Decision Tree Classifier

FROM CONCEPT TO ALGORITHM

Aryan Chauhan | ML Part 1 | 28-07-2024

Table of Contents

Introduction

Welcome to my ML concepts archive where I explain the concepts of the ML Algorithms behind the projects I make. In this document I'm going to be explaining a random forest classifier from scratch and how I implement it. The implementation is done in the titanic survival project and Parkinson disease research paper project.

For the reference of code of the projects I have mentioned above you can visit the following git repos –

1. **Titanic Prediction:** https://github.com/aryanc381/Titanic-Survival-Rate-Prediction-using-RFC
2. **Parkinson Disease:** https://github.com/aryanc381/Parkinson-Disease-Prediction-using-ML-Models

Pre-requisite - Decision Tree Classifier

Example

To understand a decision tree classifier, let me explain an example which will clear the entire concept. Consider that we must figure out the if a patient has a high risk or low risk of having a heart disease. A decision tree asks several questions to the patient to figure out the risk. As simple as that! It's like a 20 Questions game where each answer helps us get closer to the prediction.

Q1. How old are you? (The Root Node)

If the person is under 18, we go down one path.

If the person is between 18-30 years old, the person is at a low risk of having heart disease.

If the person is above 30 years old, they might have a higher chance of having heart disease, so we go down another different path.

Q2.A How much do they weigh? (for those below 18)

If the person is below 60kgs, we decide that they are at low risk.

If the person is above 60kgs, we decide that they are at high risk.

Q2. B Do they smoke? (for those above 18)

If the person doesn't smoke, they are at low risk.

If the person smokes, they are at high risk.

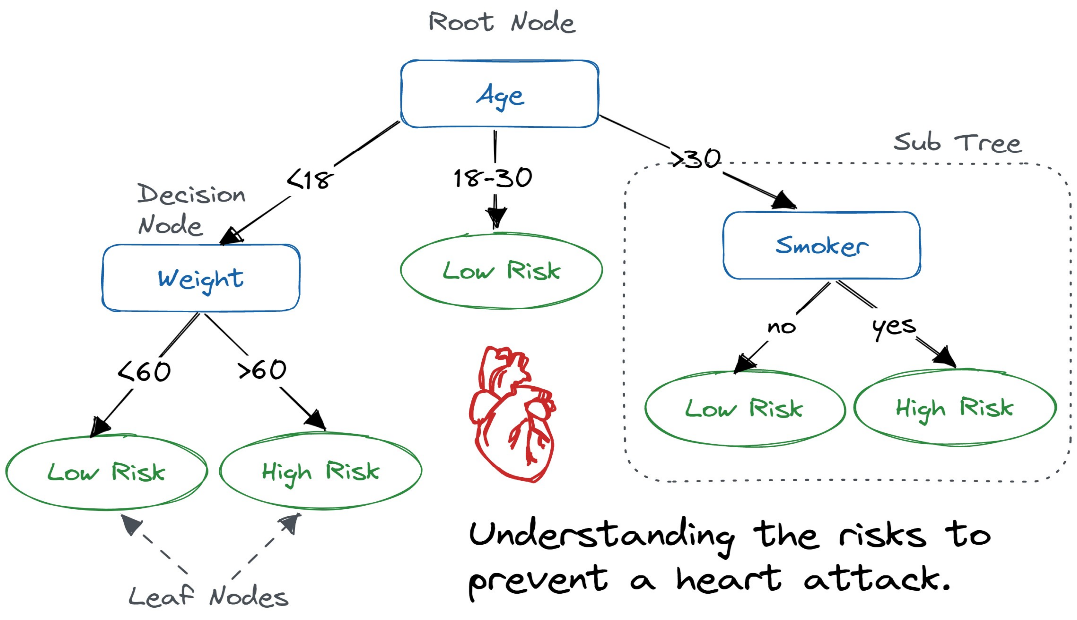
Now consider an example,

Name: Siddharth Badjate.

Age: 20 years old.

Smoking: Doesn't smoke.

Prediction: The tree immediately tells us that Siddharth has very low chances of heart disease.



1. **Root Node**: This is where the first question is asked. In our case, it is about the person's age.

