Machine Learning: A brief introduction to Random Forest

Overview: In our **Machine Learning introduction**, ML is defined as:

a subset of artificial intelligence (AI) that focuses on the development of algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed for each specific task.

Random Forest is an ensemble of decision trees (usually 500-1000 trees) that is used for prediction and classification. It is a method that has been successfully used in many different fields of research and is definitely a popular ML algorithm in health care. **Here** and **here** are a few interesting examples.

Before we go into the specifics of Random Forest, we first need to review **decision trees**, as they are the building blocks of the forest.

Decision Trees:

Decision trees are a type of classifier that aims to separate different outcomes (for example, patients with a tumor subtype A vs. subtype B). It starts with all observations and then searches for the best feature and splitting value. For example, if patient age, BMI, biological sex, diabetes status, and hypertension indicator are my potential predictors, a decision tree will not only look to see which feature is the best at tumor subtype A vs. B, but it will evaluate at which *cutoff* the best split occurs. This makes decision trees highly flexible.

For example, the potential cutoff for patient age can be any year observed in the data set (say, 29-81 years old). The decision tree evaluates each possible cutoff (age < 30 vs age ≥

30, age 31 vs age ≥ 31,….age < 81 vs age ≥ 81) to see which cutoff splits the parent node (the full data set) to produce the best two daughter nodes.

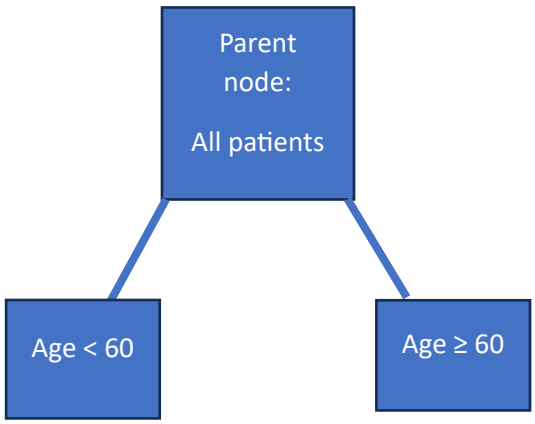

Fig 1. Example of a decision tree split.

How does the decision tree determine that using age at a cutoff at 60 years is the best way to separate tumor type A from tumor type B? Like most algorithms, "best" may be defined in different ways. Let's assume here that "best" means that we are finding a feature and split that produces two daughter nodes that are most *pure* in terms of their composition, with the goal of having all patients with subtype A in one node, and all patients with subtype B in the other node.

After the first split has been made, this same selection and splitting process continues independently with the two daughter nodes until the tree is fully grown and further splits do not improve the purity of the subsequent terminal nodes. Decision trees may be pruned or prevented from growing too large to avoid overfitting the training data set. The final tree assigns a predicted outcome (here, tumor subtype A or B) based on what the majority of the observations in that node actually are. Newly diagnosed patients can have a prediction generated by simply *running their data down the tree*, meaning that based

on their own particular features, they will end up in one of the mutually exclusive terminal nodes and the prediction associated with that terminal node.

If we look at our example again, let's see how this works:

