Random Forest

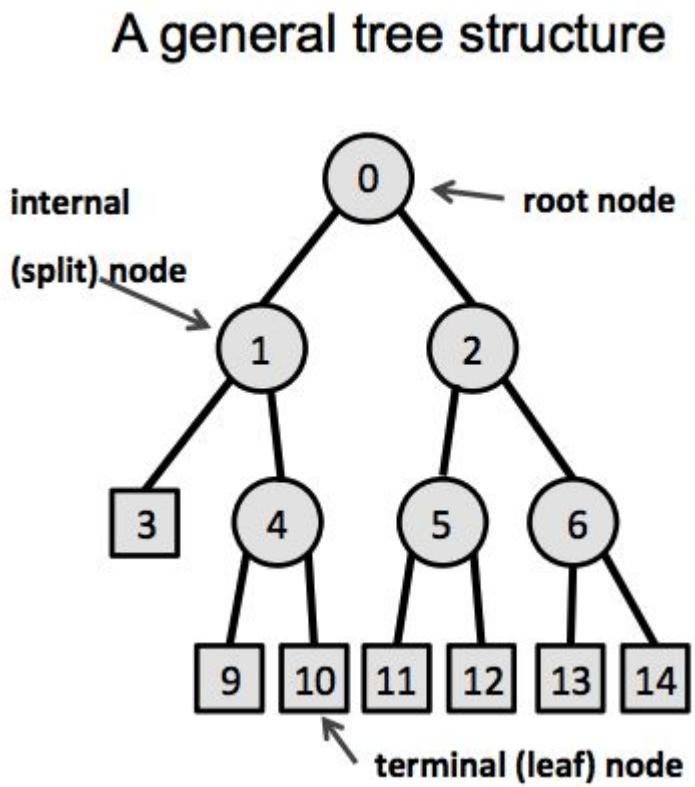
Dan Rupp
2017

Objectives

- Review Decision Trees
- What is an Ensemble?
  - Why are they useful
- What is Bagging?
  - How is it done?
  - Why is it good?
- What is Random Forest?
  - How is it different than Bagging?
  - Can we interpret variables?

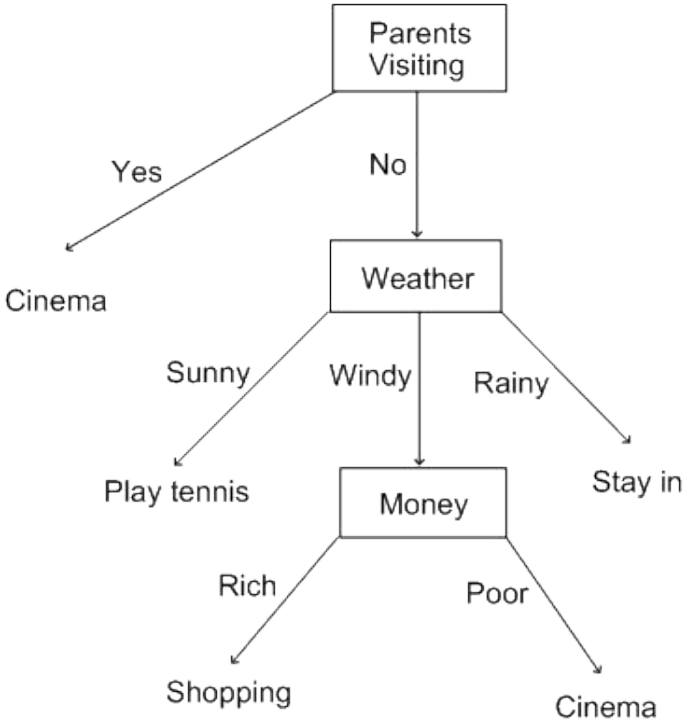
Decision Trees Review

- From a root node data is split on feature to maximize purity
- Data is recursively split at each node
- When required depth / purity / ratio is met a leaf / terminal node is reached



Decision Trees

Benefits:

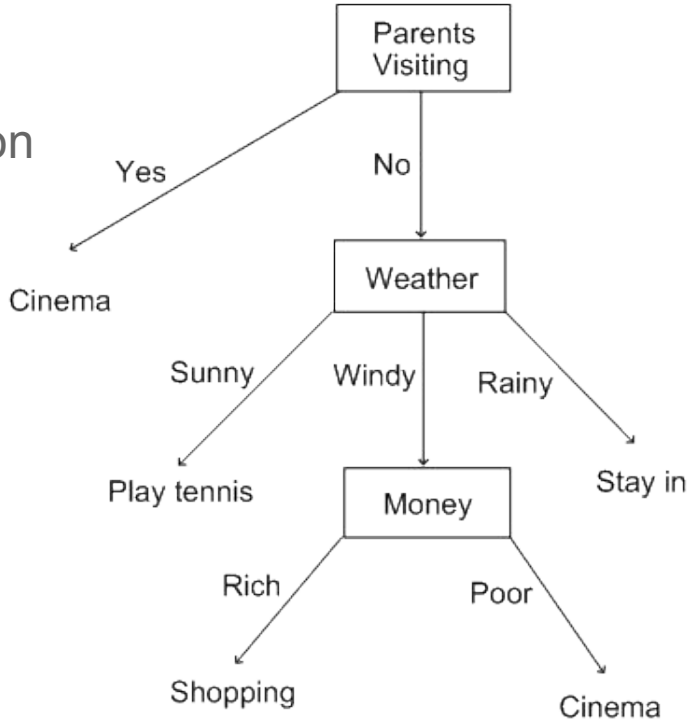


Decision Trees

Benefits:

- non-parametric, non-linear
- can be used for classification and for regression
- real and/or categorical features
- easy to interpret
- computationally cheap prediction
- handles missing values and outliers
- can handle irrelevant features

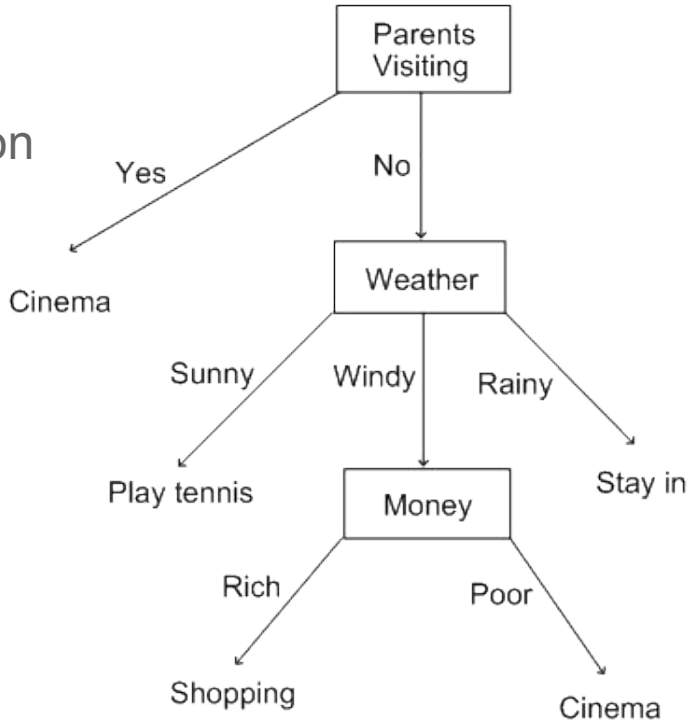
Drawbacks:



Decision Trees

Benefits:

- non-parametric, non-linear
- can be used for classification and for regression
- real and/or categorical features
- easy to interpret
- computationally cheap prediction
- handles missing values and outliers
- can handle irrelevant features



Drawbacks:

- expensive to train
- greedy algorithm (local maxima)
- easily overfits
- Sensitive to the data they are trained on **(high variance)**

Ensembles: Intuition

Suppose you are trying to predict the election...
You have 5 expert opinions, each with a 70% chance of being right,
and each expert pick is independent of the other expert picks.

How could you leverage expert picks to improve accuracy and how often would you be right?

See Jupyter Notebook

Ensembles: Intuition

Combining the models can reduce the variance

Unfortunately, if all the learners are the same, creating an ensemble model won’t help.

Ensembles: Intuition

Combining the models can reduce the variance

Unfortunately, if all the learners are the same, creating an ensemble model won't help.

A solution to this is to train each learner on a different subset of the data. But how do we do this when we only have one set of data to work with?

Bootstrapping to the rescue!

