Interpreting random forests

POSTED OCTOBER 19, 2014

Why model interpretation?

Imagine a situation where a credit card company has built a fraud detection model using a random forest. The model can classify every transaction as either valid or fraudulent, based on a large number of features. What if, after a transaction is classified as fraudulent, the analyst would like to know why the model made this decision, i.e. how much each feature contributed to the final outcome?

Or what if a random forest model that worked as expected on an old data set, is producing unexpected results on a new data set. How would one check which features contribute most to the change in the expected behaviour.

Random forest as a black box

Most literature on random forests and interpretable models would lead you to believe this is nigh impossible, since random forests are typically treated as a black box. Indeed, a forest consists of a large number of deep trees, where each tree is trained on bagged data using random selection of features, so gaining a full understanding of the decision process by examining each individual tree is infeasible. Furthermore, even if we are to examine just a single tree, it is only feasible in the case where it has a small depth and low number of features. A tree of depth 10 can already have thousands of nodes, meaning that using it as an explanatory model is almost impossible.

One way of getting an insight into a random forest is to compute feature importances, either by permuting the values of each feature one by one and checking how it changes the model performance or computing the amount of "impurity" (typically variance in case of regression trees and gini coefficient or entropy in case of classification trees) each feature removes when it is used in node. Both approaches are useful, but crude and static in the sense that they give little insight in understanding individual decisions on actual data.

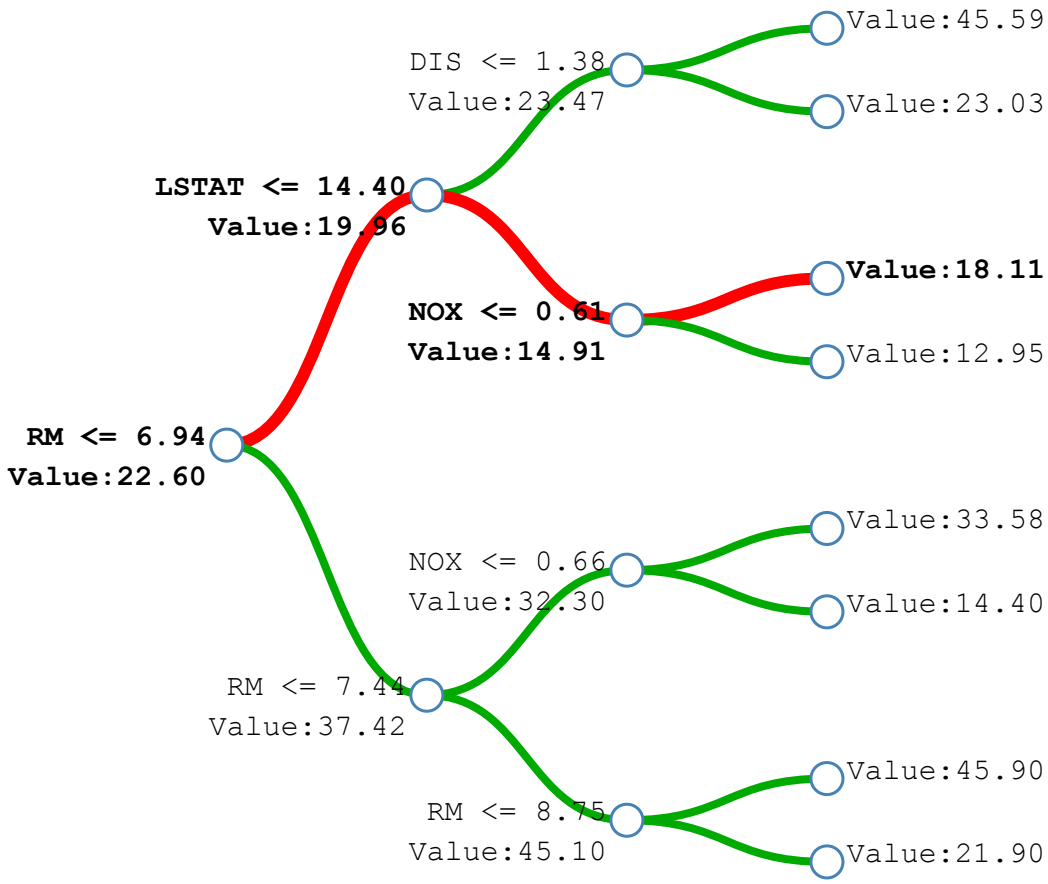
Turning a black box into a white box: decision paths