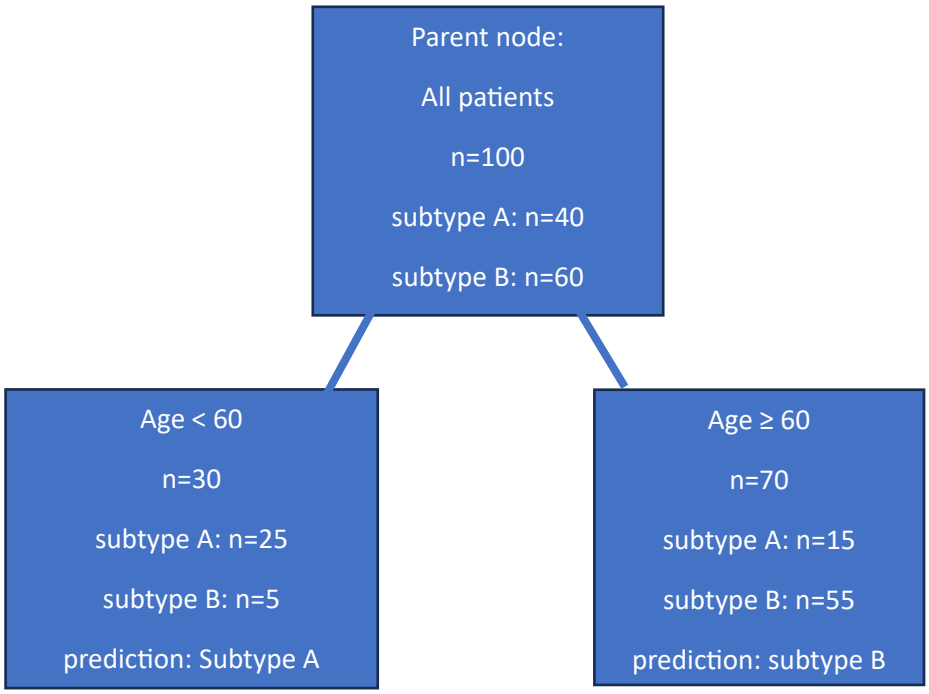

Fig 2. Predicting with a decision tree.

If we have a new patient age 71 who we want to predict a tumor subtype for we would use this very simple tree to simply put this patient in the right daughter node (all patients 60 and older). Here, the prediction is subtype B as 55/70 (78.6%) of patients in this node are subtype B.

Recall, we described the "best" split as the one that maximizes purity. So the above tree is more pure than just predicting every case to be of subtype B (as 60% of the sample is subtype B, using no other information this would be our best guess). The two daughter nodes created by splitting up cases by whether they are 60+ or under 60 in age have maximized purity. If we were to add another level to the tree by splitting each daughter node into two (thus obtaining a total of 4 terminal nodes), we would again look to maximize purity in those 4 terminal nodes to decide what variable and cut-point to use for the split. That best split may be another cut-point of age, or may be an entirely new predictor, such as biological sex.

There are many ways to compute purity, but for all of them the idea is to get as many subtype A cases in one node, and as many subtype B cases in the other node. We will focus on one purity measure called the *Gini index*. The Gini index can have a minimum value of 0 (pure) and a maximum value of 0.5 (not pure at all). We can compute the Gini index for **node 1**:

$\text{Gini index}_1 = 1 - \text{prop(subtype A in node 1)}^2 - (1-\text{prop(subtype B in node 1)})^2$

$= 1 - (25/30)^2 - (1-5/30)^2$

$= 1 - 0.694 - 0.0278 = 0.278$

Note that if we had had <u>only</u> subtype A in node 1 then Gini index would be equal to $1-(30/30)^2 - (0/30)^2 = 0$ and if we had an equal mix of subtype A and B (basically a non-informative node as it is a 50-50 mix) the Gini index would be equal to $1-(15/30)^2 - (15/30)^2 = 1-0.25-0.25 = 0.50$.

The Gini index for node 2 can be computed similarly and is equal to 0.337.

To obtain the Gini index value across the two terminal nodes, we would compute a *weighted average of the two Gini index values* we computed. The weight is based on the number of observations in each node:

Gini index of the split = 0.278 (30/100) + 0.337 (70/100) = 0.319. Note that this is purer than the Gini index of the parent node = $1 - 0.60^2 - 0.40^2 = 0.48$, which is why the decision tree split it into the two daughter nodes.

By selecting this variable and splitting at age 60, the decision tree has computed that no other variable and cut-point will produce two daughter nodes purer than the ones we get when we use age as our initial splitting variable and a cut-point of 60 years old.

When do we stop splitting the tree? This is a more complicated issue but in the extreme we could technically keep splitting until there is no further improvement in purity. Such a large tree would probably separate the data set at hand well, but if we tried to generalize to a different data set of new patients we would likely find that it will not perform well.

