Random Forest II

Taryn Heilman
Erich Wellinger
DSI Lecture

galvanize

Learning Objectives

- Discuss **Out-of-Bag (OOB) Score** as a method for evaluating model performance

- Discuss **Feature Importance and Partial Dependence** for model interpretability

Review

What is “bagging” short for? How does this algorithm work?

What are the two things that make a random forest “random”?

Why does random forest make better predictions than bagging?

What are the hyper-parameters you can tune in a random forest?

Review: Bagging vs. Random Forest

galvanize

Bagging - The average of many low bias, high variance trees created in parallel; the trees are highly correlated.

Random Forests - The average of many low bias, high variance trees created in parallel; selects a subset of features at each split to create trees with more differences.

Out of Bag Score (OOB Score)

OOB Score is a quick and dirty “replacement” for cross validation

- We already have data that each tree has not seen yet -- each bootstrapped sample only includes about ⅔ of the data.
- We can feed the data that wasn’t used in a tree as a **test set** for that tree.
- We can then aggregate the *accuracy score** for each of our points (each test data point tested on ~⅓ of our trees)

The downside is that oob_score_ in sklearn only computes accuracy or R^2, so if we want precision, recall, or other metrics we will still need to cross validate :(

```
rf = RandomForestClassifier(n_estimators=100, oob_score=True)
rf.fit(X_train, y_train)
print(rf.oob_score_)
```

Interpreting your ensemble model

Interpreting Trees, Bagged Trees and Forests

galvanize

Recall, one of the main strengths of Decision Trees is interpretability.

However, when we aggregate our trees with simple Bagging or Random Forests, it's not so easy…

- We can no longer simply rank our features in the order in which they were split on

But… We can look at ***Feature Importances*** *(Note: nowhere near as reliable as coefficients for a linear regression)*

Measuring Feature Importance

galvanize

There are a several ways we can go about identifying important features:

1. Measure the total amount the information gain increases due to splits over a given feature

2. Record what portion of points pass through a single split -- the higher in a tree a feature is split on, the more important

3. **Combine 1 & 2 with rf.feature_importances_ (where rf is your fit RandomForestClassifier / RandomForestRegressor)**

More Ways to Measure Feature Importance

galvanize

Two more methods...

1. When tree B_i is grown score it with OOB, then remove that feature and score it again to measure the change in your validation metric(s) **--** This is called **Leave One Out Feature Importances**

2. **Iterate through features dropping m_i out and plotting feature importances -- help with "multicollinearity"**

Measuring Feature Importances

galvanize

Let's get some intuition for how we calculate feature importances…

1. For each feature m_j, we calculate the decrease in our impurity criterion (MSE, Gini, etc.) for the node(s) that split on m_j
2. We then weight it by how many points passed through the nodes that split on m_j
3. And finally, we average the calculations for steps 1 and 2 across our entire forest

Feature Importance Continued...

galvanize

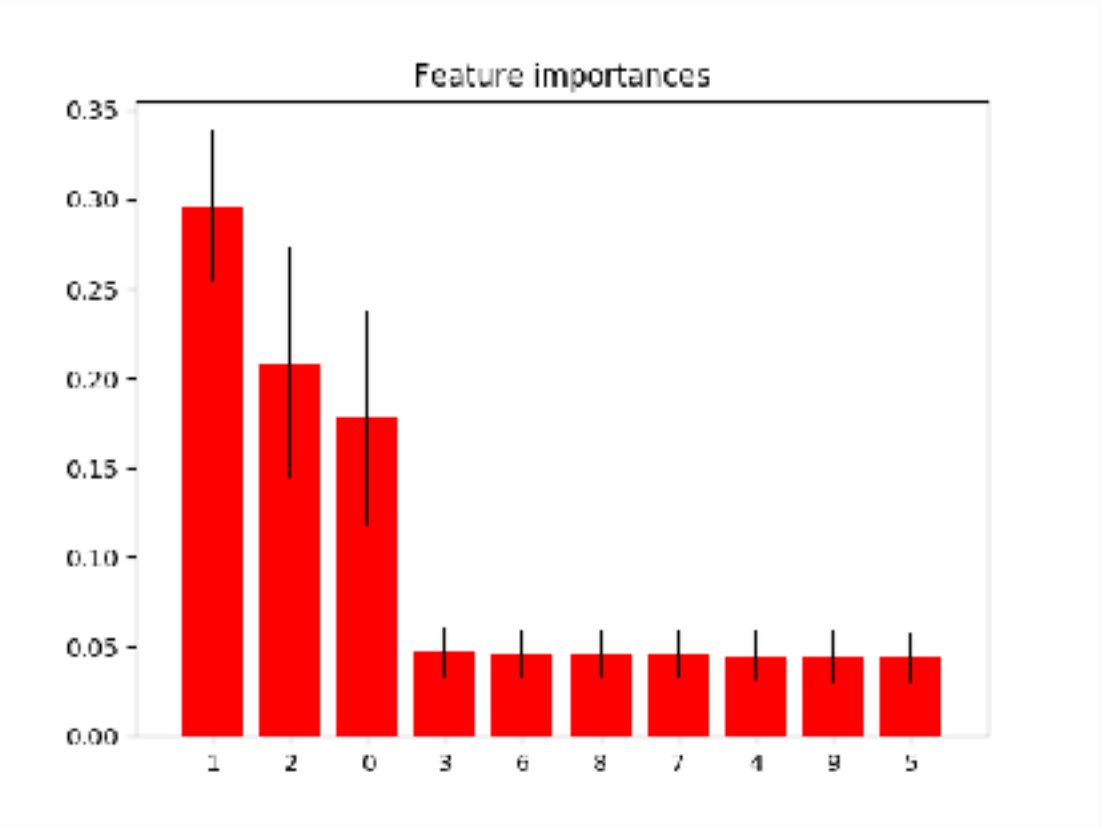
What does feature importance not tell us?

