**# Predicting Stock Movement with Random Forest**

Random Forest is a machine learning ensemble method that is widely used because of its flexibility, simplicity, and often quality results. In this tutorial, we will use the Random Forest algorithm to build a classification model that will help us predict whether a stock will close higher or lower based on range of technical indicators.

**## Background: What is Random Forest?**

Now, up above I gave you a simple definition of what Random Forest is. However, I think it's important to delve in more detail about it before we start coding our model. Now in short, Random Forests is a supervised machine learning algorithm that uses multiple decision trees in aggregate to help make more stable and accurate predictions.

Decision Trees are the building blocks of Random Forest, so to understand Random Forest we first need to understand Decision Trees.

**### Understanding Random Forest: Decision Trees**

Every day of your life, you make decisions. Your decisions can range from what to eat for lunch to what time you leave for work to avoid traffic. However, have you ever taken a step back to understand how you arrive at the decisions you make?

Sometimes you probably do but other times it feels like auto-pilot and you just do it. Well, normally, most decisions are based on a set of rules that you follow to arrive at the decision. For example, if I am hungry, I might have a set of rules I used to determine what I’ll eat. Let’s list my rules:

1. Rule 1: Determine if I am hungry.
2. Rule 2: Determine if I have money.
3. Rule 3: Determine if I want something light or heavy.

If you notice, we flow from one rule to next. In other words, I first must determine if I am even hungry, if I am then I can proceed to rule number 2 and determine if I have money. There is a flow from one rule to the next.

Additionally, I can further specify my rules by adding criteria to them. For example, at rule number 2 I can have multiple possible levels of having money $50.00, $25.00, or $10.00. Then based on that I can then move to rule number 3.

Let’s walk through a simple example. First, I determine I’m hungry, so I then proceed to rule 2. I determine I have $10.00 to my name, so I can move to rule 3. After determining I have $10.00, I decide I want something light, so I decide to go buy soup.

Well if you were able to follow all of that you need have the foundation of a Decision Tree. **Decision Trees is a flowlike chart structure where each node of the tree is used to test an attribute of the object**.

The node in our example up above would represent one of the rules we define. The attribute would be the value we were testing. For example, I tested my net worth (the amount of money I had) and I also tested my hunger level. The object we are testing is me! I’m the one who get’s tested on all the different rules.

**### Understanding Random Forest: Decision Trees Components**

To provide a more formal definition of everything, let’s walk through some points again. The first thing I want to mention is that for Decision Trees there are 3 kinds of nodes and 2 kinds of branches.

***Decision Node:***

Whenever I have to make a choice, we will define this as a “Decision Node”, so, for example, one decision node is testing whether I have money or not. A decision node is a point in the tree where a choice must be made.

***Root Node:***

The starting point of our tree is called the “Root Node”, so in our example up above determing if I am hungry or not is the root node. The root node represents the entire population or sample which is then divided into two or more homogenous sets.

***Leaf/Terminal Node:***

When we reach the end of our tree and we don’t have to make anymore decisions we have reached a “Leaf Node” also called a “Terminal Node”. In my example up above, getting soup would represent a terminal/leaf node. Nodes that do not split are Leaf/Terminal nodes.

***Decision Branches:***

The branches extending from a Decision Node are called “Decision Branches” and represent one of the possible alternative paths we can take. It’s very important to understand that each path must be mutually exclusive, in other words, if I take one path I can’t take the other. Additionally, they are collectively exhaustive, in other words, I’ve provided all possible paths.

***Splitting:***

The process of dividing a node into two or more sub-nodes, for example we split on net-worth.

***Parent and Child Node:***

A node, which is divided into sub-nodes is called parent node of sub-nodes whereas sub-nodes are the child of parent node. In our example up above, a parent node would be “determining net worth” and our child would be the balance of my net worth.

**### Understanding Random Forest: How Decision Trees Work**

Decision Trees work by first “Splitting” the data into subsets. The splits are formed on a feature of the data set.

**### How Decision Trees Work: How is a Split Determined?**

The goal is to split/partition the data until each node is homogenous in data and has as little "impurity" as possible. We will cover what impurity is later in the document.

**### How Decision Trees Work: How is Impurity Calculated?**

There are multiple functions you can use to calculate impurity but one of the most commonly used ones is the **Gini Index**. However, there are also other ways to calculate impurity to name a few we have Bayes Error and the Cross-Entropy Function. In our model, we will be using the Gini Index.