Think of it like this: if I had a group of people in a room and I wanted to classify those who drive a large SUV vs. those who drive something else, I could potentially find a tree that splits not only on number of kids the person has and their profession, but also on birthday month and favorite food (pizza vs not pizza). Is that because being born in the summer and liking pizza is predictive of SUV ownership? No, but in my small sample of people it may split car vs large SUV owners simply *by chance* – a spurious association. In my original room of people the Gini index of such a tree (**Fig. 3**) will be equal to 0 (maximum purity). But if I tried to predict ownership in the next room of people how do you think this tool would perform?

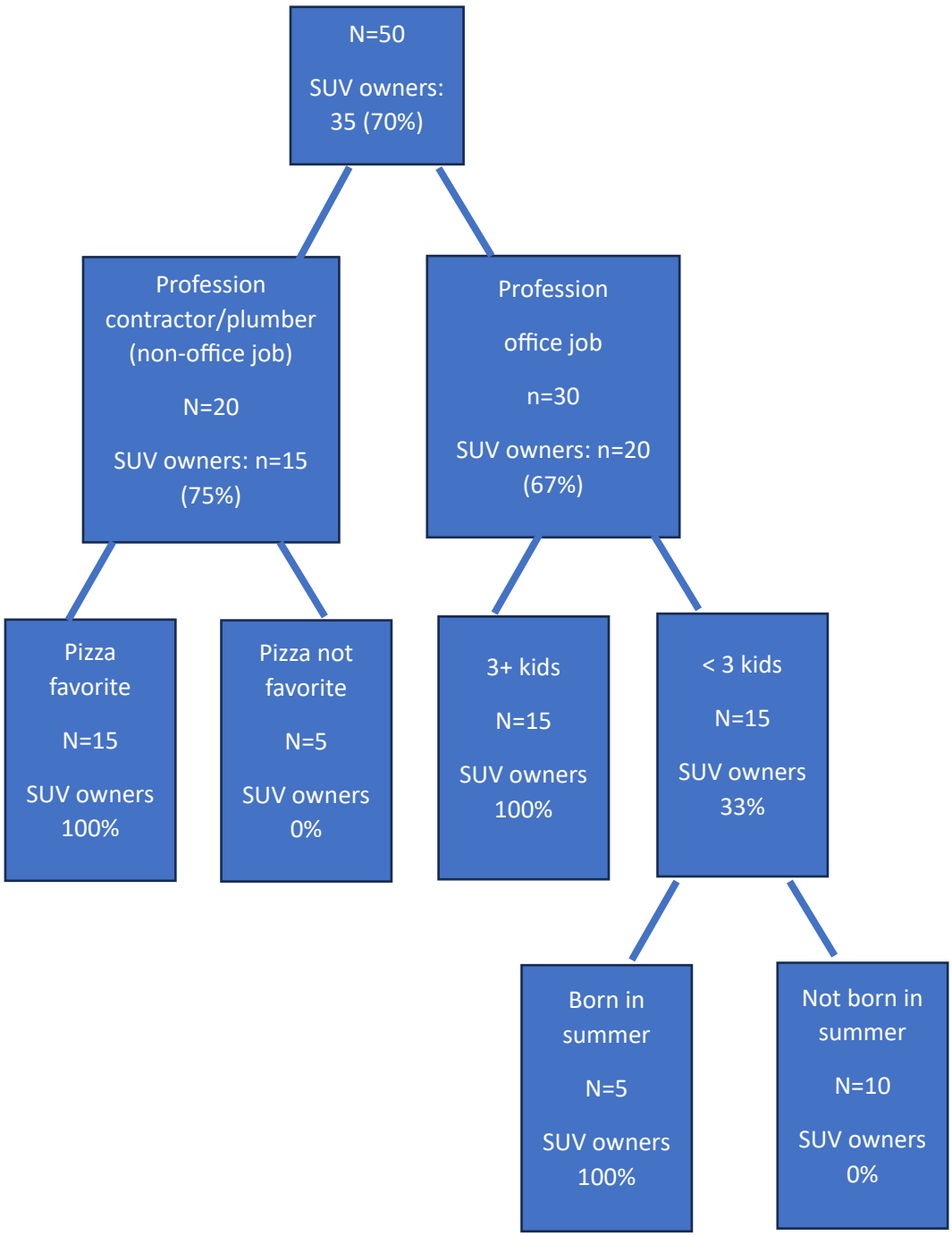

Fig 3. An overfit decision tree.

