CS M148 –

Data Science Fundamentals

Lecture #12: Classification Trees

Baharan Mirzasoleiman

UCLA Computer Science

Announcements

HW 1

- Grades are posted

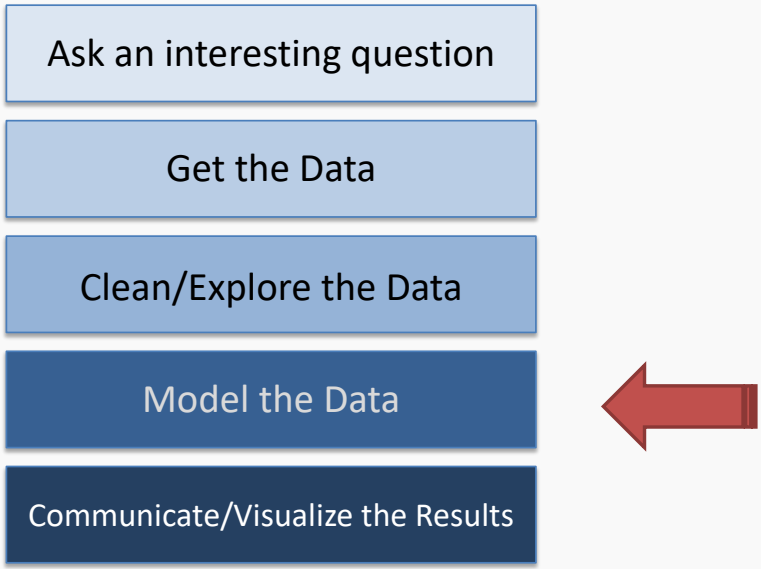
HW 2

- Is posted, due Wed Feb 16 at 2pm

Let’s quickly review what we saw last time

Still here!

The Data Science Process



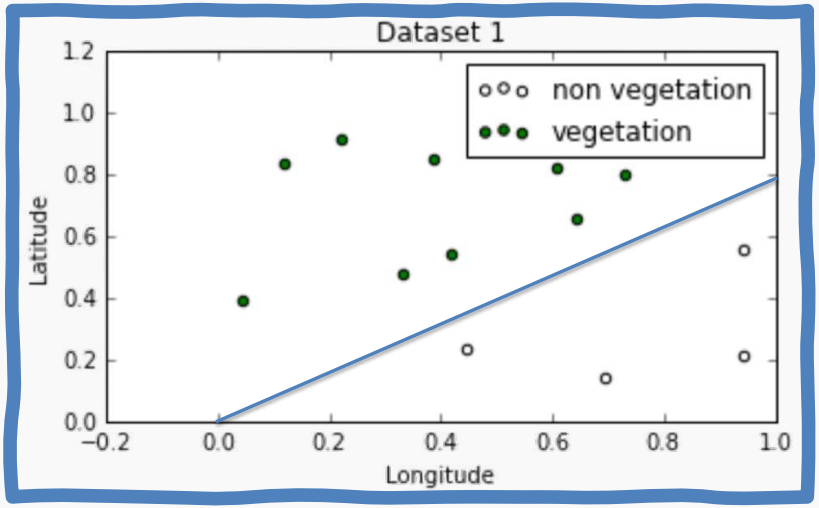
Classification

Outline

- **Review of Decision Trees**
- Splitting Criteria
- Stopping Conditions & Pruning
- Regression Trees
- Bagging
- Out of Bag Error (OOB)
- Variable Importance

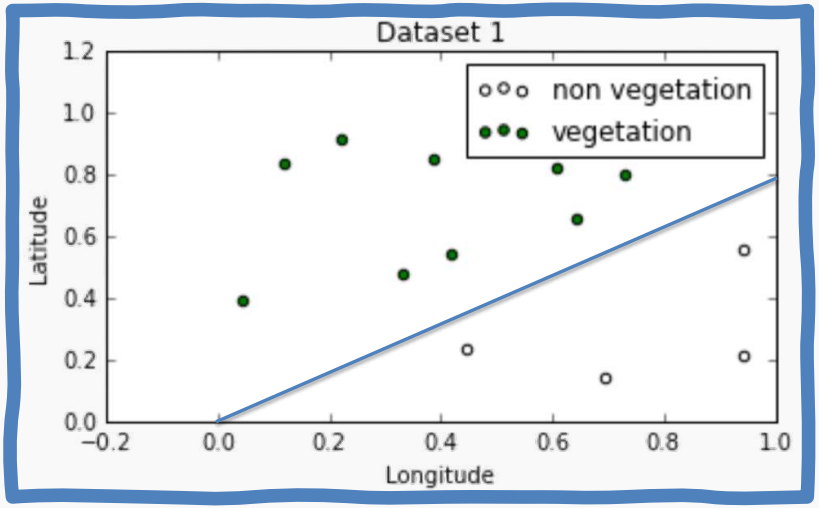
Geometry of Data

Question: Can you guess the equation that defines the decision boundary below?



