[Machine Learning]

Why Choose Random Forest and Not Decision Trees

A concise guide to Decision Trees and Random Forest.

Decision trees belong to the family of ***the supervised classification algorithm***. They perform quite well on *classification* problems, the decisional path is relatively easy to interpret, and the algorithm is fast and simple.

The ensemble version of the Decision Trees is the Random Forest.

Table of Content

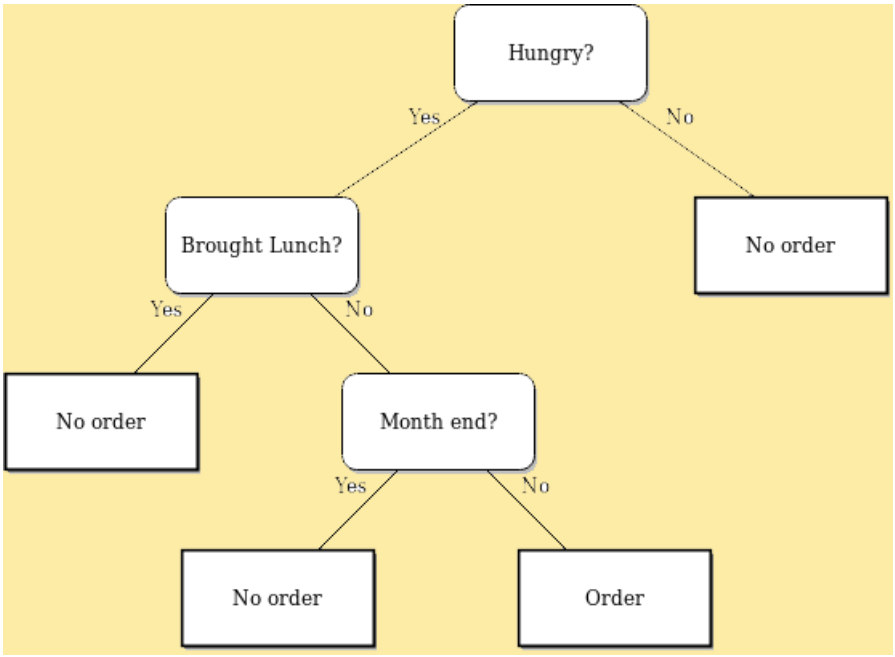
Decision Trees

Introduction to Decision Trees

A decision tree is a simple **tree-like structure** constituting nodes and

branches. At each node, data is split based on any of the input features, generating two or more branches as output. This iterative process increases the numbers of generated branches and partitions the original data. This continues until a node is generated where all or almost all of the data belong to the same class and further splits—or branched—are no longer possible.

This whole process generates a **tree-like structure.** The *first splitting node* is called the *root node*. The end nodes are called leaves and are associated with a class label. The paths from the root to the leaf produce the classification rules.



Suppose, you're an employee and you want to eat processed food.

Your course of action will depend on several circumstances.

If you aren't hungry, you won't spend money on junkies. But if you are hungry then the choices are changed, your next move depends on your next circumstance i.e. have you bought lunch or not?

Now if you don't have lunch, your action will solely depend on your next pick i.e. is it month-end or not? If it will be the last few days of the month, you will consider skipping the meal otherwise you won't take it as a preference.

Decision Trees come into play when there are several choices involved to arrive at any decision. Now you must choose accordingly to get a favorable outcome.

Tree-based learning algorithms are considered to be one of the best and mostly used **supervised learning** methods. Tree-based methods legitimize predictive models with better *accuracy, stability, and ease of interpretation*. Unlike contemporaries, they work well on non-linear relationships as well. Decision Tree algorithms are referred to as **CART (Classification and Regression Trees)**.

How do Decision Trees work?

There are two components of Decision Trees:

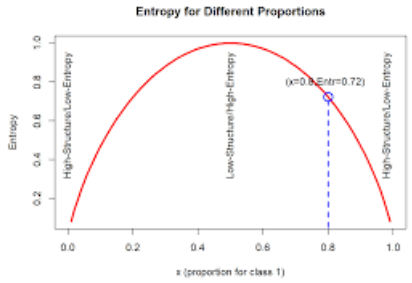
- **Entropy**—It is regarded as the **randomness of the system**. It is a measure of node purity or impurity.