When considering a decision tree, it is intuitively clear that for each decision that a tree (or a forest) makes there is a path (or paths) from the root of the tree to the leaf, consisting of a series of decisions, guarded by a particular feature, each of which contribute to the final predictions.
A decision tree with M leaves divides the feature space into M regions $R_m, 1 \leq m \leq M$. In the classical definition (see e.g. Elements of Statistical Learning), the prediction function of a tree is then defined as $f(x) = \sum_{m=1}^{M} c_m I(x, R_m)$ where M is the number of leaves in the tree(i.e. regions in the feature space), R_m is a region in the feature space (corresponding to leaf m), c_m is a constants corresponding to region m and finally I is the indicator function (returning 1 if $x \in R_m$, 0 otherwise). The value of c_m is determined in the training phase of the tree, which in case of regression trees corresponds to the mean of the response variables of samples that belong to region

R_m (or ratio(s) in case of a classification tree). The definition is concise and captures the meaning of tree: the decision function returns the value at the correct leaf of the tree. But it ignores the "operational" side of the decision tree, namely the path through the decision nodes and the information that is available there.

Example: Boston housing data

Let's take the Boston housing price data set, which includes housing prices in suburbs of Boston together with a number of key attributes such as air quality (NOX variable below), distance from the city center (DIST) and a number of others – check the page for the full description of the dataset and the features. We'll build a regression decision tree (of depth 3 to keep things readable) to predict housing prices. As usual, the tree has conditions on each internal node and a value associated with each leaf (i.e. the value to be predicted). But additionally we've plotted out the value at each internal node i.e. the mean of the response variables in that region.



Prediction: `18.11` ≈ `22.60` (trainset mean) `- 2.64`(loss from RM) `- 5.04`(loss from LSTAT) `+ 3.20`(gain from NOX)

RM	LSTAT	NOX	DIST	
3.1	4.5	0.54	2.6	Predict
6.5	16.1	0.12	2.2	Predict
7.1	10.5	0.31	1.8	Predict

You can hover on the leaves of the tree or click "predict" in the table (which includes sample values

from the data set) to see the decision paths that lead to each prediction.
What's novel here is that you can see the breakdown of the prediction, written down in terms of value changes along the prediction path, together with feature names that "caused" every value change due to being in the guard (the numbers are approximate due to rounding).

What this example should make apparent is that there is another, a more "operational" way to define the prediction, namely through the sequence of regions that correspond to each node/decision in the tree. Since each decision is guarded by a feature, and the decision either adds or subtracts from the value given in the parent node, the prediction can be defined as the sum of the feature contributions + the "bias" (i.e. the mean given by the topmost region that covers the entire training set).
Without writing out the full derivation, the prediction function can be written down as $f(x) = c_{full} + \sum_{k=1}^{K} contrib(x, k)$ where K is the number of features, c_{full} is the value at the root of the node and $contrib(x, k)$ is the contribution from the k-th feature in the feature vector x. This is superficially similar to linear regression ($f(x) = a + bx$). For linear regression the coefficients b are fixed, with a single constant for every feature that determines the contribution. For the decision tree, the contribution of each feature is not a single predetermined value, but depends on the rest of the feature vector which determines the decision path that traverses the tree and thus the guards/contributions that are passed along the way.

From decision trees to forest

We started the discussion with random forests, so how do we move from a decision tree to a forest? This is straightforward, since the prediction of a forest is the average of the predictions of its trees: $F(x) = \frac{1}{J} \sum_{j=1}^{J} f_j(x)$, where J is the number of trees in the forest. From this, it is easy to see that for a forest, the prediction is simply the average of the bias terms plus the average contribution of each feature: $F(x) = \frac{1}{J} \sum_{j=1}^{J} c_{j\,full} + \sum_{k=1}^{K} (\frac{1}{J} \sum_{j=1}^{J} contrib_j(x, k))$.

Running the interpreter

Update (Aug 12, 2015)
Running the interpretation algorithm with actual random forest model and data is straightforward via using the treeinterpreter (`pip install treeinterpreter`) library that can decompose scikit-learn's decision tree and random forest model predictions. More information and examples available in this blog post.

Summary

There is a very straightforward way to make random forest predictions more interpretable, leading to a similar level of interpretability as linear models — not in the static but dynamic sense. Every prediction can be trivially presented as a sum of feature contributions, showing how the features lead to a particular prediction. This opens up a lot of opportunities in practical machine learning and data science tasks:

- Explain to an analyst why a particular prediction is made
- Debug models when results are unexpected
- Explain the differences of two datasets (for example, behavior before and after treatment), by comparing their average predictions and corresponding average feature contributions.