It does NOT tell you effect size and direction of a feature… But this is the *price we pay* for a model that can handle *nonlinear relationships*.

For most real world problems, *features don't have a constant effect size* across all X-values, and sometimes the *effect direction can even reverse* at different levels of X.

What it does tell us: what proportion of our splits (weighted by how much data passed through the split) used this feature - LOOSELY translates to how much variance in y is explained by this feature.

Feature Importances



http://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

Partial Dependence Plots

galvanize

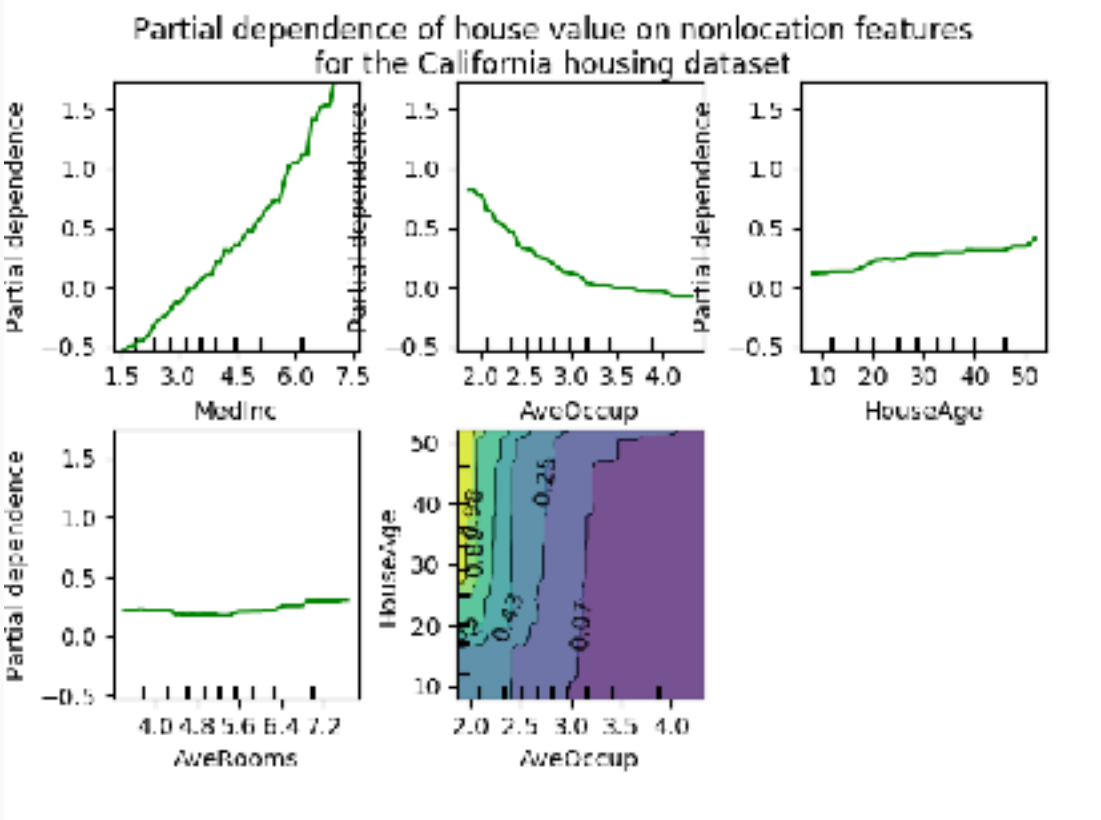
Permute values for each feature column and compare predictive “success” of the feature at each value.

