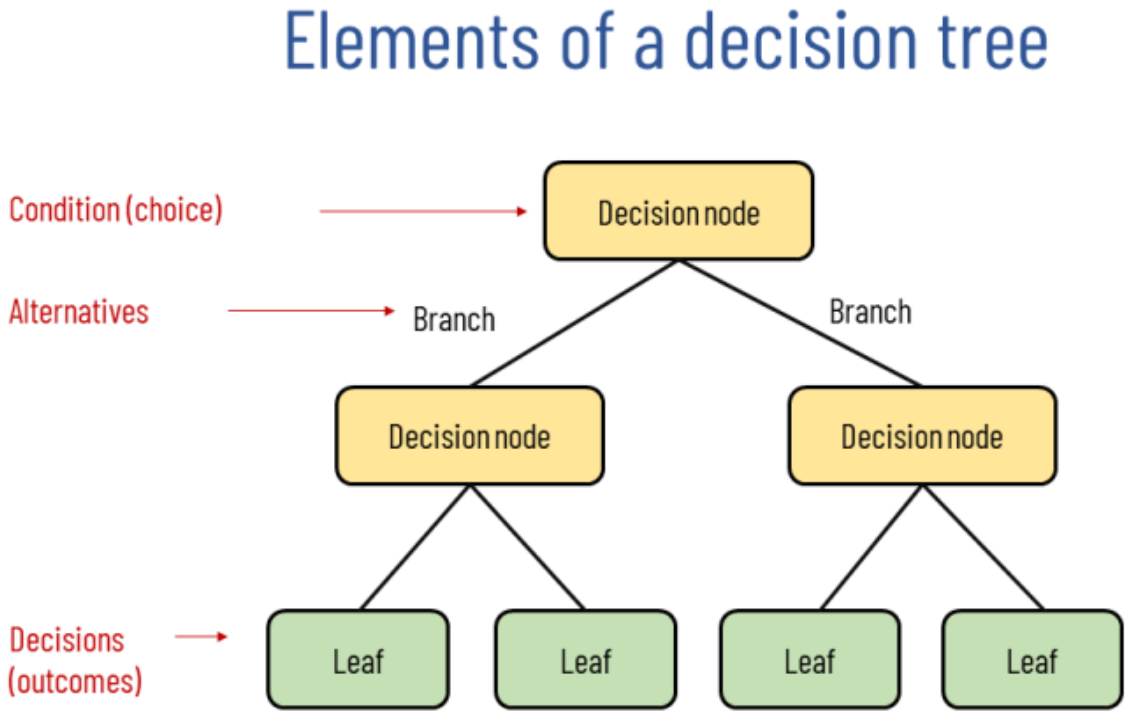
Contents

Session 22: Decision Tree

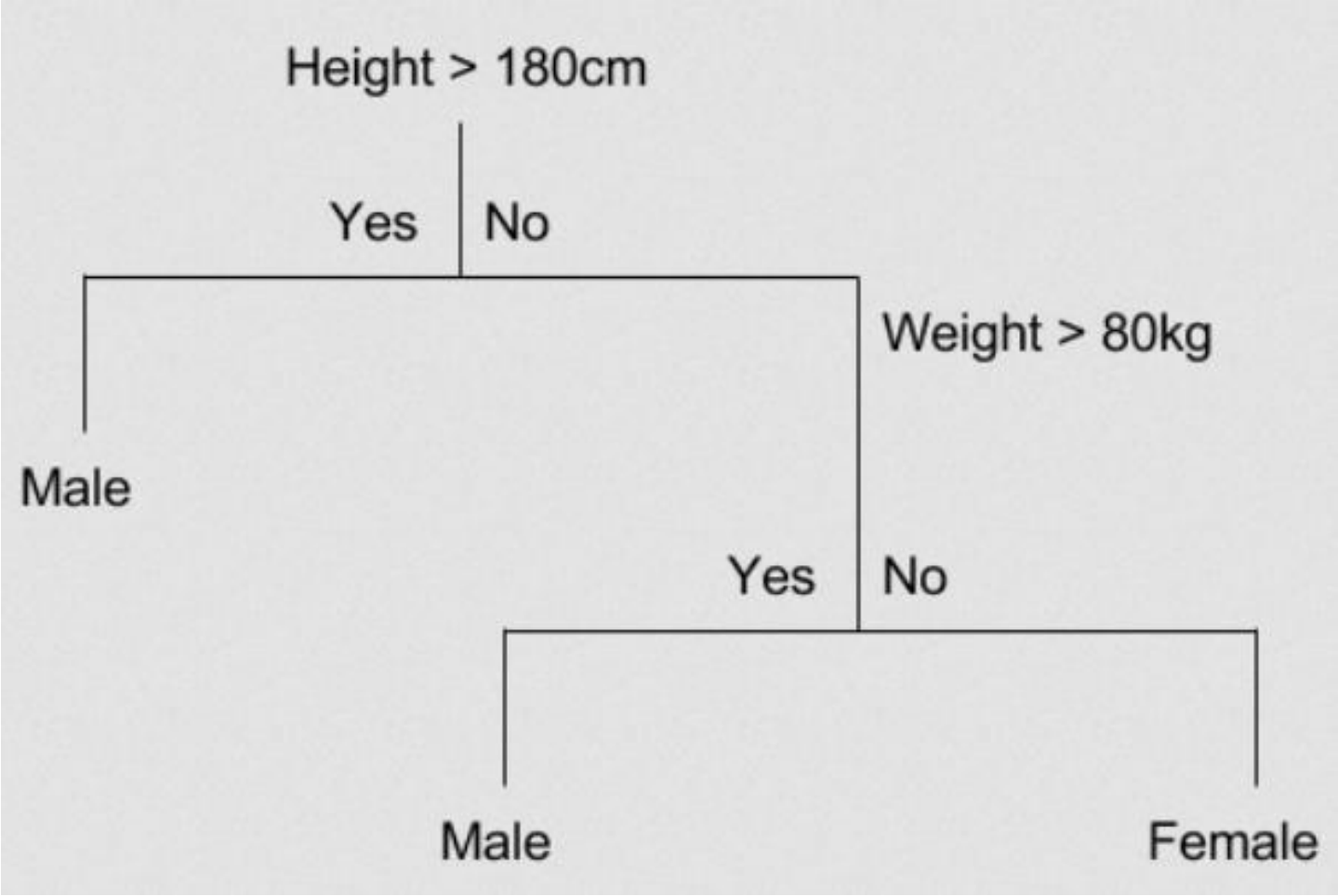
- Introduction
- Intuition behind DT
- Terminology in Decision Tree
- The CART Algorithm - Classification
- Splitting Categorical Features
- Splitting Numerical Features
- Understanding Gini Impurity?
- Geometric Intuition of DT

Introduction

A **decision tree** is a flowchart-like structure used to make decisions or predictions. It consists of **nodes** representing decisions or tests on attributes, **branches** representing the outcome of these decisions, and leaf nodes representing final outcomes or predictions. Each internal node corresponds to a test on an attribute, each branch corresponds to the result of the test, and each **leaf** node corresponds to a class label or a continuous value.

