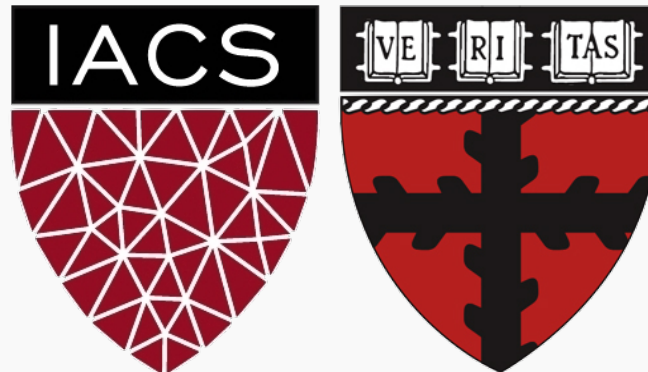


Random Forest

CS109A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai



**I Entered
A Random Forest**



**Now I see
The Future**

Random Forests

Random Forest is a modified form of bagging that creates ensembles of independent decision trees.

To de-correlate the trees, we:

1. train each tree on a separate bootstrap sample of the full training set (same as in bagging)
2. for each tree, at each split, we **randomly** select a set of J' predictors from the full set of predictors.

From amongst the J' **predictors**, we select the optimal predictor and the optimal corresponding threshold for the split.

Tuning Random Forests

Random forest models have multiple **hyper-parameters** to tune:

1. the number of predictors to randomly select at each split
2. the total number of trees in the ensemble
3. the maximum depth or minimum leaf node size

In theory, each tree in the random forest is **full**, but in practice this can be computationally expensive (and added redundancies in the model), thus, imposing a minimum node size or `max_depth` is not unusual.

Tuning Random Forests

There are standard (default) values for each of random forest hyper-parameters recommended by long time practitioners, but generally these parameters should be tuned through **OOB** (making them data and problem dependent).

e.g. number of predictors to randomly select at each split:

- $\sqrt{N_j}$ for classification
- $\frac{N}{3}$ for regression

Using out-of-bag errors, training and cross validation can be done in a single sequence - we cease training once the out-of-bag error stabilizes

Variable Importance for RF

1. Mean Decrease in Impurity (MDI)

- Same as Bagging.
- Record the prediction accuracy on the *oob* samples for each tree.
- Calculate the total amount that the RSS (for regression) or Gini index (for classification) is decreased due to splits over a given predictor, averaged over all trees.
- The decrease in accuracy as a result of this permuting is averaged over all trees, and is used as a measure of the importance of variable j in the random forest.
- The default in Scikit-learn `feature_importances_`

