Introduction to Decision Trees

[(https://github.com/learn-co-curriculum/dsc-introduction-to-decision-trees)](https://github.com/learn-co-curriculum/dsc-introduction-to-decision-trees) [(https://github.com/learn-co-curriculum/dsc-introduction-to-decision-trees/issues/new/choose)](https://github.com/learn-co-curriculum/dsc-introduction-to-decision-trees/issues/new/choose)

Introduction

In this lesson, we'll take a look at ***decision tree classifiers***. These are rule-based classifiers and belong to the first generation of modern AI. Despite the fact that this algorithm has been used in practice for decades, its simplicity and effectiveness for routine classification tasks is still on par with more sophisticated approaches. They are quite common in the business world because they have decent effectiveness without sacrificing explainability. Let's get started!

Objectives

You will be able to:

- Describe a decision tree algorithm in terms of graph architecture
- Describe how decision trees are used to create partitions in a sample space
- Describe the training and prediction process of a decision tree

From graphs to decision trees

We have seen basic classification algorithms (a.k.a classifiers), including Naive Bayes and logistic regression, in earlier sections. A decision tree is a different type of classifier that performs a **recursive partition of the sample space**. In this lesson, you will get a conceptual understanding of how this is achieved.

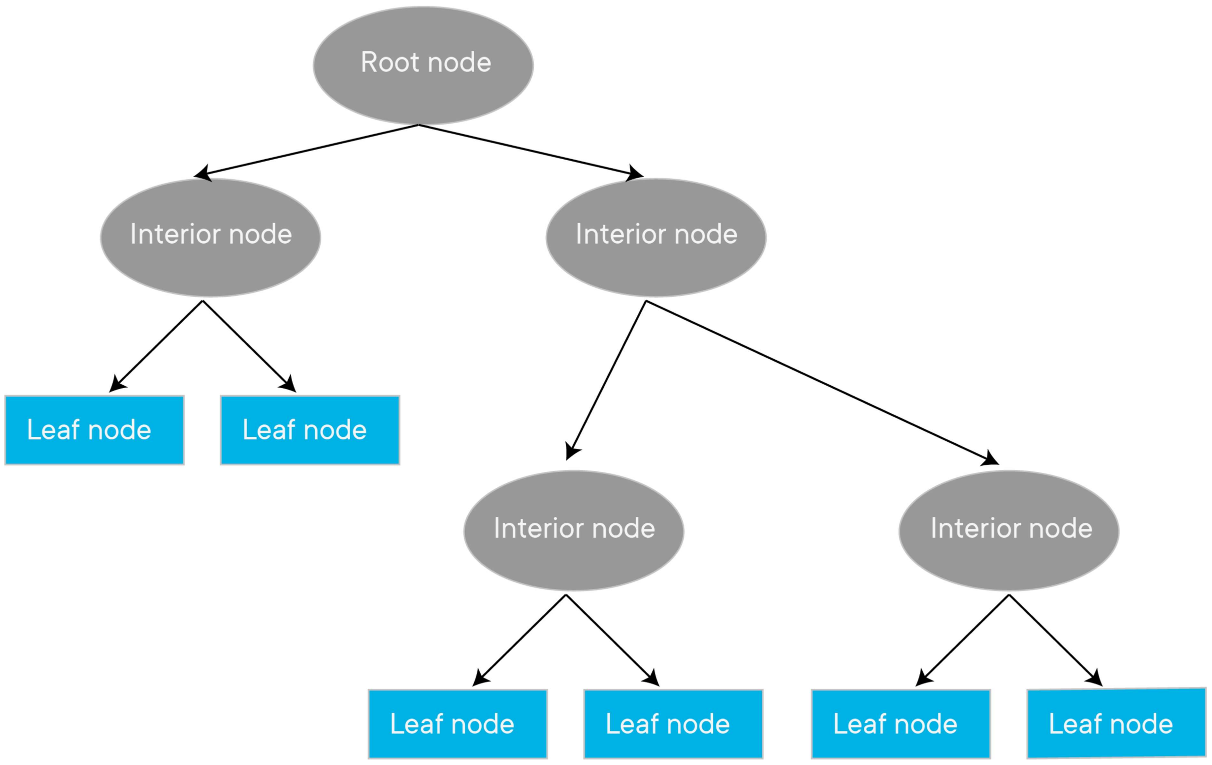
A decision tree comprises of decisions that originate from a chosen point in sample space. If you are familiar with Graph theory, a tree is a **directed acyclic graph with a root called "root node" that has no incoming edges**. All other nodes have one (and only one) incoming edge. Nodes having outgoing edges are known as **internal** nodes. All other nodes are called **leaves**. Nodes with an incoming edge, but no outgoing edges, are called **terminal nodes**.