Bootstrap Aggregation (Bagging)

- By training M different trees on subsets of the data (chosen randomly with replacement) we can compute the ensemble
- Bagging uses majority vote for classification and averaging for regression
- The result of the ‘Ensemble’ of multiple trees decreases the variance.

$$f(x) = {}^{1}\!/_{M} \sum_{m=1}^{M} f_m(x)$$

Bagging

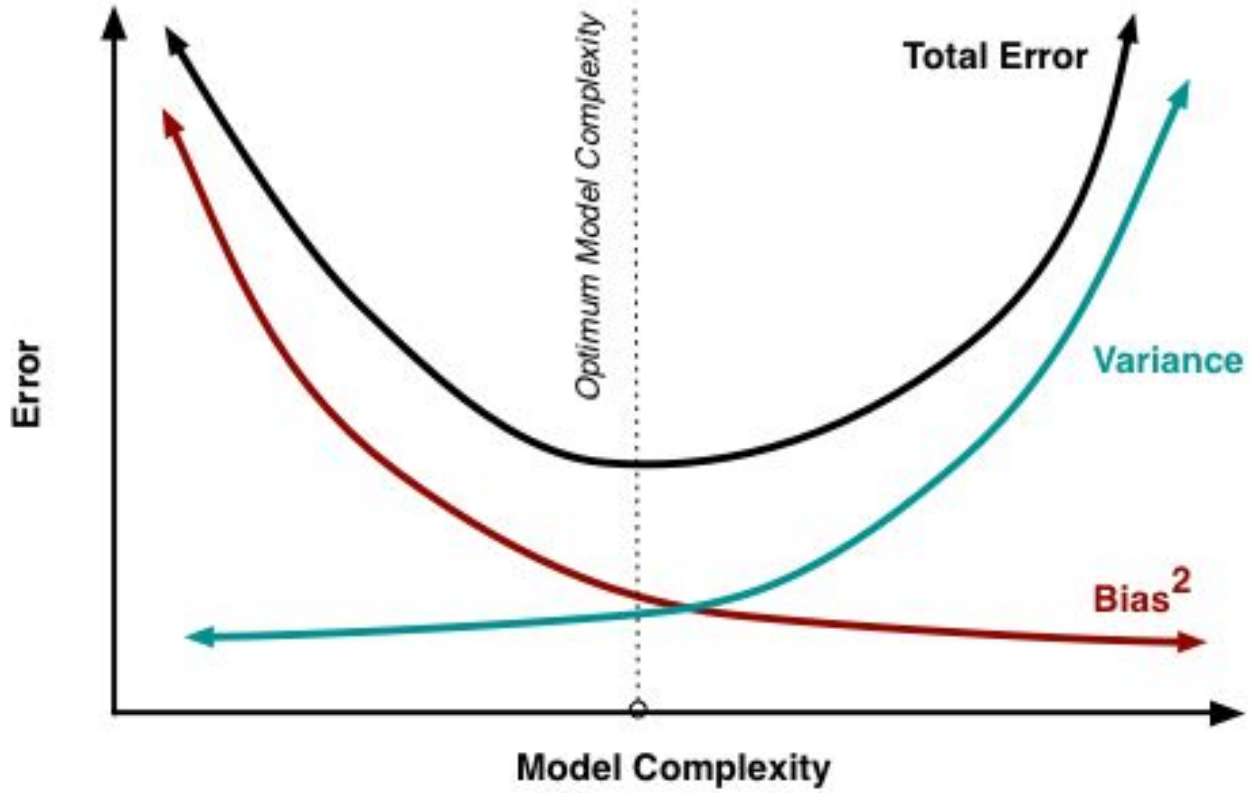
- Bootstrapped trees provide unbiased, high variance predictors
- Averaged estimators lower variance
- Averaged predictors are still unbiased

If each $\hat{f}^{(j)}(\boldsymbol{x}_0)$ is an unbiased predictor of $f(\boldsymbol{x}_0)$ then

$$\mathrm{E}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = f(\boldsymbol{x}_0)$$

$$\mathrm{Var}\left[\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \sigma^2, \text{ then } \mathrm{Var}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \frac{\sigma^2}{m}$$

Bias vs Variance



Issues of Correlation?