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Playing football	
Yes	No
9	5

$$\text{Entropy}(\text{Playingball}) = \text{Entropy}(5,9)$$
$$= \text{Entropy}(0.36, 0.64)$$
$$= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64)$$
$$= 0.94$$

Entropy is maximum when p = 0.5 i.e. both outcome has the same favor.



- **Information gain**—It is a **reduction in entropy.** It is the difference between the uncertainty of the starting node and weighted impurity of the two child nodes.

Information Gain = Entropy(parent) – [Average entropy(children)]

$$\text{Entropy} = \sum_{i} -P_i(\log_2 P_i)$$

P_i is probability of class i

It helps us to find the root node for our decision tree, the node with maximum **Information Gain is regarded as the root node as it**

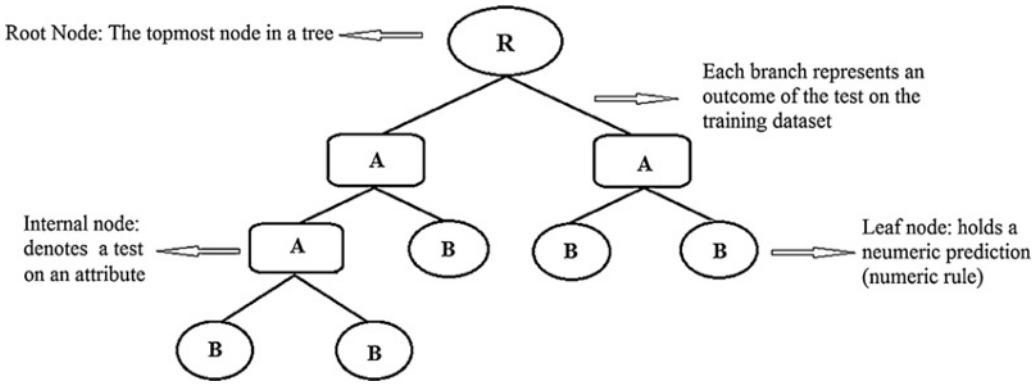
has maximum uncertainty.

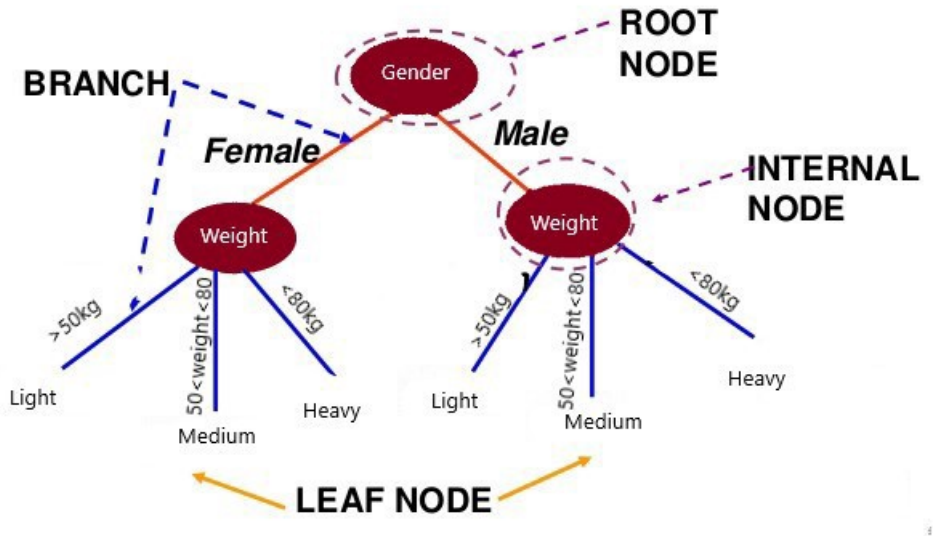
We will first split the feature with the **highest information gain**. This is a recursive process until all child nodes are pure or until the information gain is zero.

***Goal of Decision Tree**: Maximize Information Gain and Minimize Entropy*

Let's say we have a sample of 60 students with three variables *Gender* (Boy/ Girl), *Class* (XI/ XII), and W*eight* (50 to 100 kg). 15 out of these 60 play football in leisure time. Now, we want to create a model to predict who will play football during free time? In this problem, we need to divide students who play football in their leisure time based on a highly significant input variable among all three.

