Random Forests

The Story of Jones and His High MPG Cars . . .

Jones is a car collector who has moved to a new planet and is looking for cars to collect. In choosing which car he wants, his only desire is on one thing: quickly buying new cars which have an excellent fuel efficiency (How many miles the car can run on one gallon of fuel). Unfortunately, this planet is without easily accesible data on cars fuel efficiency and without used car salesmen to easily give us all the details on the cars at the dealership. We'll have to develop an algorithm to quickly estimate the cars miles per gallon.

After spending some time looking at algorithms, Jones has found that the decision tree is a great way to estimate the cars fuel efficiency given a features readily available at the dealership, such as # of engine cylinders, model year, and weight.

For example, after going between a few dealerships, he finds that first marking if the cars' weights are under 3000 pounds, then checking if their model year is newer than 1980, then if they have less than 5 cylinders, and finally if the displacment is greater than 3000 cubic centimeters. Which works to a find solid choices, but he often finds that trying to create his own decision tree to split up cars he desires from only his observations leads to overcomplex sequences of decisions that don't generalize well between different car dealerships.

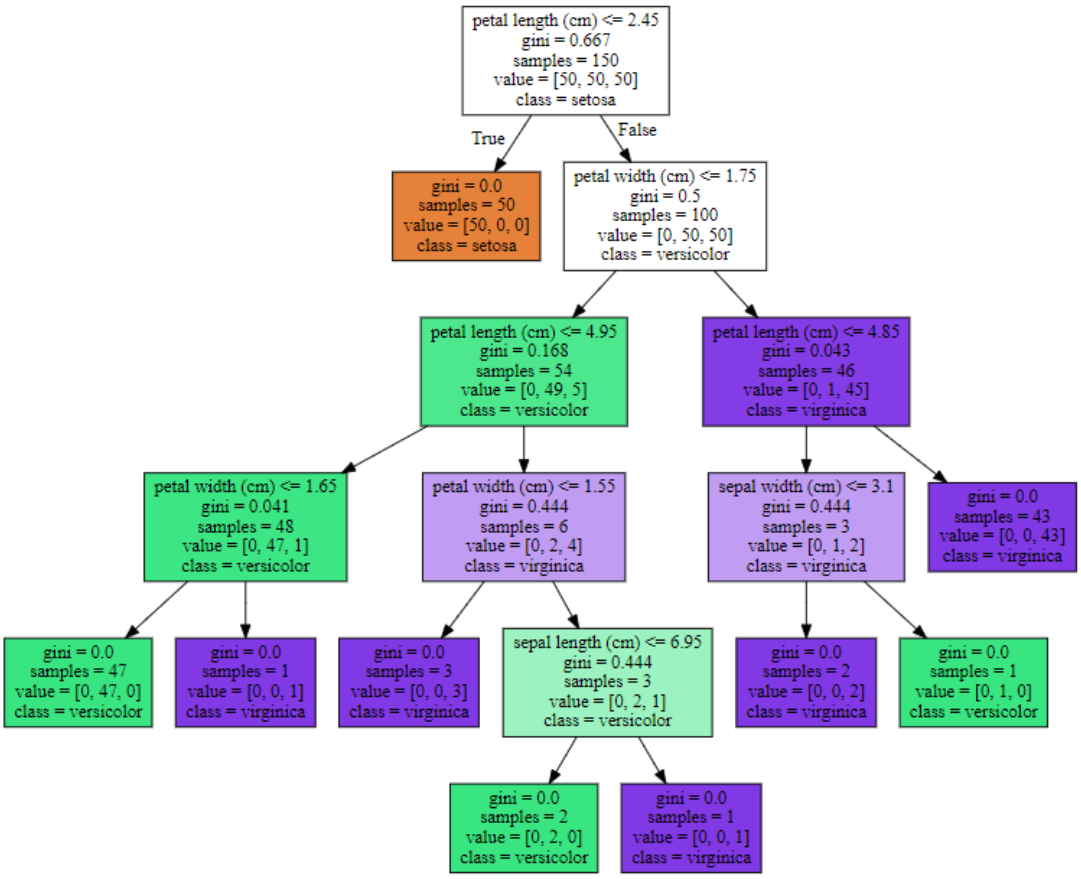
He realizes that if he enlists his henchmen to develop their own decision trees for each car dealership and everyone average their results, that should work better than solely doing it alone.