Compare one (or two, if you use a 3D plot) features and how predictive they are at different values.

How to interpret y - axis: It’s all relative. For a classification example, a negative value means for that particular value of predictor variable it is less likely to predict the positive class on that observation and having a positive value means it more likely to predict the positive class. Same applies to two variable plots, color represents the intensity of effect on model.

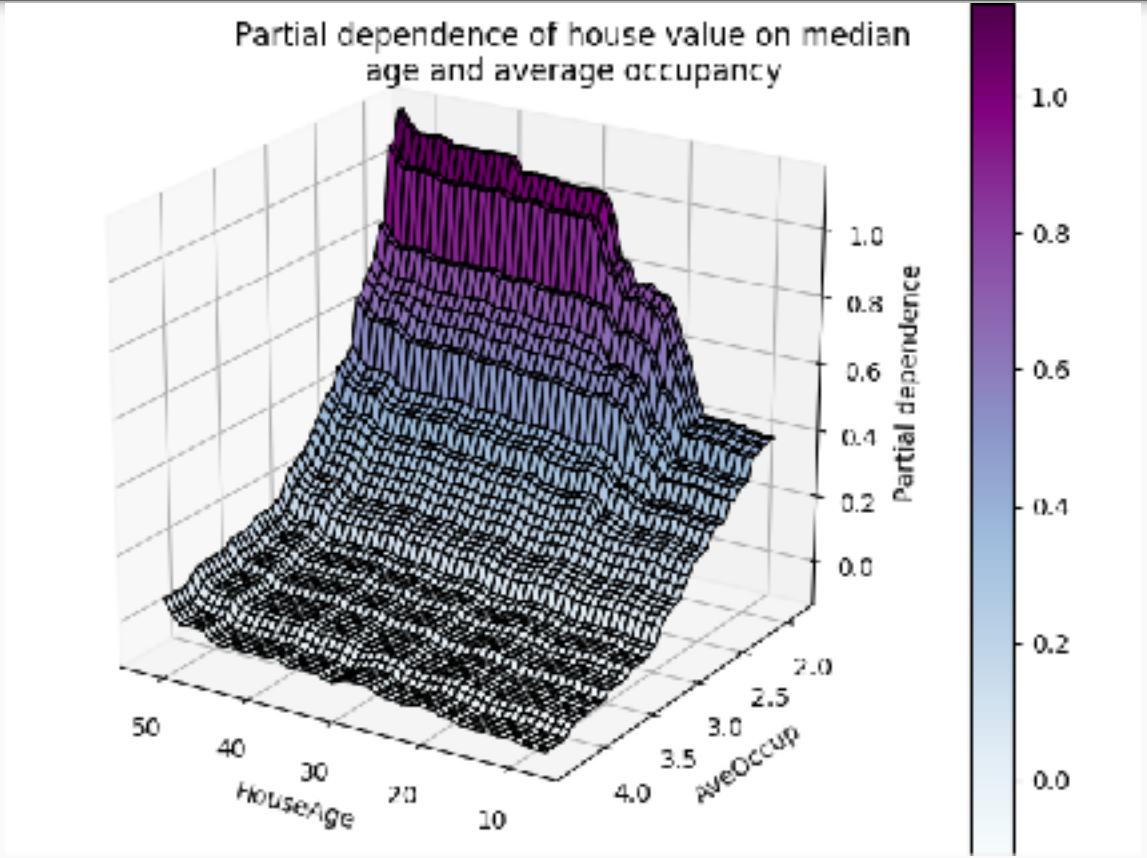
https://www.kaggle.com/dansbecker/partial-dependence-plots

Partial Dependence Plots



http://scikit-learn.org/stable/auto_examples/ensemble/plot_partial_dependence.html

Partial Dependence Plots



Recap: Learning Objectives

galvanize

- ✓ Discuss **Out-of-Bag (OOB) Score** as a method for evaluating model performance

- ✓ Discuss **Feature Importance and Partial Dependence** for model interpretability