If $\text{Cor}\left[\hat{f}^{(j)}, \hat{f}^{(k)}\right] = \rho$ for all j and k

then

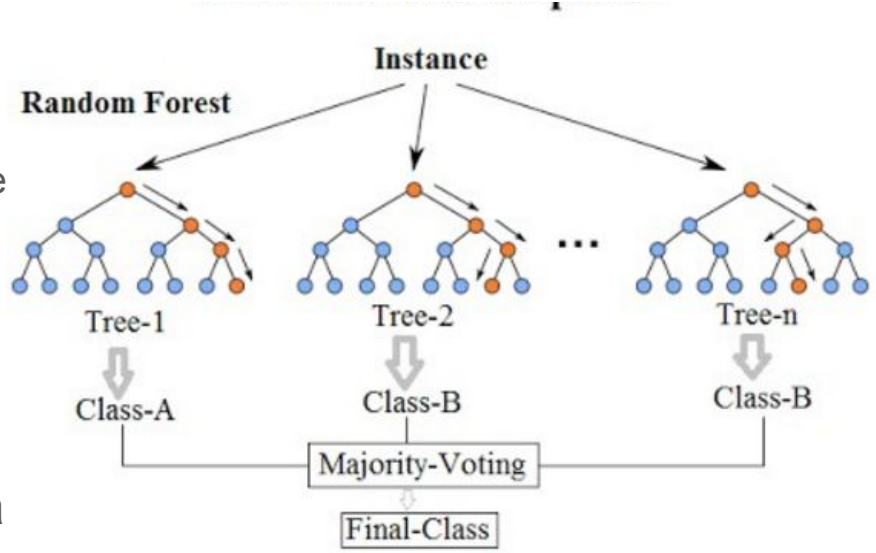
$$\text{Var}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \rho\sigma^2 + (1-\rho)\frac{\sigma^2}{m}$$

Only "uncorrelated parts" of $\hat{f}^{(j)}$ and $\hat{f}^{(k)}$ get "CLT effect"

So is there any way to get ρ close to 0?

Decorrelating Trees

- Trees are still ‘correlated’ because
  - Bootstrap samples are usually about the same
  - Important features tend to stay the same

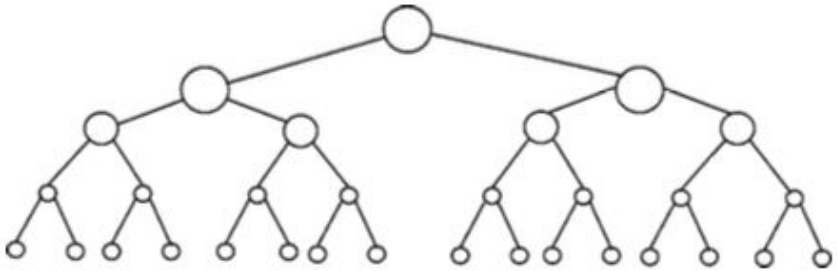


If there is a high feature importance that feature will likely end up towards that top of a tree. Making them more correlated.

Random Forest

How can we further decorrelate trees?

Trees are constructed by recursive best split on features



Random Forest

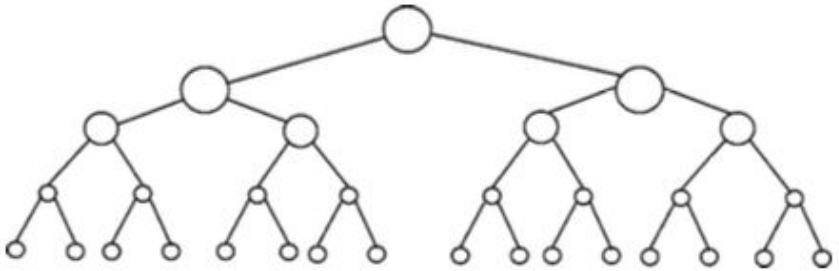
How can we further decorrelate trees?

