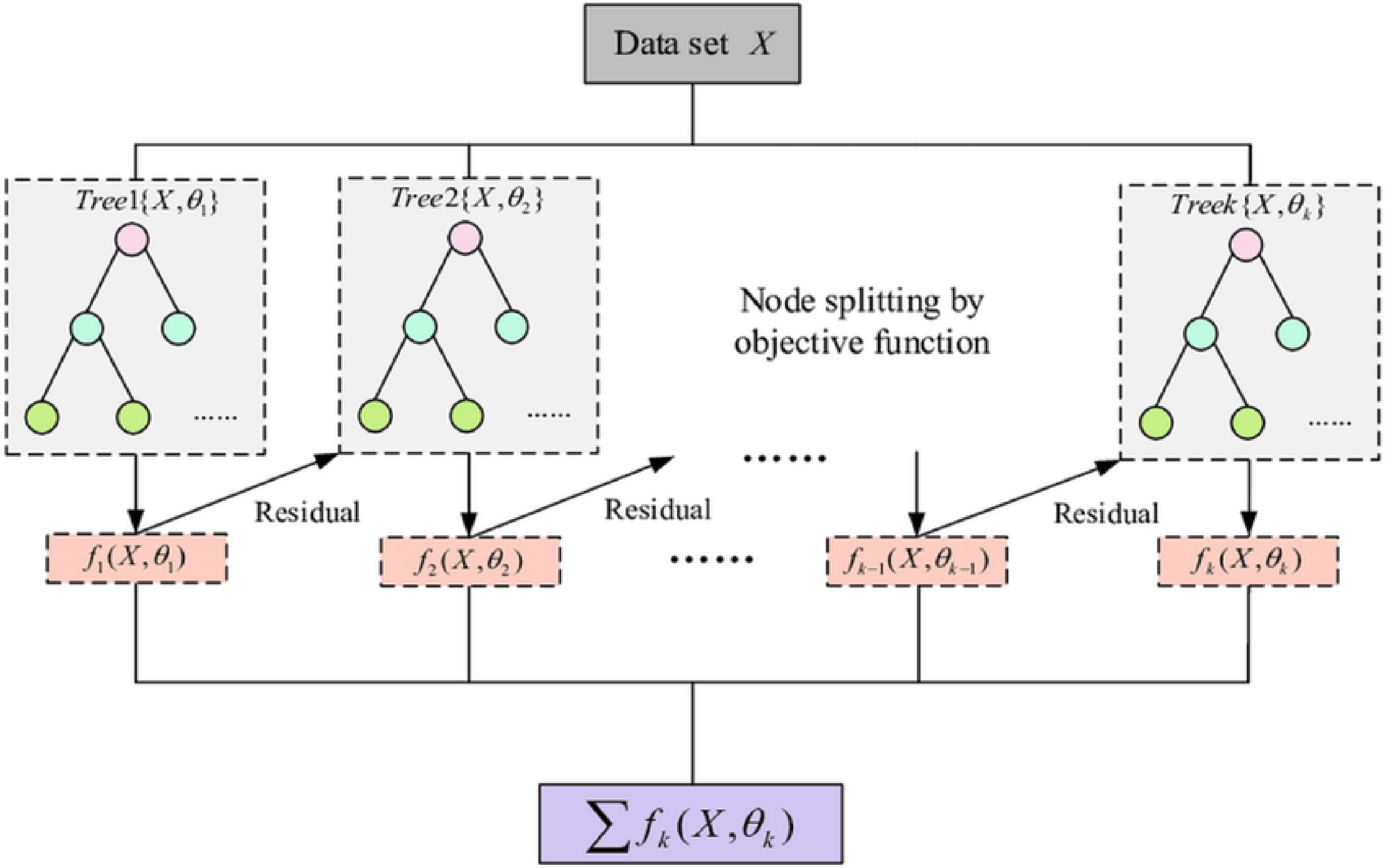
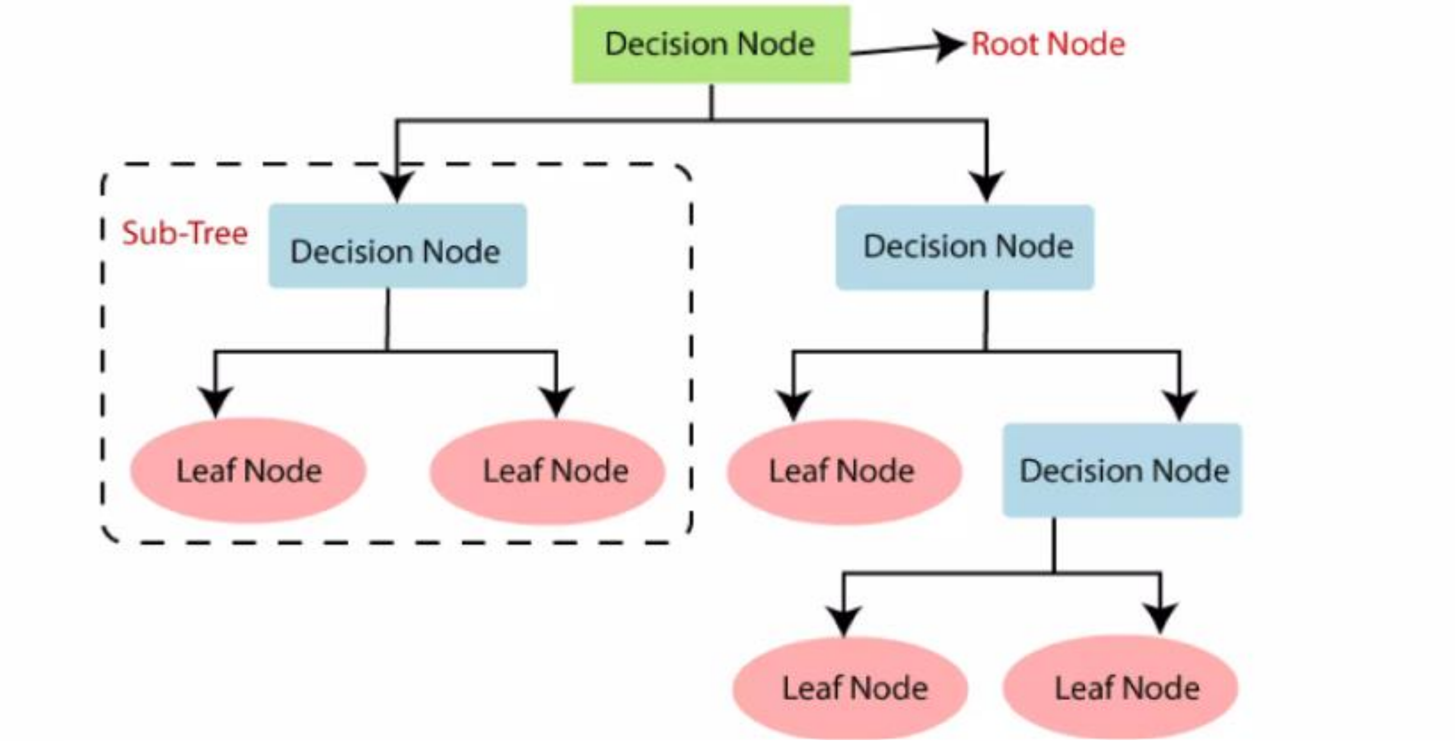
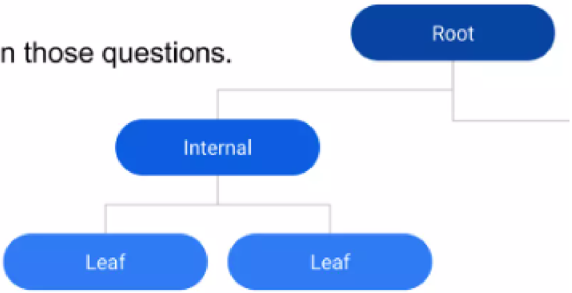
XGBOOST REGRESSION