This is where the decision tree comes into play, it will classify the students based on all values of three variables and identify the variable, which creates the best homogeneous sets of students.





Using Decision Tree, we are easily able to solve our problem and classify students based on traits that whether they will prefer playing football in their leisure time or not?

Decision Trees Implementation from scratch

```
def divide_data(x_data, fkey, fval):       #dividing data in left and right
    x_right = pd.Dataframe([], column = x_data.columns)
    x_left = pd.Dataframe([], column = x_data.columns)  #creating empty dataframes
    for ix in range(x_data.shape[0]):
        val = x_data[fkey].loc[ix]            #fkey = name of column
        if val>fval :                         #fval = threshold value i.e. node root of successive DT
            x_right = x_right.append(x_data.loc(ix))
        else:
            x_left = x_left.append(x_data.loc(ix))
    return x_right, x_left
```

Dividing data in different parts

```
import numpy as np
import pandas as pd
def entropy(col):  # calculating entropy
    counts = np.unique(col, return_counts = True)
    N = float(col.shape[0])
    ent = 0.0
    for i in counts[1]:
        p=i/N
        ent+= (-1.0*p*log2(p))
    return ent
```

Calculating Entropy

```
def information_gain(x_data, fkey, fval):
    left,right = divide_data(x.data, fkey, fval)     #getting total samples of left and right
    l=float(left.shape[0])/x_data.shape[0]
    r=float(right.shape[0])/x_data.shape[0]
    if left.shape[0]==0 or right.shape[0]==0:
        return -100000
    i_gain = entropy(x_data.survived)- (l*entropy(left.survived))+ (r*entropy(right.survived))
    return i_gain
```

Calculating Information Gain

Sci-kit Learn implementation

```
from sklearn.tree import DecisionTreeClassifier
sk_tree = DecisionTreeClassifier(criterion = "entropy")
sk_tree.fit(train_data[input_cols], train_data[output_cols])     #training our model
sk_tree.predict(test_data[input_cols])          #predicting the class
sk_tree.score(test_data[input_cols], test_data[outpu_cols])    #getting score
```

Visualizing your Decision Tree

```
[ ]: import pytdotplus
     form sklearn.externals.six import StringIO
     from IPython.display import Image
     from sklearn import export_graphviz

     dot_data = StringIO()
     export_graphviz(sk_tree, out_file = dot_data, filled=True, rounded=True)
     graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
     Image(graph.create_png())
```

Pros & Cons of Decision Trees

Pros

- Easy to interpret
- Handles both categorical and continuous data well.
- Works well on a large dataset.

- Not sensitive to outliers.
- Non-parametric in nature.

Cons

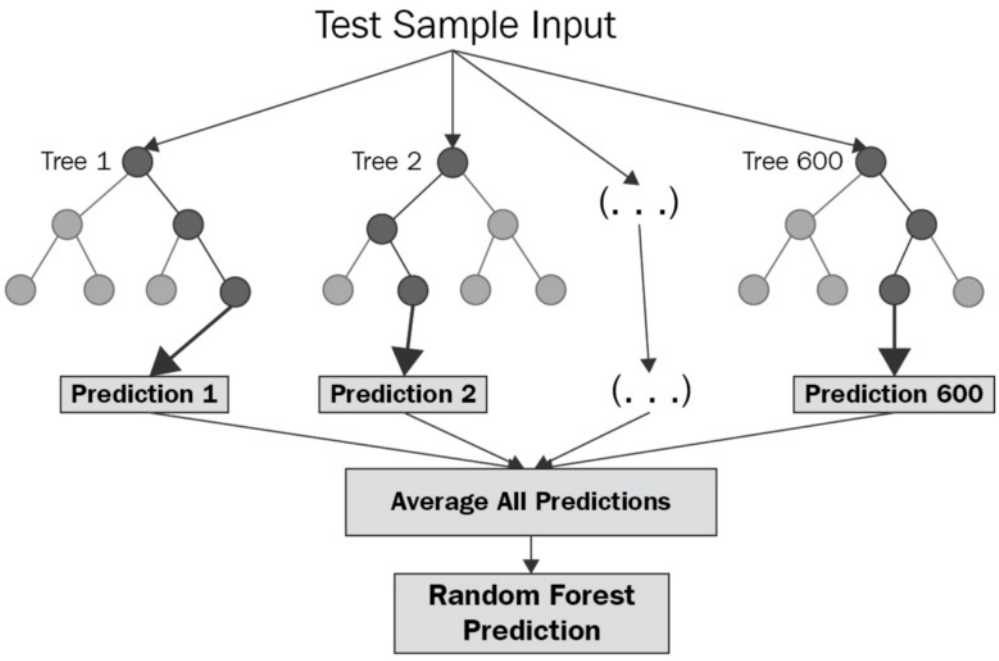
- These are prone to overfitting.
- It can be quite large, thus making pruning necessary.
- Can't guarantee optimal trees.
- It gives low prediction accuracy for a dataset as compared to other machine learning algorithms.
- Calculations can become complex when there are many class variables.
- High Variance(Model is going to change quickly with a change in training data)