Variable Importance for RF

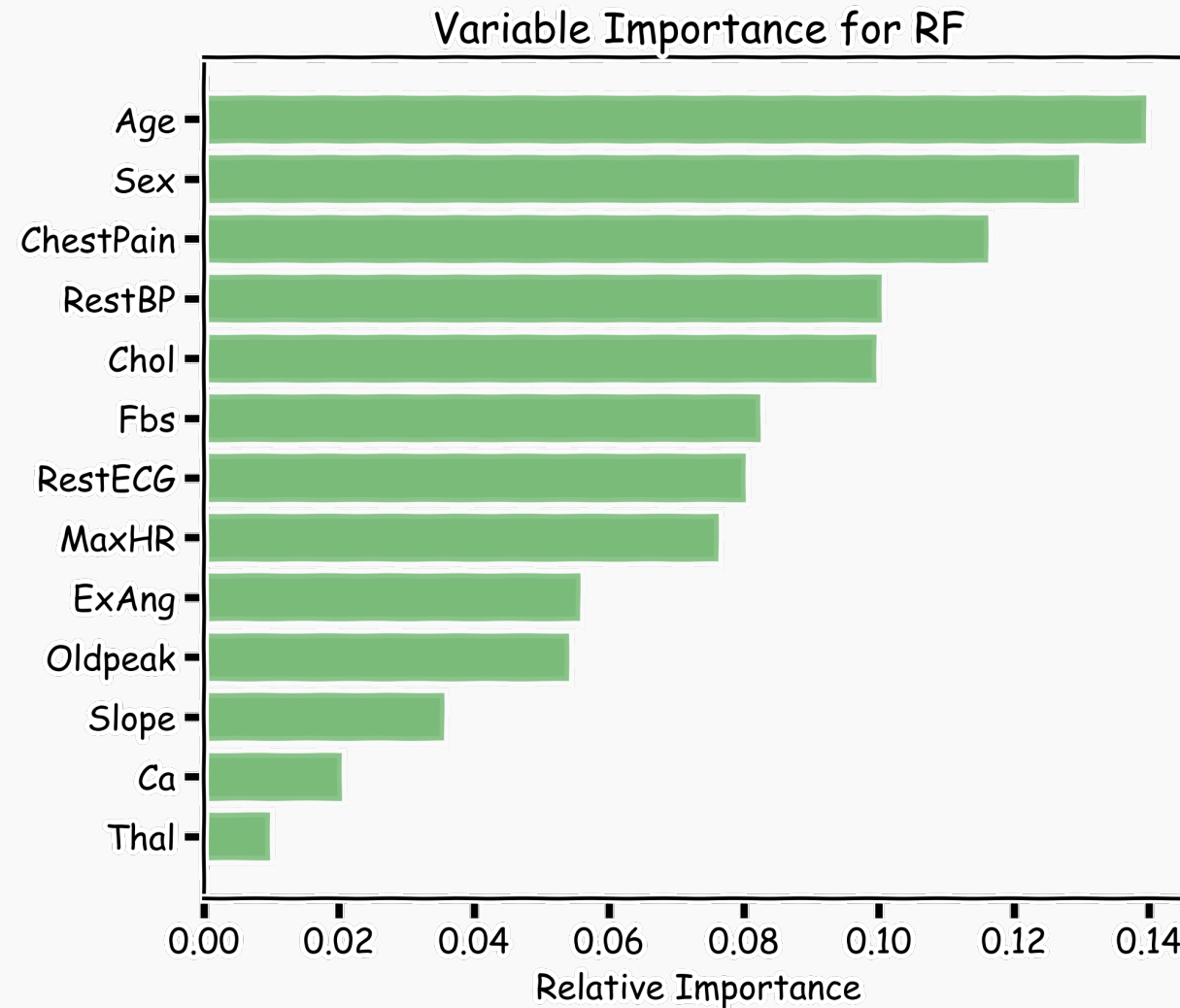
2. Permutation Importance

- Record the prediction accuracy on the *oob* samples for each tree.
- Randomly permute the data for column j in the *oob* samples the record the accuracy again.
- The decrease in accuracy as a result of this permuting is averaged over all trees, and is used as a measure of the importance of variable j in the random forest.