**### How Decision Trees Work: Impurity & Calculating the Gini Index**

Up above I mentioned that decision trees work by splitting the data into subsets and that they’ll keep splitting the data until the data is homogenous (the same) and has as little impurity as possible. Well, what exactly is impurity?

Impurity refers to the probability of a variable being wrongly classified when randomly chosen. Again, just to reiterate, impurity refers the PROBABILITY of a VARIABLE BEING WRONLY CLASSIFIED. To better understand this concept, let’s start with a PURE dataset.

If all the data belongs to a single class, we would call it pure. For example, imagine I have a dataset that contains the breed labels of different dogs. If all the labels are the SAME BREED, then it’s considered pure. Why you may ask? There isn’t a chance to classify it to any other class if there all Labrador retrievers than that’s all it’s going to be. There isn’t a chance to possibly be labeled as a Pug!

The Gini index or sometimes called Gini impurity, measures the degree or probability of a variable being wrongly classified when it is randomly chosen.

The degree of Gini index varies between 0 and 1, where 0 denotes that all elements belong to a certain class or if there exists only one class, and 1 denotes that the elements are randomly distributed across various classes. A Gini Index of 0.5 denotes equally distributed elements into some classes.

Here is a simple table for you to remember the values:

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0.0 | A pure data set. All values belong to a certain class or 1 class. |
| 0.5 | All values are EQUALLY distributed among classes. |
| 1.0 | All values are RANDOMLY distributed among classes. |

The formula for the Gini Index is as follows:

Where pi is the probability of an object being classified to a particular class. I do want to cover the concept of Entropy as this point. Entropy is just another method for determining the split in a decision tree. However, it is calculated slightly differently. Here is the formula:

Both methods use the idea of Information gain when it comes to determining the optimal value. Information Gain in this context refers to the difference in information before splitting the parent and information after splitting the child. It should be noted that different impurity measures (Gini index and entropy) usually yield similar results.

**### How Decision Trees Work: Splitting in Detail**

We know that decision trees work by splitting the data so that the Gini Index is minimized, remember lower is better in this case. The Gini Index, serves as the metric we use to determine the split, so how is it used? Imagine we have a dataset and I choose a random spot to split the data.

We now have two impurity measures, one for the left side and one for the right side. We repeat this process many times of splitting the data and then pick the best (lowest impurity) one.

**### How Decision Trees Work: Pruning**

Great, we have a decision tree, and we’ve determined the number splits and we’ve built our decision tree. The next step is the “Pruning” process. Pruning involves reducing the size of the Decision Tree by converting certain branches into leaf nodes and removing leaf nodes.

You may be asking, “Why would we want a smaller decision tree?”. The reason is simple, Decision Trees have an easy time over fitting the data. In fact, the larger the tree the more likely this is to happen. Generally, a simpler decision tree avoids over-fitting the data. Additionally, we may find that pruned decision trees provide relatively the same level of accuracy. That means we almost get the entire picture for 70% of the cost.

### **Understanding Random Forest: Advantages and Disadvantages of Decision Trees**

Here are some disadvantages and advantages of decision trees:

**Advantages:**

1. Intuitive in nature.
2. Have value even with little hard data.
3. Help determine worst, best and expected values for different scenarios.
4. Use a white box model. If a given result is provided by a model.
5. Can be combined with other decision techniques.
6. The number of hyper-parameters to be tuned is almost null.

**Disadvantages:**

1. Instability: Even small changes to the input data can have dramatic changes to the overall structure of the decision tree.
2. They are often relatively inaccurate. Many other predictors perform better with similar data.
3. For data including categorical variables with different number of levels, information gain in decision trees is biased in favor of those attributes with more levels.
4. Calculations can get very complex, particularly if many values are uncertain and/or if many outcomes are linked.

**### Understanding Random Forest: Bootstrap Aggregation**

One of the challenges with decision trees is it’s hard to model a complex problem with a single decision tree. What was discovered is that we can use multiple decision trees to help make more accurate predictions with models that are complex in nature.