Random Forest

Introduction to Random Forest

Random forest is yet another powerful and most used **supervised learning** algorithm. It allows quick identification of significant information from **extremely large datasets.** The biggest advantage of Random forest is, it relies on **the collection of various decision trees** to arrive at any solution.

This is an **ensemble algorithm** that takes into account results of more than one algorithms of the same or different kind for classification.



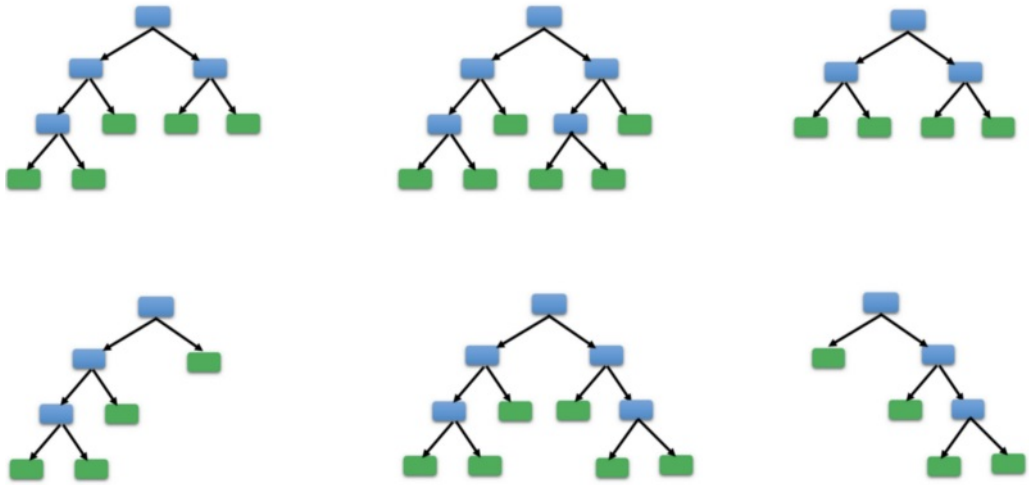
Random Forest, Source

Suppose you want to go for a vacation but baffled about the destination. So you decide to ask your friend *Lakshay* for advice. Lakshay will ask you for your last vacation and how did you like it. If not, what did you do there. To get precise results he might even inquire about your preferences and based on your remark he will provide you a recommendation. Here, Lakshay is **using the Decision Tree technique** to provide you feedback that is based on your response.

But you think Lakshay's advice is a bit biased and you asked Meghna(your other friend), the same question. She too came up with a recommendation but you again considered it to be a dicey choice. You iterated this process and asked "n" friends, the same question, now you're up to some common places recommended by your friends. You collect all the votes and aggregate them. You decide to go to the place with the most votes. **Here, you are using a random forest**

technique.

In Decision Tree, **the deeper you go, the more prone to overfitting you're** as you are more specified about your dataset. So Random Forest tackles this by presenting you, the product of Decision Tree's simplicity and Accuracy through Randomness.