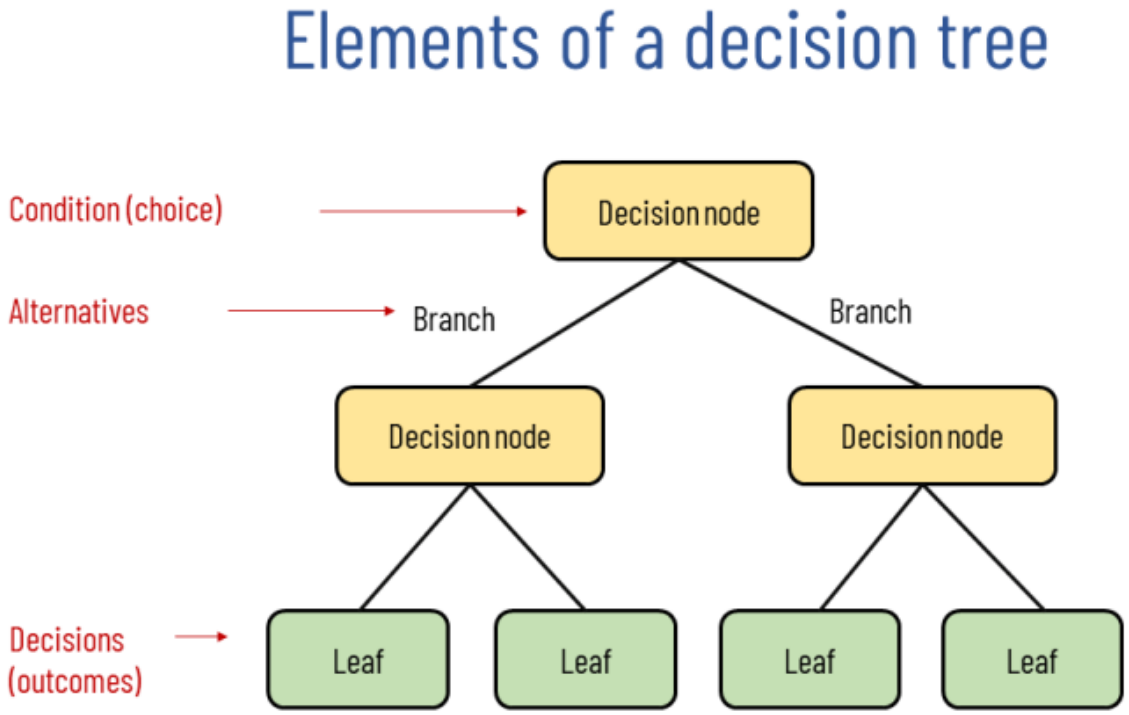


Introduction



Introduction

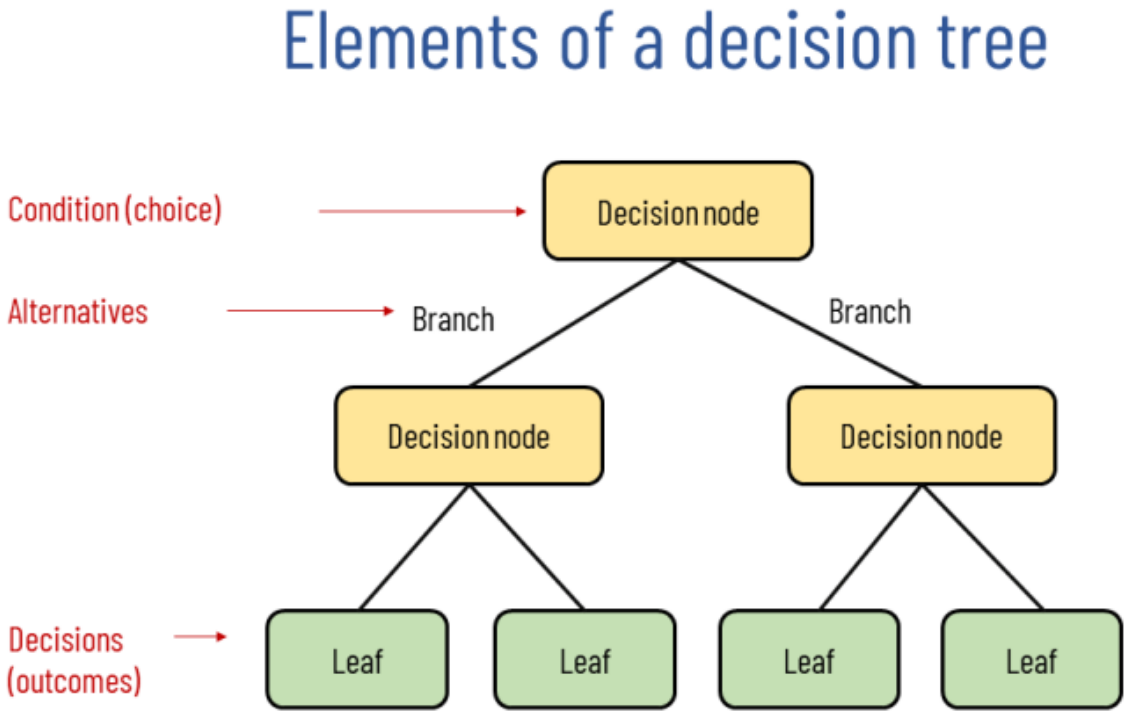
A Decision Tree is a supervised learning algorithm used for both classification and regression tasks. It splits the dataset into smaller subsets while incrementally developing a tree structure. At each node, the algorithm selects the feature that best splits the data based on a certain criterion. Decision trees are simple, interpretable, and can model complex, non-linear relationships.



Introduction

The idea behind decision trees mimics human decision-making:

- A decision tree asks a series of "if then" questions to reach a conclusion.
- Each internal node represents a decision on a feature, while leaf nodes represent outcomes or predictions.
- It progressively divides the data into homogeneous subsets until no further meaningful splits are possible or a stopping condition is met.



Introduction

- **Root Node:** The topmost node of the tree, representing the entire dataset.
- **Internal Nodes**: Nodes that split the dataset based on feature conditions.
- **Branches**: Represent the outcome of a decision or test, leading to another node.
- **Leaf Nodes**: Terminal nodes that represent the final prediction or output class.
- **Splitting**: The process of dividing a node into child nodes based on a feature.
- **Pruning**: The process of removing sections of the tree to reduce overfitting.
- **Depth of Tree**: The length of the longest path from the root to a leaf.
- **Impurity**: A measure of the heterogeneity of the data at a node.

How Decision Trees Work

The process of creating a decision tree involves:

1.Selecting the Best Attribute: Using a metric like Gini impurity, entropy, or information gain, the best attribute to split the data is selected.

2.Splitting the Dataset: The dataset is split into subsets based on the selected attribute.