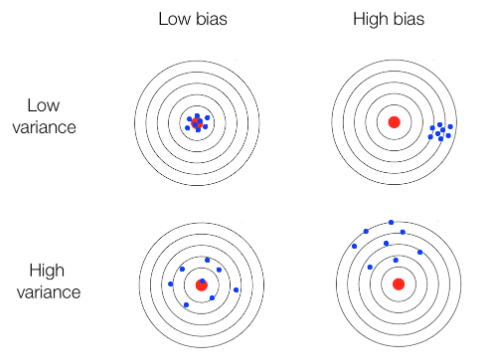
The idea of using the results of many models, aggregating them, and using the output to make a prediction refers to the concept of ensemble learning. An Ensemble Learning model is a model in which decisions are used from MULTIPLE MODELS to improve the overall performance of the model. The old idea that 2 minds are better than 1, perfectly summarizes ensemble learning. We use the results of multiple models to get a better idea of what the true answer is.

Bootstrap Aggregation, also called Bagging, is a machine learning ensemble algorithm used to help improve the stability and accuracy of predictions. Bootstrap Aggregation seem to work well for high-variance, low-bias procedures, such as trees. How Bootstrap Aggregation works, depends on the type of problem we are modeling.

For regression, we simply fit the same regression tree many times to “bootstrap-sampled” versions of the training data and average the result. For classification, a committee of trees each cast a vote for the predicted class. While Bootstrap Aggregation reduces the variance of the base learner it has limited effect on the bias.

Why does Bootstrap Aggregation reduce the variance? Well, consider we have a single classifier that is unstable, in other words the variance is high. With Bootstrap Aggregation, the process of repeating a process multiple times and averaging those results is what we can think of as an approximation of the true average.

Why does Bootstrap Aggregation have little impact on the bias? Imagine we have many models that have either low bias and we aggregate their results all we are doing is compressing that variance closer together. What that means is if we have low bias, we still maintain it. Additionally, if we have high bias, we still maintain it. Here is a visual to help you understand:



In the lower right-hand corner, we have high variance and high bias. Say we perform bootstrap aggregations and we take the results of multiple high bias model all we are doing is compressing the variance so we closer to the average of the data points. Hence, we maintain the bias but only reduce the variance.

**### Understanding Random Forest: Bootstrap Aggregation Flaws**

1. Bagging algorithm still uses Decision Trees which we know have issues.
2. Decision Trees uses Gini-Index, a greedy algorithm to find the best split. Greedy meaning, the very best split point is chosen each time
3. We end up with trees that are structurally similar to each other. The trees are highly correlated among the predictions. Random Forest addresses this problem

**### Understanding Random Forest: Why Use it?**

We saw Decision Trees have some disadvantages, so we used Bootstrap Aggregation to help improve those weaknesses, but we also saw it has some disadvantages as well. Is there a way to improve Bootstrap Aggregation? We could use Random Forest!

What exactly does Random Forest do to fix these issues? **Well, it uses Bootstrap Aggregation but it what it does instead is to create trees that have no correlation or weak correlation with each other**. You’re probably confused at this point because it doesn’t sound like anything really changed. We are still using Bootstrap aggregation. However, the key here is what we are aggregating before we used HIGHLY CORRELATED trees, with Random Forest we devise a way to create WEAKLY CORRELATED OR UNCORRELATED trees. What Random Forest tries to do is avoid over-fitting of the data and improve the stability and accuracy of predictions.

**### Understanding Random Forest: How it works?**

With Random Forest we are still doing a lot of the same things, we are still building our forest (creating many trees like we did in bootstrap aggregation) and we are still taking a vote of the trees, at least for classification problems. The key here is how we build the tree. We build a tree doing the following steps:

1. Take a random sample of size N **with replacement** from the data.
2. Take a random sample **without replacement** of the predictors.
3. Construct the first CART partition of the data.
4. Repeat Step 2 for each subsequent split until the tree is as large as desired. Do not prune.
5. Repeat Steps 1–4 many times (e.g., 500).

**It’s important to understand that step 2 & step 1 is how Random Forest deals with the highly correlated trees generated by bagging**. That’s the magic that goes on behind the scenes that improves the model!

For step 5, I want to be a little clearer on that. It’s important to realize that there is no magic number for “many times” it depends on your data. Best advice is to evaluate the model and use cross validation.

When the training set for the current tree is drawn by sampling with replacement, about one-third of the cases are left out of the sample. This OOB (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance.

After each tree is built, all the data is run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees.

Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data.

**### Closing Notes**

With that we have reached the end of our overview of Decision Trees, Bootstrap Aggregation and Random Forest. Now with a high-level overview we can begin the process of building our model.