2. **Decision Node**: These are the points where the tree asks another question based on the previous answer.
3. **Leaf Node**: These are the end points of the tree where a decision is made, either "Low Risk" or "High Risk".
4. **Branches**: These are the lines that connect the questions and decisions, showing the path you take based on each answer.

A decision tree makes it easy to see how different factors (like age, weight, and smoking) affect the risk of a heart attack. By following the tree, we can quickly decide if someone is at low or high risk. It's a simple and visual way to make decisions based on different pieces of information.

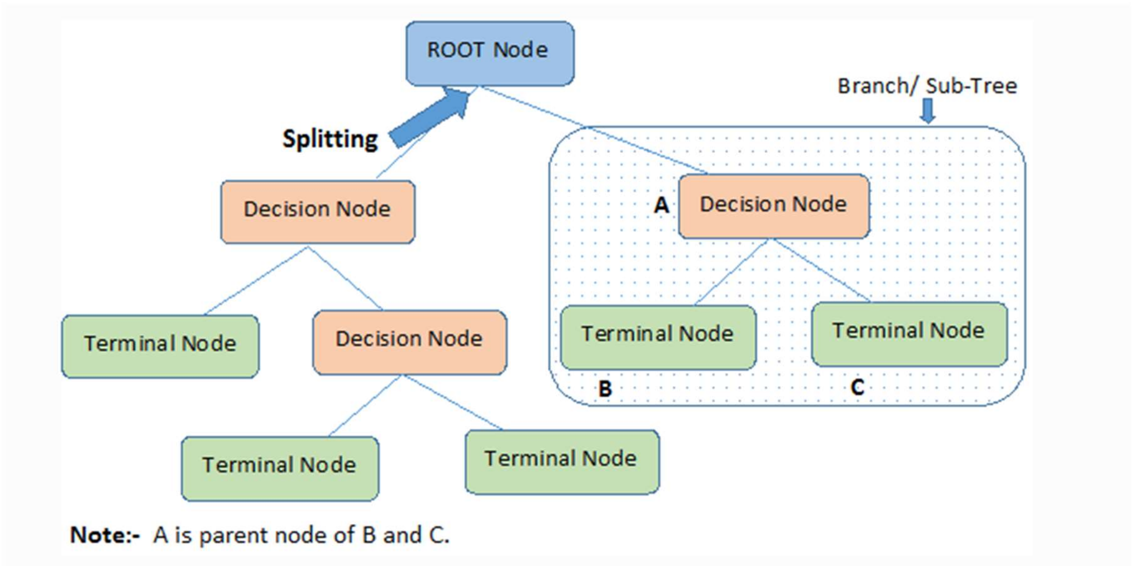
Concept of DTC

A decision tree is a flowchart-like structure used to make decisions or predictions. It's a type of supervised machine learning that can be used for both classification and regression problems, but it is mostly preferred while solving classification problems. It is a tree structure.

In a decision tree, there are two nodes, Decision node and Leaf node. Decision nodes are used to make decisions and have multiple branches and Leaf nodes are the final decisions made by the tree and have no sub/multiple branches.

The decision tree is a graphical representation for getting all possible solutions to a problem/decision based on given conditions. It is called a "tree" as it starts with a single root node and expands into multiple decision nodes which constructs a tree-like structure. In order to build a tree, we use a CART Algorithm – Classification and Regression Tree Algorithm where the tree asks a simple question – (Yes/No) and based on the answer further splits into multiple subtrees.

The structure of a decision tree classifier is as follows:



1. **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
2. **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
3. **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
4. **Branch/Sub Tree:** A tree formed by splitting the tree.
5. **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
6. **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