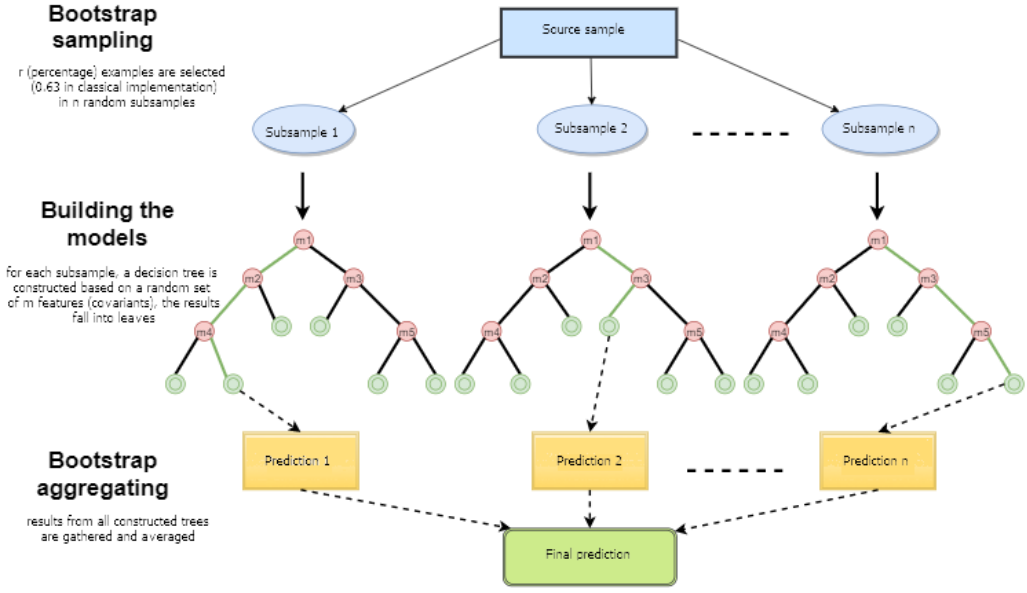


Random Forest = Decision Tree's simplicity * Accuracy through Randomness

How does Random Forest Works?

Assume "*m*" features in our dataset:

1. Randomly chose "*k*" features satisfying condition $k < m$.
2. Among the *k* features, calculate the root node by choosing a node with *the highest Information gain*.
3. Split the node into child nodes.
4. Repeat the previous steps *n* times.
5. You end up with a forest constituting *n* trees.
6. Perform *Bootstrapping* i.e. combining the results of all Decision Trees together.



Source

Sci-kit implementation for Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimator = 10, criterion = "entropy", ma
rf.fit(X_Train, Y_Train)
rf.score(X_Train, Y_Train)
rf.score(X_Train, Y_Train)
y_pred = rf.predict(X_Test)

#n_estimator = number of trees used in Random Forest(hyperparameter)

```

random forest hosted with ❤ by GitHub view raw

Pros & Cons of Random Forest

Pros:

- Robust to outliers.
- Works well with non-linear data.
- Lower risk of overfitting.
- Runs efficiently on a large dataset.
- Better accuracy than other classification algorithms.

Cons:

- Random forests are found to be biased while dealing with categorical variables.
- Slow Training.
- Not suitable for linear methods with a lot of sparse features

---

Conclusion

Hopefully, this article will help you to understand about Decision Trees and Random Forest in the best possible way and also assist you to its practical usage.

As always, thanks so much for reading, and please share this article if you found it useful!

Feel free to connect:

LinkedIn ~ https://www.linkedin.com/in/dakshtrehan/

Instagram ~ https://www.instagram.com/_daksh_trehan_/

Github ~ https://github.com/dakshtrehan

Follow for further Machine Learning/ Deep Learning blogs.

Medium ~ https://medium.com/@dakshtrehan

Want to learn more?

Detecting COVID-19 Using Deep Learning

[The Inescapable AI Algorithm: TikTok]

[Why are YOU responsible for George Floyd's Murder and Delhi Communal Riots?]

[Clustering: What it is? When to use it?]

[Start off your ML Journey with k-Nearest Neighbors]

[Naive Bayes Explained]

[Activation Functions Explained]

[Parameter Optimization Explained]

[Gradient Descent Explained]

[Logistic Regression Explained]

[Linear Regression Explained]

[Determining Perfect Fit for your ML Model]

Cheers!

By Daksh Trehan on July 2, 2020.

Canonical link

Exported from Medium on July 15, 2020.