This is the intuition behind the advantage of Random Forests over Decision Trees.

We've covered decision trees before in the previous note, but let's review a few of the mechanics of a decision tree and see how they compare to Random Forests.

Decision Trees

Before Jones thinks about enlisting help, he'll first ensure that his own methods for splittig up lot into cars he wants are ideal.



Remember a Decision Tree is a model which works to predict whether items in the dataset belong to different classes or regression values by iteratively splitting up the dataset (in the case of Scikit-Learn) based on quantitative decisions.

Entropy

Remember that entropy is a measure of homogeneity within an existing dataset split among a set of classes or the .

Say we had a dataset of 100 cars which are either red or green.

In what distributions of red/green cars would the dataset be completely homogeneous?

- All 100 Cars Green: Completely Homogeneous
- All 100 Cars Red: Completely Homogeneous

In what distribution would the dataset be completely non-homogenous? Or more simply put, as mixed as possible?

- 50 Cars Green, 50 Cars Red

Entropy Formula

$$Entropy(Z) = -(\sum_{i=0}^{c} p_i * \log_2(p_i))$$

- *Z* is the input dataset
- *C* is the number of classes
- p_i is the probability of picking a datapoint with class i.

Scenario 1: 70 Red, 30 Green

$$-(710 * \log_2 710 + 310 * \log_2 310) = .88$$

Scenario 2: 0 Red, 100 Green

$$-(010 * \log_2 010 + 1010 * \log_2 1010) = 0$$

Scenario 3: 50 Red, 50 Green

$$-(510 * \log_2 510 + 510 * \log_2 510) = 1$$

Gini Impurity Index

The Gini Impurity Index also provides another way of assessing homogeneity of dataset given a set of data points within certain / the quality of a split if the dataset were to be split among those classes.

What does the Gini Impurity Index have to say about homogeneity and non-homogeneity? Let's input a few input a few scenarious from our 100 cars dataset above and find out.

$$\text{Gini}(Z) = \sum_{i=0}^{C} p_i * (1 - p_i)$$

- *Z* is the input dataset
- *C* is the number of classes
- p_i is the probability of picking a datapoint with class i.