Geometry of Data

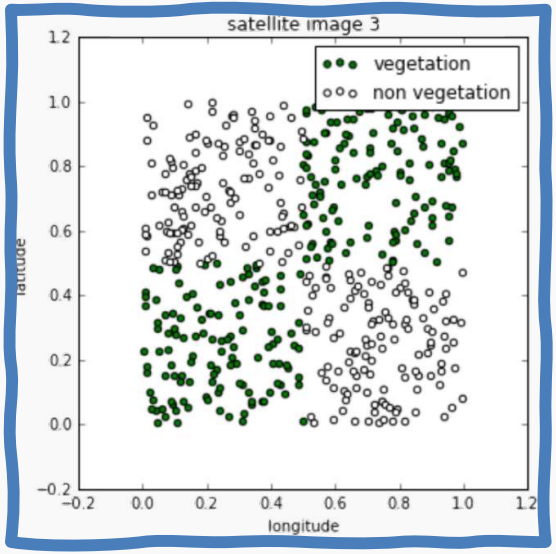
Question: Can you guess the equation that defines the decision boundary below?



$-0.8x_1 + x_2 = 0 \implies x_2 = 0.8x_1 \Rightarrow \ Latitude = 0.8\ Lon$

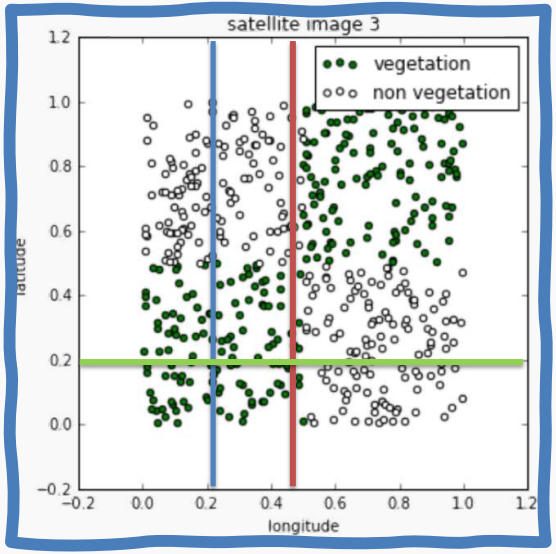
Geometry of Data

Complicate decision boundaries can not be explained with LogRegression.



Geometry of Data

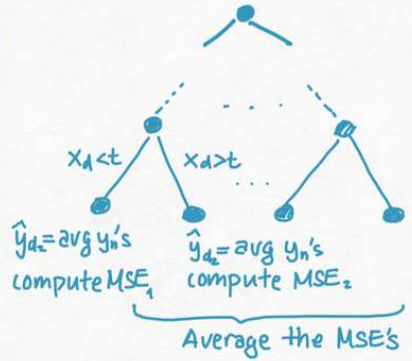
But can be using Trees. Which one represents the first split of a decision tree?



Decision Trees

To learn a decision tree model, we take a greedy approach:

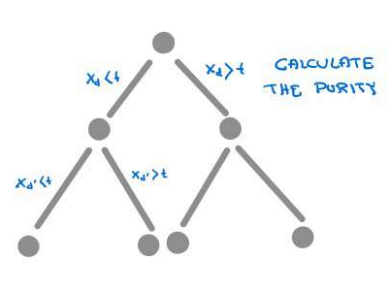
1. Start with a node containing all the data.
2. If stopping condition is not met:
   A. Choose the 'optimal' predictor and threshold and divide the data in the node into two sets.
3. For each new node, repeat step 2.



Decision Trees: Splitting Criteria

Splitting Criteria:

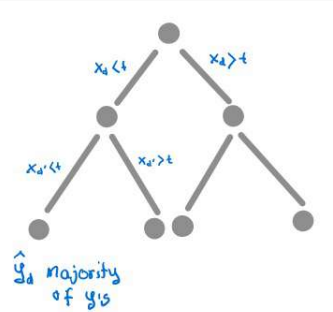
For classification, purity of the regions is a good indicator the performance of the model. Entropy as a splitting criterial minimizes the cross-entropy (greedy). Gini is also a splitting criteria.