Trees are constructed by recursive best split on features

We use a random subset of features (p) to consider at each split

Typically $\sqrt{p}$ features used for classification

p/3 features for regression



Random Forest Tuning

- m - Number of trees in forest
- g(p) - Number of features to consider at each split
  - Sqrt(p) for classification
  - p/3 for regression
- Nb - Sample size of bootstrap sample
- X - Tree Characteristics (same as Decision Trees)

Large numbers of ‘bushy’ trees are usually used in Random Forest

End of morning lecture

Objectives

- Review Random Forest
- Go into tuning parameters of Random Forest in detail
- What is OOB (Out of Box Error)?
-

Random Forest

Pros:

Cons:

Random Forest

Pros:

- Often give near state-of-the-art performance Good out-of-the-box performance
- No feature scaling needed
- Model nonlinear relationships

Cons:

- Can be expensive to train (though can be done in parallel)
- Models can be quite large (the pickled version of a several hundred tree model can easily be several GBs)
- Not interpretable (although techniques such as predicted value plots can help)

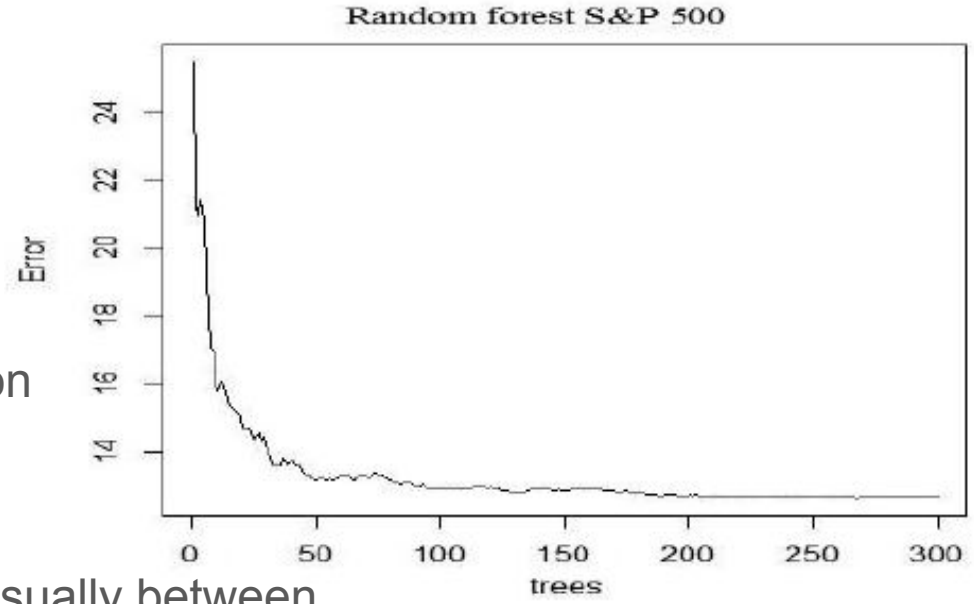
Random Forest Tuning

- m - Number of trees in forest
- g(p) - Number of features to consider at each split
  - Sqrt(p) for classification
  - p/3 for regression
- Nb - Sample size of bootstrap sample
- X - Tree Characteristics (same as Decision Trees)

Large numbers of ‘bushy’ trees are usually used in Random Forest

Number of Trees to Use (m)

Increasing trees decreases variance and increases accuracy of predictions but how many trees should we model?



More trees increase computation

The recommended number is usually between

Out of Box Error (OOB)

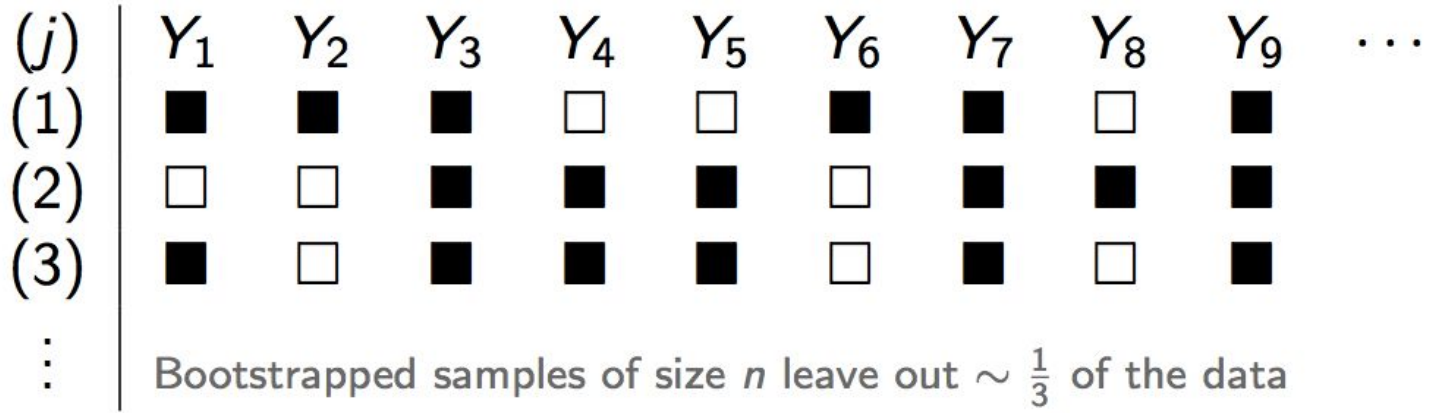
Out of Box Error (OOB)