Scenario 1: 70 Red, 30 Green

$$710 * (1 - 710) + 710 * (1 - 710) = .42$$

Scenario 2: 0 Red, 100 Green

$$010 * (1 - 010) + 1010 * (1 - 1010) = 0$$

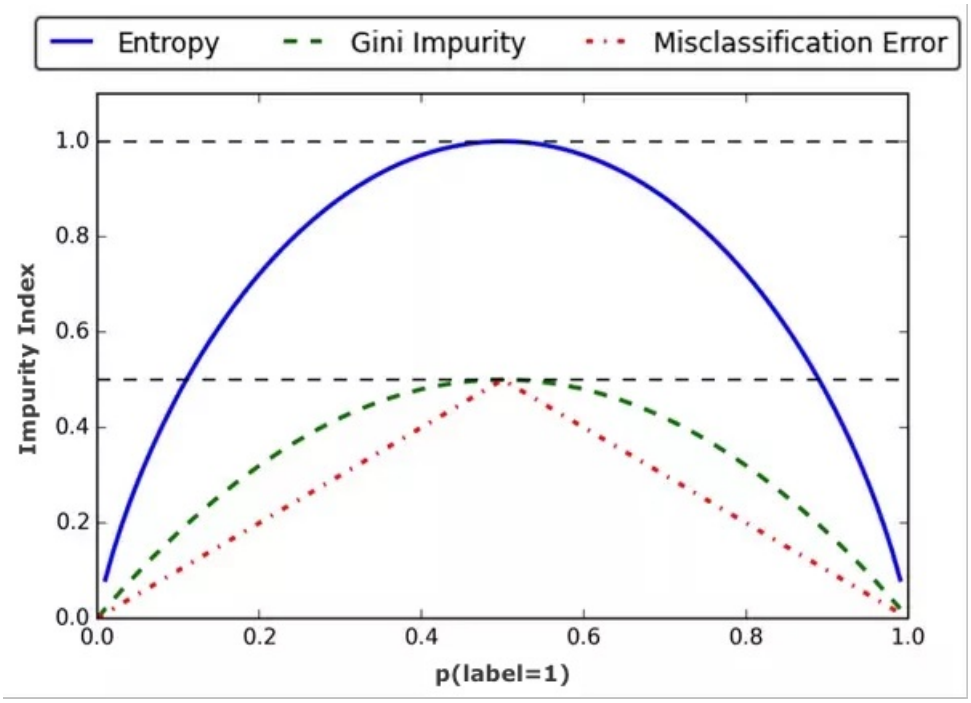
Scenario 3: 50 Red, 50 Green

$$510 * (1 - 510) + 510 * (1 - 510) = .5$$

Here we can see that the values of Gini Impurity reach a max value of .5 rather than 1 in Entropy.

Remember that these probilities for each class p_i all sum up to 1.

If we were to randomly pick an item in our dataset and then randomly classify it according the class distribution of our dataset, the *Gini Impurity Index* measures the proablity we would correctly classify this data point.



Issues Within Decision Trees

Decision trees can create models that do not generalize well due to overcomplex classification. This can be solved by pruning, setting the maximum search depth, and minimum number of samples needed at a leaf node.

- For Jones, this means he could get rid of the final decisions made when asessing cars, stopping after make a certain number of decisions, or stopping when his splits create a group of cars that have a minimum count.

Small tweaks in the dataset can create widely different trees.

- Slightly different car lots could result in drastically different decision trees for Jones.

The most widely applicable decision tree modeling algorithms rely on greedy heuristic algorithmsn (where the optimal decision is made locally at each node without contemplating how this may affect decisions in the future). Algorithms which rely on greedy heuristics cannot be guarenteed to return the globaly optimal tree.

- Since Jones' decision process relies on greedily making the best decision at each split, rather than weighing his past choices down the line, it's not guarenteed he creates the ideal decision tree for that car lot.

The Overfitting Problem

A model which is highly flexible (high variance) will be able to very effectively incorporate new data to create a model which fits the new points snugly. However, this may cause issues as it may unneccessarily fit to noise within the training data. However if the model is more biased towards wider generalizations, the model will better account for differences among datasets. A too highly biased model however will fail ideal classifications for any data.