> **Directed Acyclic Graphs**
>
> In computer science and mathematics, a directed graph is a collection of nodes and edges such that edges can be traversed only in a specified direction (eg, from node A to node B, but not from node B to node A). An acyclic graph is a graph such that it is impossible for a node to be visited twice along any path from one node to another. So, a directed acyclic graph (or, a DAG) is a directed graph with no cycles. A DAG has a **topological ordering**, or, a sequence of the nodes such that every edge is directed from earlier to later in the sequence.

Partitioning the sample space

So, a decision tree is effectively a DAG, such as the one seen below where **each internal node partitions the sample space into two (or more) sub-spaces.** These nodes are partitioned according to some discrete function that takes the attributes of the sample space as input.

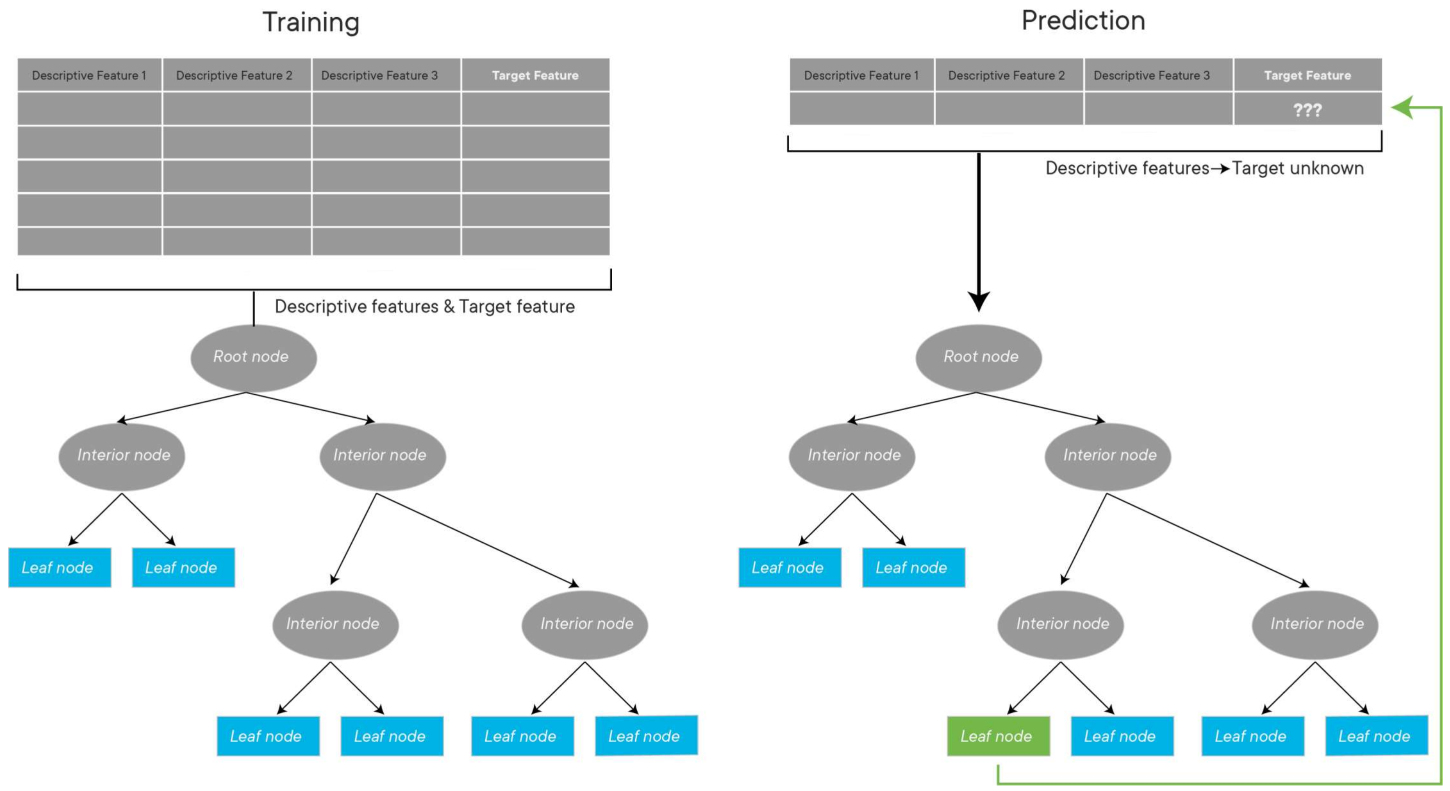
In the simplest and most frequent case, each internal node considers a single attribute so that space is partitioned according to the attribute's value. In the case of numeric attributes, the condition refers to a range.