If I had data from a new group of people and ran them down the tree to predict whether they drive large SUVs, it is very likely that we would misclassify a good proportion of people because we know that liking pizza and being born in summer are probably not very predictive of SUV ownership, or even related to things that are.

How and when to stop splitting the tree (or pruning an overgrown tree) will not be discussed in this tutorial, but more information may be found **here**. For now, it is important to understand that while we want the trees to fit our data well, we can see how in the SUV example how growing a large tree would probably not generalize very well. And the point of building a predictive tool is to predict new observations accurately.

In general, decision trees are highly flexible and easy to use. However, their drawback is that they tend to **overfit** the data at hand unless carefully pruned or stopped before they get too large and start splitting on variables with spurious association to outcome. Relatedly, overfit decision trees are considered **high variance** classifiers, meaning that with different data sets sampled from the population (all assumed to arise from the same underlying true model) the resulting trees and predictions will vary a lot because large unpruned decision trees tend to overfit to the data at hand.

Now that we understand how and why a decision tree is created, its strengths, and its drawbacks, we will now examine what Random Forest is doing to improve how decision trees perform.

Random Forest:

Random Forest is an **ensemble** of typically 500-100 decision trees. To build each tree to be as independent as possible, Random Forest perturbs the data set by **bootstrapping** (i.e.: randomly samples, with replacement, members of the original data set so you end up with the same size data set but consisting of a constantly perturbed version of the data), and it also **randomly selects only a subset** of the total available features to be evaluated at each node as potential splitters. We can predict whether a new patient will have tumor subtype A by getting the prediction from each decision tree and allowing the final prediction to be based on the proportion of trees that predict subtype A (if 800/1000 trees all predict subtype A then the final prediction will be that the patient has subtype A). The benefit of Random Forest is that by counting the "votes" of each member tree, the prediction tends to be much more stable than the prediction from any single decision tree.

Let's break down some of this to further understanding:

Bootstrapping: We have a training set that consists of n individuals. When we bootstrap this set, we are sampling n individuals from this set *with replacement* – so for a particular bootstrap sample some people will be in the sample more than once, and others (about 1/3 of the data) not in the sample at all. When we bootstrap many times, we are essentially creating multiple new training set sets that are similar, but all slightly perturbed from one another and from the original data set. This allows Random Forest to build a set of different trees. After all, if we just had our one training data set, every tree fit to the data would be exactly the same!

Randomly selecting a subset of predictors as candidates for each split: Recall that every time we attempt to split a parent node into 2 daughter nodes in a decision tree, we evaluate all variables at all possible cut-points. Random Forest only selects a random sample of predictors that the tree can choose from (instead of the entire set of predictors). Why does it do this? Because, like bootstrapping, it is another way to allow the resulting trees to be as different from one another as possible.

How does Random Forest improve prediction compared to a single decision tree?

Recall, decision trees are flexible and tend to find patterns in the data well, but also (and because of their flexibility and tendency to grow very large) they are high variance classifiers. By taking all these high-variance trees and averaging across predictions, the resulting classification tends to be much lower variance. It is the same reason a doctor may take your blood pressure 3 times at the same visit to get an average reading – *averaging smooths over the variation in BP measurements*.

In summary, Random Forest creates an ensemble of decision trees that are forced to be as different as possible through bootstrap sampling and random predictor selection. **It is this averaging across trees that makes Random Forest so successful at reducing the high variance of a decision tree.**

Where can I read more about these different classifiers?

There are many **excellent online resources** that will allow you to more fully understand machine learning and the various algorithms used.