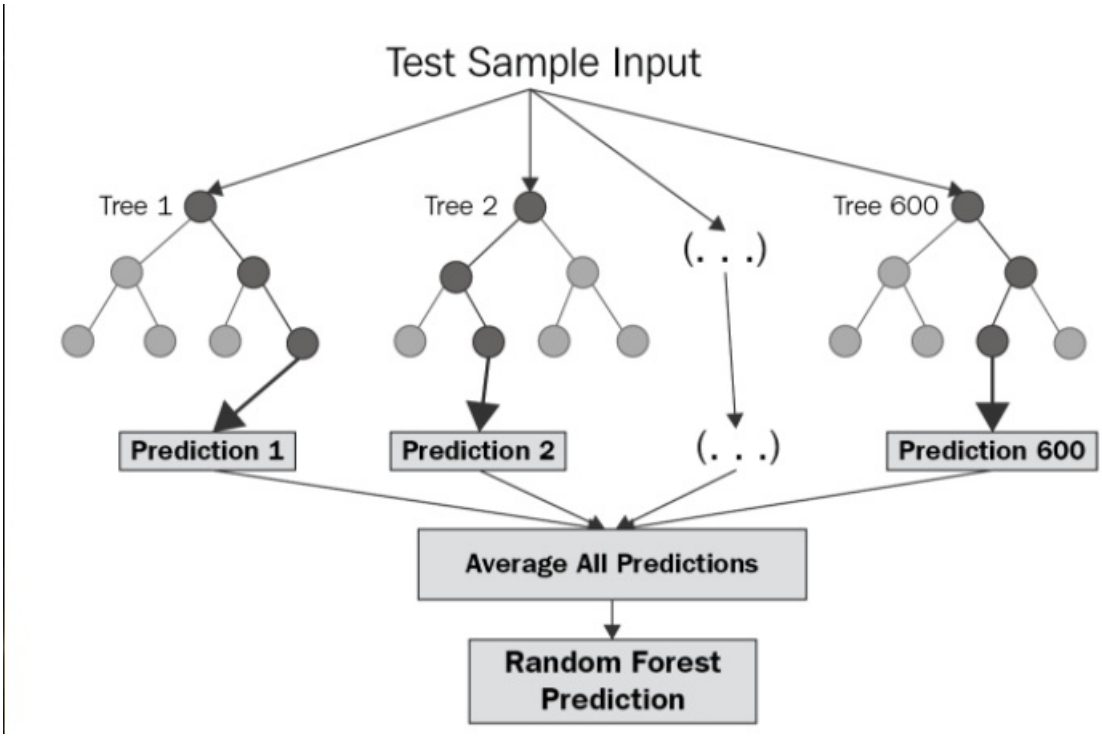
- If Jones develops a decision tree on one car lot, random differences in the distributions of cars in each lot won't allow his decision tree to carry over well when he tries it on a new lot. On the flip side, if he makes a decision tree that is too general, making wide assumptions about the cars betwen lots, his model will fail to effectively classify or regress any cars MPGs.

Decision Trees can easily suffer from having too high of a variance. Although there are methods to

combat this as mentioned above, Random Forests do a spectacular job at reducing this high variance problem.

Random Forest

Contain an *ensemble* of *n* decision trees and *m* boostrap samples, which stop splitting after reaching a specified depth *d*.



We can show the importance of Random Forests mathematically by using the analogy of a set of n uncorrelated random variables Z_i who have the same mean and variance.

$$Z_i ... Z_n \quad E[Z_i] = \mu \quad Var(Z_i) = \sigma$$

The average of all the variables results in the same expectation:

$$E[\frac{1}{n}\sum_{i=1}^{n} Z_i] = \frac{1}{n}\sum_{i=1}^{n} E[Z_i] = \frac{1}{n} * n\mu = \mu$$

However, the group of random variables have less variance compared to the individuals apart.

$$Var(\frac{1}{n}\sum_{i=1}^{n} Z_i) = (\frac{1}{n})^2 \sum_{i=1}^{n} Var(Z_i) = \frac{1}{n^2} * n\sigma^2 = \sigma^2 n$$

The variance is now scaled down by the number of variables in the set.

The random variables Z_i are akin to the prediction by each decision tree made in the random forest. Although the expected prediction of the trees is the same, the variance will be less.

Let's now see how the algorithm functions.