Decision Trees: Prediction

Prediction:

For classification, we label each region in the model with the label of the class to which the **plurality** of the points within the region belong.



Stopping Conditions & Pruning

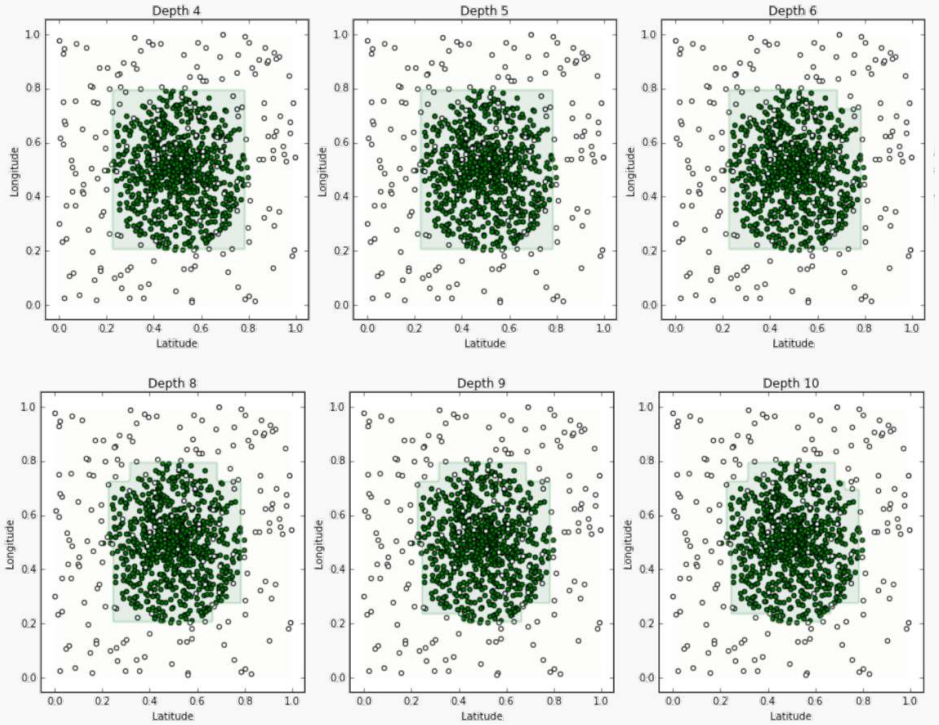
Variance vs Bias

If we don't terminate the decision tree learning algorithm manually, the tree will continue to grow until each region defined by the model possibly contains exactly one training point (and the model attains 100% training accuracy).

To prevent this from happening, we can simply stop the algorithm at a particular depth.

But how do we determine the appropriate depth?

Variance vs Bias



Variance vs Bias

We make some observations about our models:

- **(High Bias)** A tree of depth 4 is not a good fit for the training data - it's unable to capture the nonlinear boundary separating the two classes.
- **(Low Bias)** With an extremely high depth, we can obtain a model that correctly classifies all points on the boundary (by zig-zagging around each point).
- **(Low Variance)** The tree of depth 4 is robust to slight perturbations in the training data - the square carved out by the model is stable if you move the boundary points a bit.
- **(High Variance)** Trees of high depth are sensitive to perturbations in the training data, especially to changes in the boundary points.

Not surprisingly, complex trees have low bias (able to capture more complex geometry in the data) but high variance (can overfit). Complex trees are also harder to interpret and more computationally expensive to train.

Stopping Conditions

Common simple stopping conditions:

- Don’t split a region if all instances in the region belong to the same class.
- Don’t split a region if the number of instances in the sub-region will fall below pre-defined threshold (**min_samples_leaf**).
- Don’t split a region if the total number of leaves in the tree will exceed pre-defined threshold.

The appropriate thresholds can be determined by evaluating the model on a held-out data set or, better yet, via cross-validation.

Stopping Conditions

More restrictive stopping conditions:

- Don't split a region if the class distribution of the training points inside the region are independent of the predictors.
- Compute the gain in purity, information or reduction in entropy of splitting a region *R* into R_1 and R_2*:*

$$Gain(R) = \Delta(R) = m(R) - \frac{N_1}{N} m(R_1) - \frac{N_2}{N} m(R_2)$$

where *m* is a metric like the Gini Index or entropy. Don't split if the gain is less than some pre-defined threshold (**min_impurity_decrease**).