- Out Of Bag error is a method of estimating the error of ensemble methods that use Bagging.

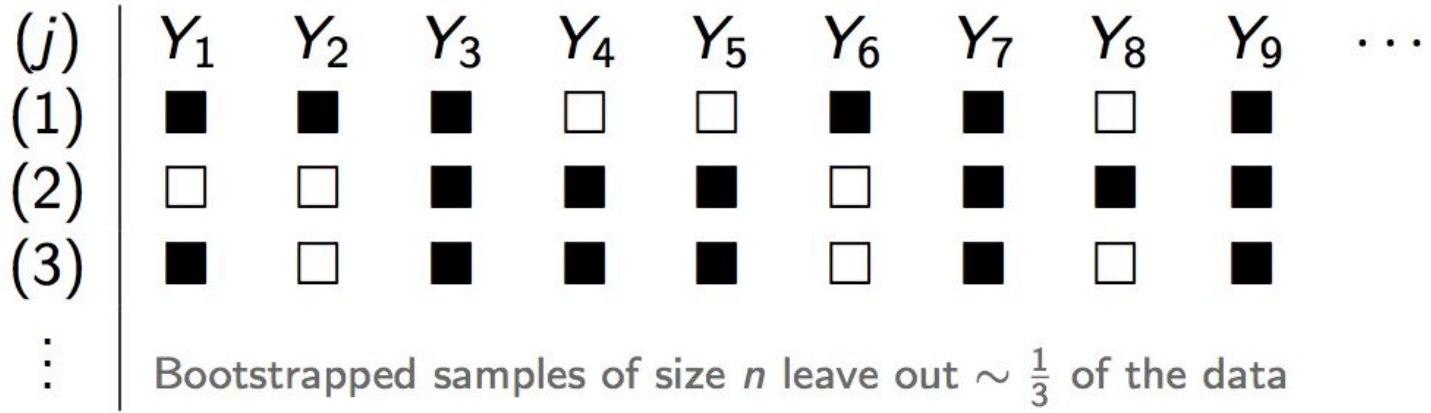
- About 1/3 of the estimators will not have been trained on each data point. (Why?)

- Test each data point only on the estimators that didn’t see that data point during training.

Out of Box Error (OOB)



Out of Box Error (OOB)



Often we'll use cross validation anyway because we're comparing random forest to other models and we want to measure the accuracy the same way.

Feature Importance

- Determining what features impact predictions is critical in many questions for business.
- Because Decision Trees and thus Random Forest can effectively model with features of low importance we must be able to see what features are important
- Example: Churn analysis - Businesses usually want to understand why customers are churning rather than just predict which ones will churn

Feature Importance: Mean Decrease Impurity

How much does each feature decrease the impurity?

Compute importance of the feature j:

- For each tree, each split is made to reduce the impurity (Gini/entropy/MSE), we can record the magnitude of the reduction
- The reduction can then be averaged across all trees in the forest
- This is implemented in sklearn

Keep in mind:

- It is biased towards variables with more categories
- Correlated features can be interchangeably used, if one is used the importance scoring of the others is greatly decreased

Feature Importance: Mean Decrease Accuracy

How does mixing values of features affect accuracy?

Compute importance of the feature j:

- When the n^(i) tree is grown, use it to predict the OOB samples and record accuracy
- Shuffle the values of the j feature in the OOB samples and run the prediction again
- Average the decrease in accuracy across all trees

Random Forest - Quick Review

- Advantages
  - Model nonlinear relationships
  - Can easily deal with continuous or categorical data*
  - No feature scaling necessary
  - Can easily handle missing values*
  - Somewhat interpretable
  - Good prediction accuracy (even ‘Out of the Box’)
- Disvantages
  - Expensive to train
  - Large models

*Not in Python