Random Forests Algorithm

1. At the current node, select p features randomly from the available features D.
2. Using the specified splitting metric, compute the ideal split point for tree i of n. Split this node into daughter nodes, which have various subsets of features from their parent node.
3. Repeat the above steps until a the specified tree depth d has been reached.
4. For each tree i of k in the forest, repeat steps 1-3

5. Aggregate or vote on the outputs calculated by each of the trees.

The number of trees in the ensemble k can be very large, in the magnitude of *hundreds to thousands.*

Random Forests Algorithm (In Jones' Case)

1. When time comes to make a split, select p car features randomly from the available features on deck D.
2. Using the specified splitting metric (gini or entropy), compute the ideal decision to split on for henchman *i* of *k*. Split this set of cars into two subsets, which each also have a few features randomly selected from their parent node.
3. Repeat the above steps until a the specified tree depth *d* has been reached.
4. For each henchman *i* of *k* in the bunch, repeat steps 1-3
5. Aggregate or vote on the outputs calculated by each of the henchmen.

The number of henchmen in the group *k* can be very large, in the magnitude of *hundreds to thousands.*

Coding Skills to Help Jones

To create our Decision Trees and Random Forests for Jones, we'll be using the Python library Scikit-Learn. Scikit-Learn contains an adundance of tools to be used for machine learning purposes, and it will be covered in greater depth in later lessons, but for now we'll cover a few features that will help Jones' classify his cars.

Many of the following functions contain more parameters that can be adjusted. For now we'll just cover those that are relevant to our issue. More information can be found in the scikit-learn guide:

https://scikit-learn.org/stable/user_guide.html

Scikit-Learn (Sklearn)

How to create train test splits for datasets:

Train Test Split

```
train_test_split(X, y, test_size, random_state):
Parameters:
# X: data to be split
# y: features to be split
# test_size: proportion of data (rows, not columns) that goes to the test set
# random_state: seed of random process which picks which items go to which split.
```

Decision Tree Classifier

```
DecisionTreeClassifer(criterion, max_depth, max_features, random_state):
# **Parameters**
# criterion: {"gini", "entropy"}, default="gini"
# max_depth: int, default=None
# max_features: int, float or {"auto", "sqrt", "log2"}, default=None
# random_state: int, RandomState instance, default=None

# ** Attributes **
# tree: Tree
```

Random Forests Classifier

```
RandomForestClassifer(criterion, max_depth, max_features, random_state):
# **Parameters**
# criterion: {"gini", "entropy"}, default="gini"
# max_depth: int, default=None
# max_features: int, float or {"auto", "sqrt", "log2"}, default=None
# bootstrap: bool, default=True
# random_state: int, RandomState instance, default=None

# ** Attributes **
# tree: Tree
```

Graphviz and dtreeviz

These are libaries that streamline the process of visualizing decision trees.

Graphviz: https://graphviz.org/

dtreeviz: https://github.com/parrt/dtreeviz

Decision Tree Visualization

```
export_graphviz(classifier, out_file, feature_names, class_names, filled)
# classifier: classifer, classifer to create visualization from
# out_file: str, file_name to export visualization to
# feature_names: iterable, dataset feature names
# class_names: iterable, name of class labels
# filled: bool, to color the nodes or not
```

Sources

Images:

Car Lot
https://www.whichcar.com.au/features/used-cars-1980-ripper-car-movies

Decision Tree
https://scikit-learn.org/stable/modules/tree.html

Random Forest
https://levelup.gitconnected.com/random-forest-regression-209c0f354c84

Gini Impurity and Entropy
https://medium.com/@jason9389/gini-impurity-and-entropy-16116e754b27

Person at Computer with Visualizations
https://flashbak.com/paleotechnology-a-curious-glimpse-into-an-80s-computer-book-2714/

Material:

Implementation of Random Forests
https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

Decision Trees on Scikit-Learn
https://scikit-learn.org/stable/modules/tree.html

Decision Tree Classifier
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decision tree classifier#sklearn.tree.DecisionTreeClassifier

Random Forests Classifier
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Train Test Split
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html?highlight=train test split#sklearn.model_selection.train_test_split

Iris Dataset
https://scikit-learn.org/0.16/modules/generated/sklearn.datasets.load_iris.html

Decision Trees and Random Forests
https://www.eecs189.org/static/notes/n25.pdf