Xgboost is an ensemble techniques supervised learning and it uses the boosting technique. It minimizes a loss function by sequentially building trees and improving predictions iteratively. The decision tree gets trained based on the residual values.





1. Nodes and Branches
   - A Decision Tree consists of nodes and branches.
   - Nodes represent decisions or questions based on features.
   - Branches represent possible outcomes or decisions based on those questions.



2. Root Node
   - The top-most node is called the root node.

3. Internal Nodes
   - Nodes other than the root node are internal nodes.
   - They represent intermediate decisions/questions.

4. Leaf Nodes
   - Terminal nodes are called leaf nodes.
   - They represent the final outcomes or predictions.

How XGBOOST Works:

Step-1 : Find the base model(average of the output field)

Step-2 : Calculate the Residual(output value – average value)

Step-3 : Built the Decision binary tree with the help of input values along with the found residual values

Step-4 :Calculate the similarity weight for left, right, root node:

$$\frac{Sum\ Of\ Residuals}{Number\ Of\ Residuals + \lambda}$$

Step-5 :Calculate the Gain : Left node weight+ Right node weight-root node weight

Step-6 :Predict the output: Base Model+0.50 * t1+.... [Note: t value calculated using: sum of residuals of tree/total num of residual]

Repeat steps from 2- 6, we will build another tree with new residuals and make a prediction, we keep building trees until the residuals are super small, or we have reached the maximum number.

Pros:

1. combining multiple models leads to superior accuracy compared to single models like decision trees
2. It’s optimized for handling large datasets and Helps prevent overfitting. The training speed is also faster.
3. XGBoost provides insights into feature importance. This helps understand which factors significantly impact the predictions.
4. Automatically identifies the most important features and prunes unnecessary ones.

Cons:

1. Compared to simpler models, XGBoost can be more complex to understand and fine-tune.
2. Potential for Overfitting.
3. XGBoost may not perform well on small datasets compared to simpler models.
4. Requires careful tuning of parameters like learning rate, max depth, and number of estimators to achieve optimal performance.