3. One step further (SHAP values, LIME)

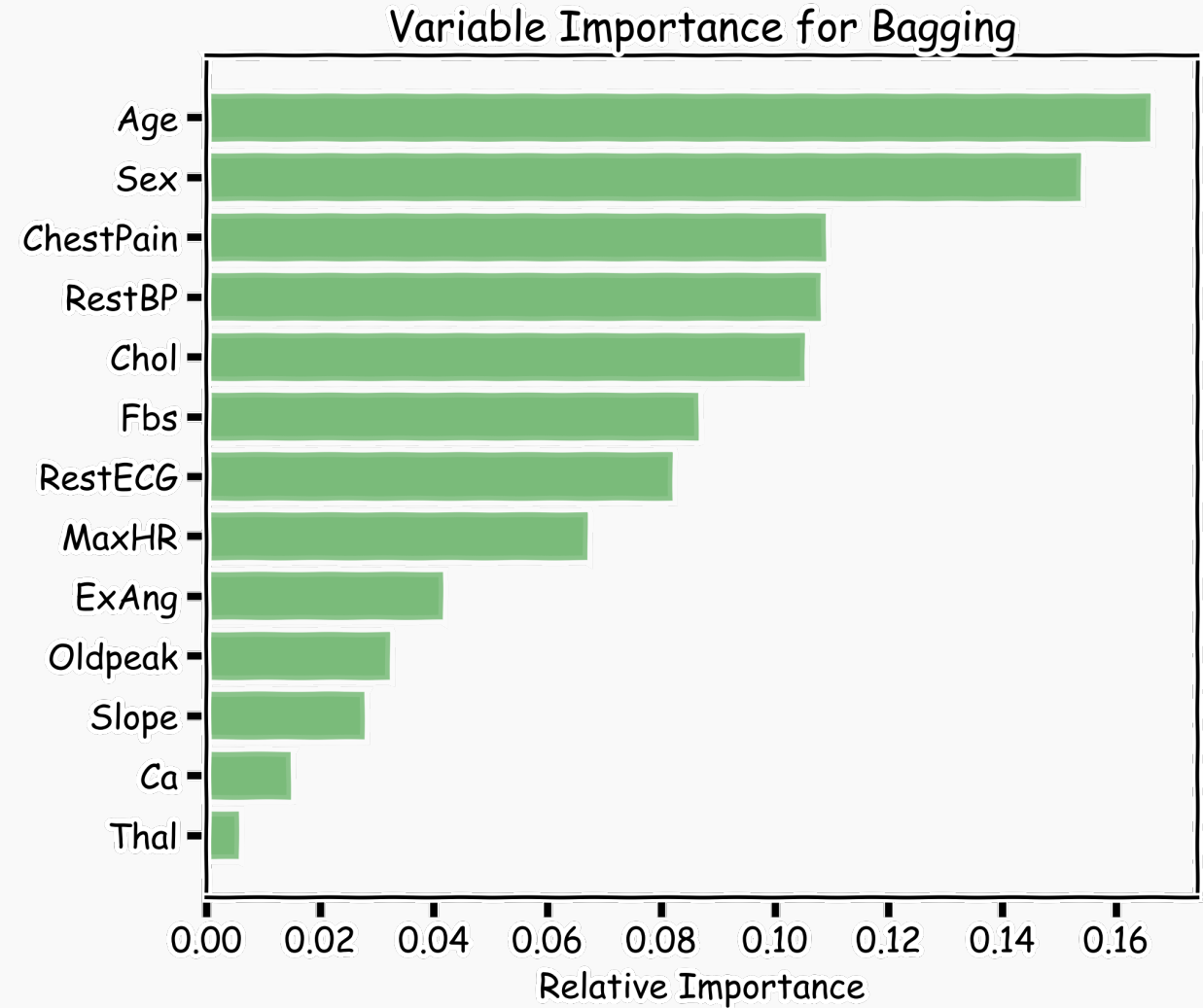
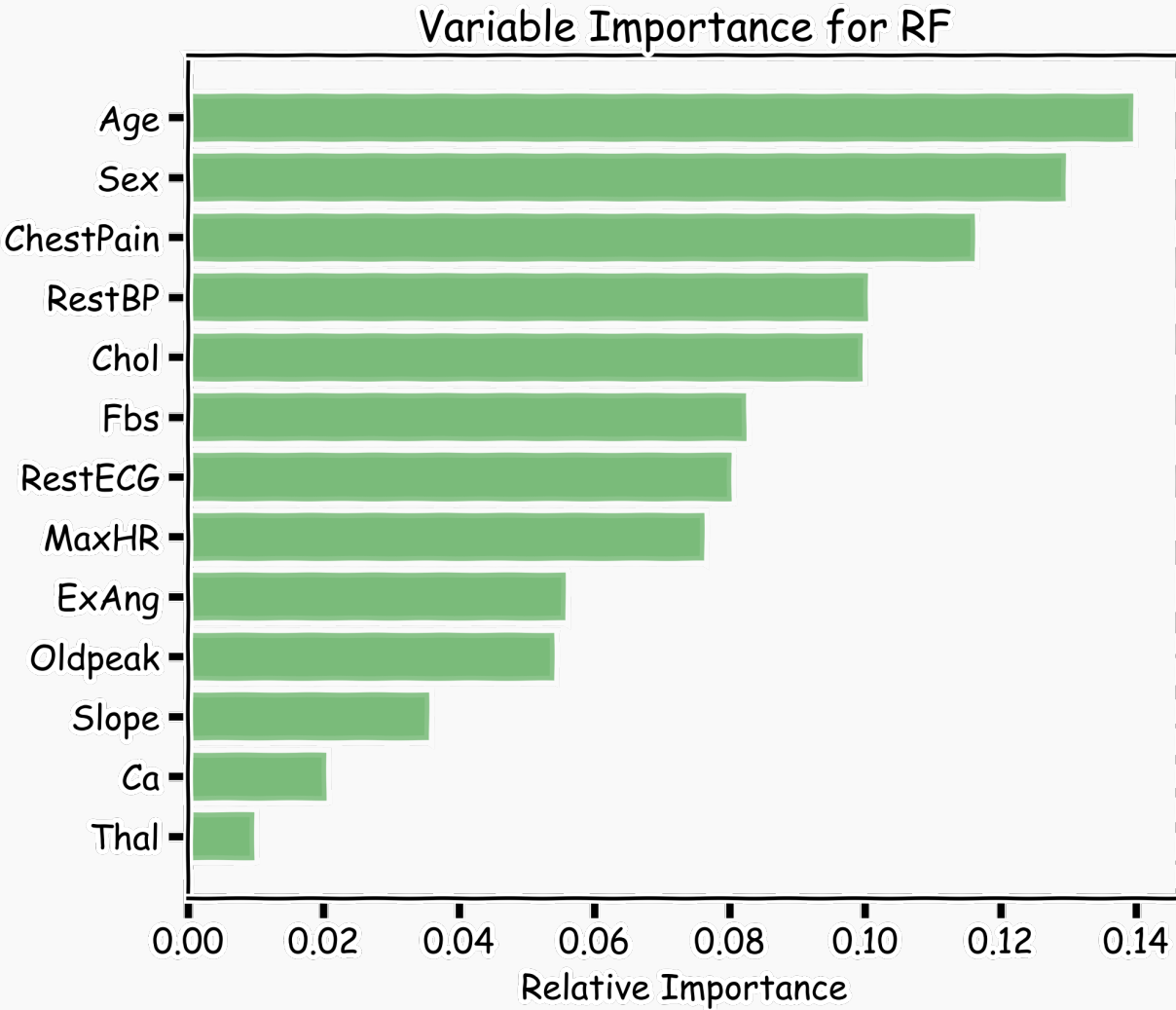
- We will see these methods in later lectures.