The algorithm behind the Decision Tree Classifier

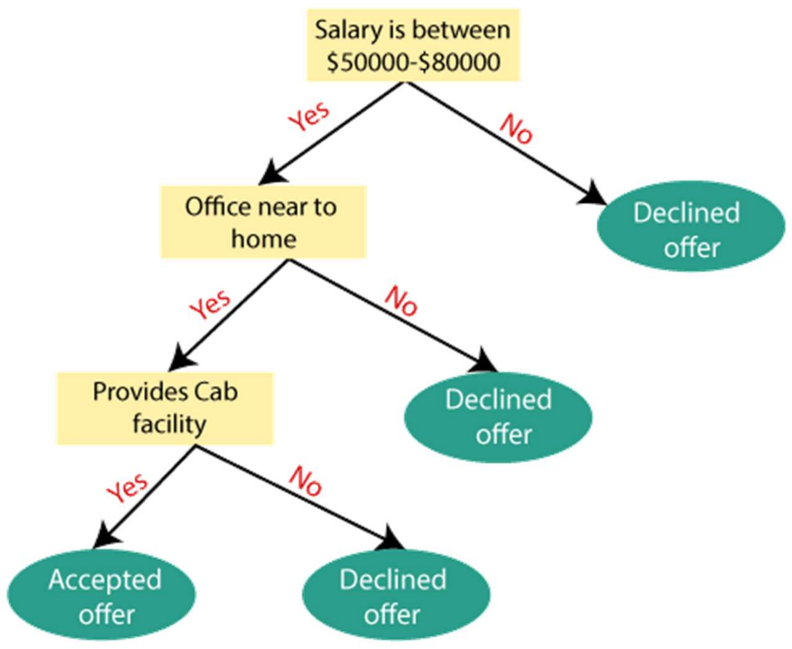
In decision tree, the algorithm again compares the attribute value with the other sub-nodes and moves further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.

- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**

- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measure (ASM)

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

1. **Information Gain**

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:
- Entropy is a metric to measure the impurity in each attribute. It specifies randomness in data. Entropy can be calculated as:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy (each feature)

```
Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)
Here,
S - Total number of samples.
P(yes) - probability of yes.
P(no) - probability of no.
```

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:
  Gini Index= $1- \sum_j P_j^2$

Pruning – Achieving the optimal Decision Tree

Pruning is the process of deleting unnecessary nodes to get the ideal decision tree.

A too large decision tree increases the risk of over-fitting, and a small decision tree may not capture all the features of the dataset. Thus, a technique that reduces the size of the decision tree while maintain the accuracy is called pruning.

There are mainly two types of pruning,

1. Cost Complexity Pruning

Cost Complexity Pruning, also known as weakest link pruning, is a type of post-pruning. It involves trimming the tree to minimize a cost function that balances tree size and fit to the training data.

How it Works:

1. **Initial Tree Growth**: First, a fully grown tree is built without any pruning.
2. **Calculate Cost-Complexity**: For each non-leaf node, calculate a cost complexity value α\alphaα.
   - α\alphaα is the penalty parameter for the number of leaves, controlling the trade-off between tree complexity and fit to the data.
3. **Pruning Step**: The node with the smallest increase in cost complexity (weakest link) is pruned first. This step is repeated iteratively.
4. **Optimal Subtree Selection**: The process continues until the tree is reduced to a single node. During the process, a sequence of subtrees is generated. The optimal subtree is chosen by cross-validation or another validation technique to determine which subtree has the best performance on unseen data.

2. Reduced Error Pruning

Reduced Error Pruning is a simpler form of post-pruning that directly uses a validation set to make pruning decisions.

How it Works:

1. **Initial Tree Growth**: A full decision tree is first grown using the training data.
2. **Pruning Step**: Nodes are pruned if the resulting pruned tree performs no worse than the original tree on a validation set.
   - For each node, temporarily prune the node, replacing it with a leaf.
   - Measure the accuracy of the pruned tree on the validation set.

    - If the pruned tree's accuracy is not worse than the original tree's accuracy, make the pruning permanent.

3. **Repeat Until Stable**: This process continues until no further improvement can be achieved by pruning any nodes.

Advantages of DTC:

1. It is easy to understand and implement as humans use the same cycle to make decisions.
2. It helps to think about all possible outcomes.
3. There is less requirement for data cleaning when compared to other algorithms.

Disadvantages of DTC:

1. The decision tree contains a lot of layers which requires time to decipher.
2. Usually, Decision Tree Classifier have an overfitting issue that is resolved using Random Forest Classifier.

In the next chapter, I shall explain the concept of Random Forest Classifier to reduce the overfitting problem.

If you have any doubts in this document, please feel free to mail me at the given email that I have provided in my profile section of my git.

Git Profile – https://github.com/aryanc381

End of Document