This is the basic idea behind decision trees: every internal node checks for a condition and performs a decision, and every terminal node (AKA leaf node) represents a discrete class. Decision tree induction is closely related to **rule induction**. In essence, a decision tree is a just series of IF-ELSE statements (rules). Each path from the root of a decision tree to one of its leaves can be transformed into a rule simply by combining the decisions along the path to form the antecedent, and taking the leaf's class prediction as the consequence (IF-ELSE statements follow the form: IF *antecedent* THEN *consequence*).

Definition

> A decision tree is a DAG type of classifier where each internal node represents a choice between a number of alternatives and each leaf node represents a classification. An unknown (or test) instance is routed down the tree according to the values of the attributes in the successive nodes. When the instance reaches a leaf, it is classified according to the label assigned to the corresponded leaf.

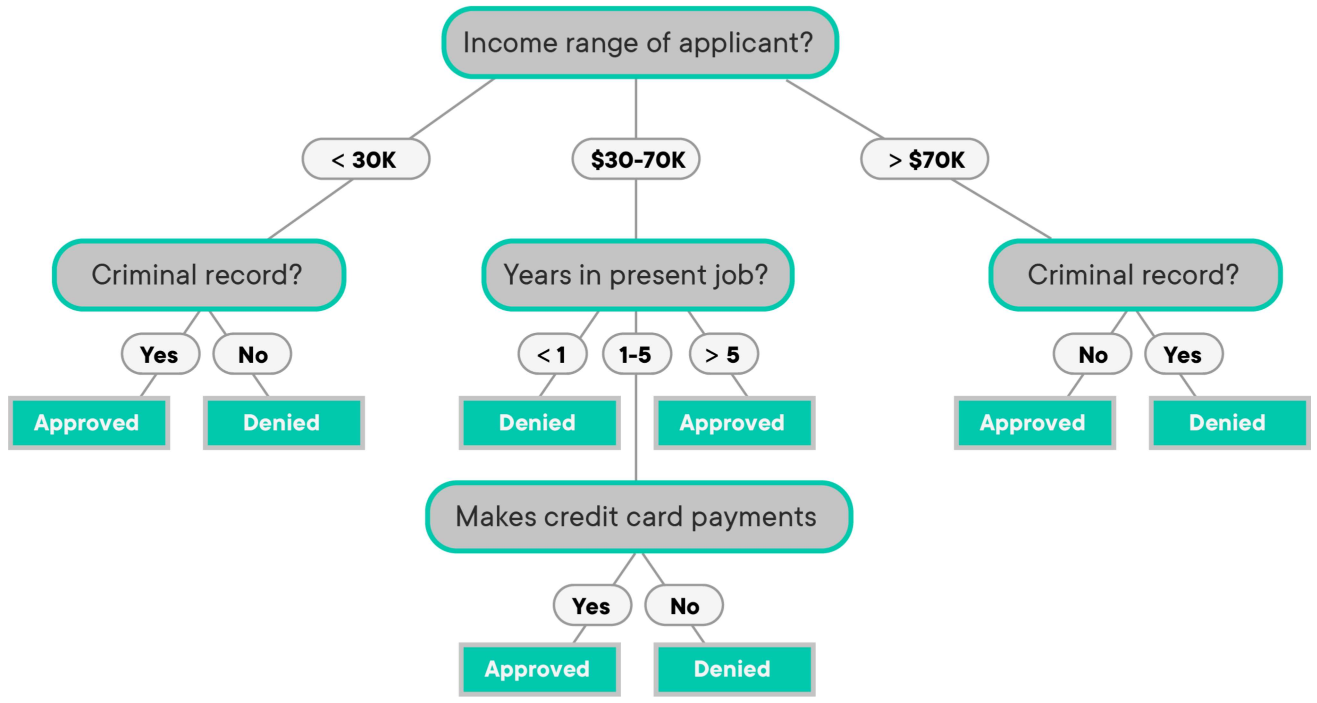


A real dataset would usually have a lot more features than the example above and will create much bigger trees, but the idea will remain exactly the same. The idea of feature importance is crucial to decision trees, since selecting the correct feature to make a split on will affect the complexity and efficacy of the classification process. Regression trees are represented in the same manner, but instead they predict continuous values like the price of a house.

Training process

The process of training a decision tree and predicting the target features of a dataset is as follows:

1. Present a dataset of training examples containing features/predictors and a target (similar to classifiers we have seen earlier).
2. Train the tree model by making splits for the target using the values of predictors. Which features to use as predictors gets selected following the idea of feature selection and uses measures like "**information gain**" and "**Gini Index**". We shall cover these shortly.
3. The tree is grown until some **stopping criteria** is achieved. This could be a set depth of the tree or any other similar measure.
4. Show a new set of features to the tree, with an unknown class and let the example propagate through a trained tree. The resulting leaf node represents the class prediction for this example datum.



Splitting criteria