Variable Importance for RF



100 trees, max_depth=10

Variable Importance for RF



100 trees, max_depth=10



When the number of predictors is large, but the number of relevant predictors is small, random forests can perform poorly.

Question: Why?

In each split, the chances of selected a relevant predictor will be low and hence most trees in the ensemble will be weak models.

Final Thoughts on Random Forests

Increasing the number of trees in the ensemble generally does **not increase the risk of overfitting**.

Again, by decomposing the generalization error in terms of bias and variance, we see that increasing the number of trees produces a model that is at least as robust as a single tree.

However, if the number of trees is too large, then the trees in the ensemble may become more correlated, increase the variance.



Probabilities:

- Random Forrest Classifier (and bagging) can return probabilities.
- **Question:** How?



Jordan Goldmeier
@Option_Explicit

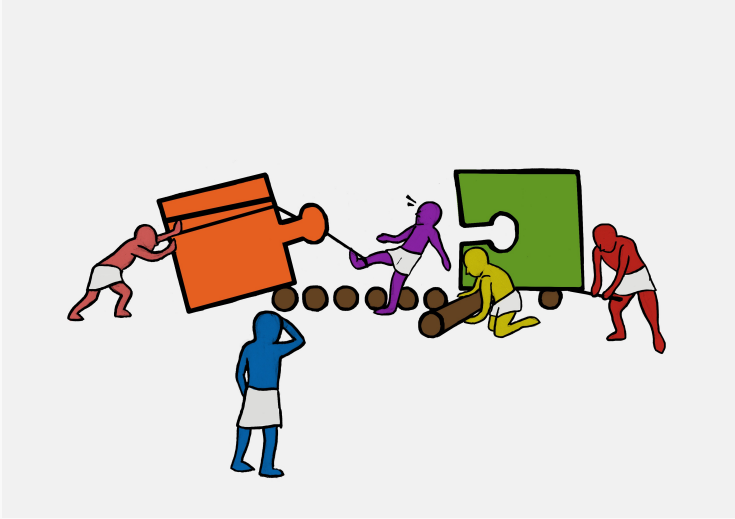


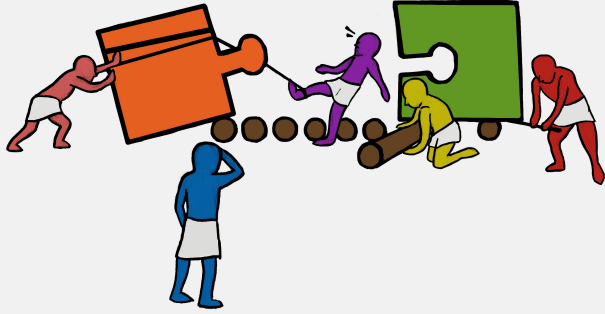
Where do data scientists go
camping?

In random forests

Next Lecture

- Unbalance dataset
 - Weighted samples
- Categorical data
- Missing data
- Different implementations





🏋️ Exercise: Bagging vs Random Forest (Tree correlation)

How does Random Forest improve on Bagging?

The goal of this exercise is to investigate the correlation between randomly selected trees from *Bagging* and *Random Forest*.

Instructions:

- Read the dataset `diabetes.csv` as a pandas dataframe, and take a quick look at the data.
- Split the data into *train* and *validation* sets.
- Define a `BaggingClassifier` model that uses `DecisionTreeClassifier` as its base estimator.
- Specify the number of bootstraps as 1000 and a maximum depth of 3.
- Fit the `BaggingClassifier` model on the *train* data.
- Use the helper code to predict using the mean model and individual estimators. The plot will look similar to the one given below.
- Predict on the test data using the first estimator and the mean model.
- Compute and display the *validation* accuracy
- Repeat the modeling and classification process above, this time using a `RandomForestClassifier`.





Exercise: Hyperparameter tuning

Tuning the hyperparameters

Random Forests perform very well out-of-the-box, with the pre-set hyperparameters in `sklearn`. Some of the tunable parameters are:

- The number of trees in the forest: `n_estimators`, int, default=100
- The complexity of each tree: stop when a leaf has \leq `min_samples_leaf` samples
- The sampling scheme: number of features to consider at any given split: `max_features` {"auto", "sqrt", "log2"}, int or float, default="auto".