3.Repeating the Process: The process is repeated recursively for each subset, creating a new internal node or leaf node until a stopping criterion is met (e.g., all instances in a node belong to the same class or a predefined depth is reached).

How Decision Trees Work

•**Gini Impurity**: Measures the likelihood of an incorrect classification of a new instance if it was randomly classified according to the distribution of classes in the dataset.

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

- where *pi* is the probability of an instance being classified into a particular class.

•**Entropy**: Measures the amount of uncertainty or impurity in the dataset.

- where *pi* is the probability of an instance being classified into a particular class.

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

How Decision Trees Work

•**Information Gain**: Measures the reduction in entropy or Gini impurity after a dataset is split on an attribute.

$$\text{InformationGain} = \text{Entropy}_{\text{parent}} - \sum_{i=1}^{n} \left(\frac{|D_i|}{|D|} * \text{Entropy}(D_i) \right)$$

- where *Di* is the subset of *D* after splitting by an attribute.

How Decision Trees Work

The Gini index is a metric for the classification tasks in CART. It stores the sum of squared probabilities of each class. It computes the degree of probability of a specific variable that is wrongly being classified when chosen randomly and a variation of the Gini coefficient. It works on categorical variables, provides outcomes either “successful” or “failure” and hence conducts binary splitting only. The degree of the Gini index varies from 0 to 1,

- Where 0 depicts that all the elements are allied to a certain class, or only one class exists there.
- Gini index close to 1 means a high level of impurity, where each class contains a very small fraction of elements, and
- A value of 1-1/n occurs when the elements are uniformly distributed into n classes and each class has an equal probability of $1/n$. For example, with two classes, the Gini impurity is $1 - 1/2 = 0.5$.

How Decision Trees Work

In conclusion, Gini impurity is the probability of misclassification, assuming independent selection of the element and its class based on the class probabilities.

https://scikit-learn.org/1.5/api/sklearn.tree.html

https://scikit-learn.org/1.5/modules/generated/sklearn.tree.DecisionTreeClassifier.html

How Decision Trees Work

Advantages of CART

•Results are simplistic.
•Classification and regression trees are Nonparametric and Nonlinear.
•Classification and regression trees implicitly perform feature selection.
•Outliers have no meaningful effect on CART.
•It requires minimal supervision and produces easy-to-understand models.

Limitations of CART

•Overfitting.
•High Variance.
•low bias.
•the tree structure may be unstable.

How Decision Trees Work

When using Decision Tree (DT) classifiers, there are several concerns and limitations that users should be aware of:

1. Overfitting

- Problem: Decision Trees often grow very complex, capturing noise and small patterns in the training data.

- Solution: Use pruning (e.g., reduce tree depth or remove unnecessary nodes). Set constraints like maximum depth, minimum samples per leaf, or minimum samples to split. Use ensemble methods like Random Forest or Gradient Boosted Trees.

How Decision Trees Work

2. High Variance

- Problem: A small change in the training data can lead to a significantly different tree structure.
- Solution: Train multiple trees (e.g., in a Random Forest). Use techniques like bagging to stabilize predictions.

3. Bias Toward Features with More Levels

- Problem: DTs may favor features with more unique values (e.g., ID-like attributes) because they tend to give better splits.
- Solution: Use proper feature engineering or algorithms like Random Forest that handle this bias better.

How Decision Trees Work

4. Difficulty in Handling Continuous Variables

- Problem: Continuous variables need careful thresholding, which can result in overfitting or suboptimal splits.
- Solution: Use discretization or ensemble methods to manage continuous variables effectively.

5. Computational Complexity for Large Data

- Problem: Growing a deep tree can be computationally expensive for large datasets.
- Solution: Use efficient implementations like scikit-learn, and limit tree depth or the number of features considered for splits.

How Decision Trees Work

6. Lack of Smooth Decision Boundaries

- Problem: Decision Trees create step-like decision boundaries, which might not be ideal for certain datasets.
- Solution: Use ensemble techniques (e.g., boosting) to create smoother boundaries.

7. Interpretability in Large Trees

- Problem: While small trees are easy to interpret, large trees become complex and lose their intuitive appeal.
- Solution: Limit tree growth or use interpretable rules from simpler models.

How Decision Trees Work

8. Imbalanced Data

- Problem: Decision Trees can be biased towards the majority class in imbalanced datasets.
- Solution: Use techniques like oversampling, undersampling, or cost-sensitive training.

Practice

https://www.kaggle.com/code/hkhnhduy/decision-tree-classifier-tutorial

https://www.kaggle.com/code/serhatyzc/diabetes-prediction-with-cart/notebook

https://www.geeksforgeeks.org/text-classification-using-decision-trees-in-python/

Thank You!