The training process of a decision tree can be generalized as "**recursive binary splitting**".

> In this procedure, all the features are considered and different split points are tried and tested using some **cost function.** The split with the lowest cost is selected.

There are a couple of algorithms used to build a decision tree:

- **CART (Classification and Regression Trees)** uses the Gini Index as a metric
- **ID3 (Iterative Dichotomiser 3)** uses the entropy function and information gain as metrics

Greedy search

We need to determine the attribute that **best** classifies the training data, and use this attribute at the root of the tree. At each node, we repeat this process creating further splits, until a leaf node is achieved, i.e., all data gets classified.

> This means we are performing a top-down, greedy search through the space of possible decision trees.

In order to identify the best attribute for ID3 classification trees, we use the "information gain" criteria. Information gain (IG) measures how much "information" a feature gives us about the class. Decision trees always try to maximize information gain. So, the attribute with the highest information gain will be split on first.

Let's move on to the next lesson where we shall look into these criteria with simple examples.

Additional resources

- **R2D3 (http://www.r2d3.us/visual-intro-to-machine-learning-part-1/)** : This is highly recommended for getting a visual introduction to decision trees. Excellent animations explaining the training and prediction stages shown above.
- **Dataversity: Decision Trees Intro (http://www.dataversity.net/introduction-machine-learning-decision-trees/)** : A quick and visual introduction to DTs.
- **Directed Acyclic Graphs (https://cran.r-project.org/web/packages/ggdag/vignettes/intro-to-dags.html)** : This would help relate early understanding of graph computation to decision tree architectures.

Summary

In this lesson, we saw an introduction to decision trees as simple yet effective classifiers. We looked at how decision trees partition the sample space based on learning rules from a given dataset. We also looked at how feature selection for splitting the tree is of such high importance. Next, we shall look at information gain criteria used for feature selection.

How do you feel about this lesson?

Have specific feedback?

Tell us here! (https://github.com/learn-co-curriculum/dsc-introduction-to-decision-trees/issues/new/choose)