Alternative to Using Stopping Conditions

What is the major issue with pre-specifying a stopping condition?

- you may stop too early or stop too late.

How can we fix this issue?

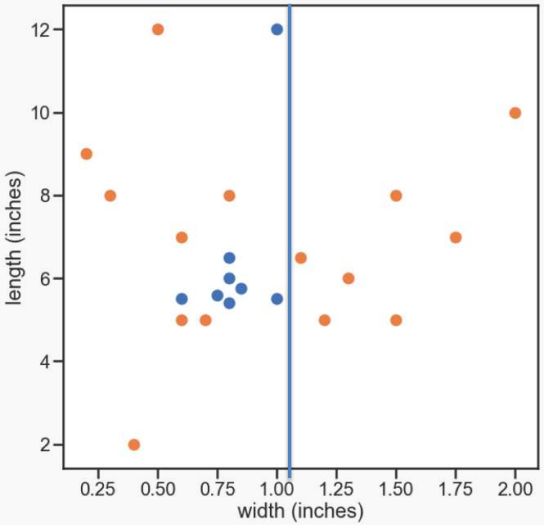
- choose several stopping criterion (set minimal Gain(R) at various levels) and cross-validate which is the best.

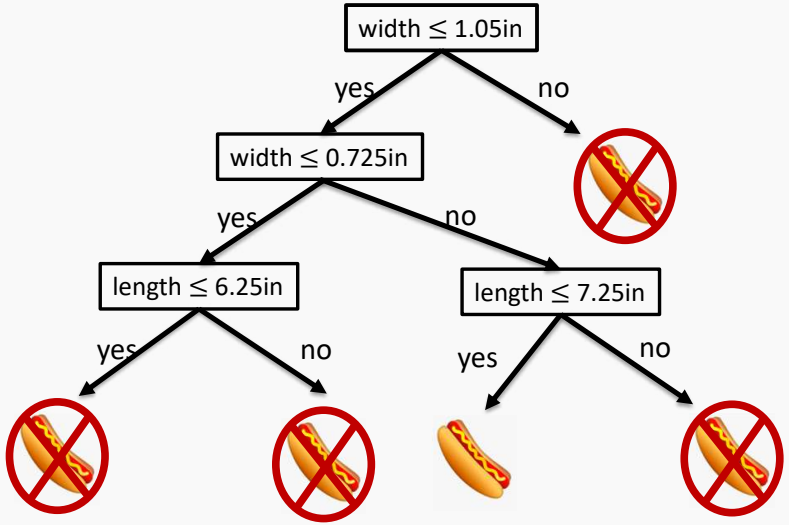
What is an alternative approach to this issue?

- Don’t stop. Instead prune back!

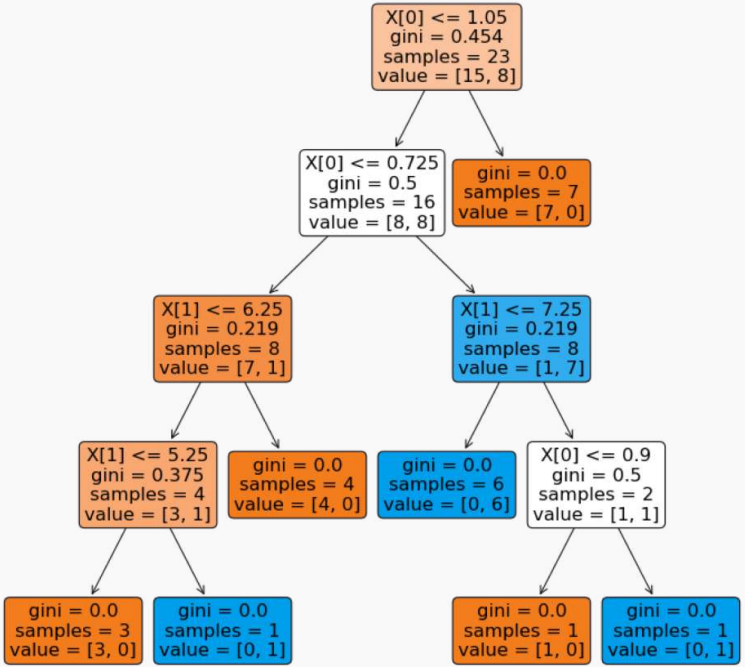
To Hot Dog or Not Hot Dog…

Hot Dog or Not

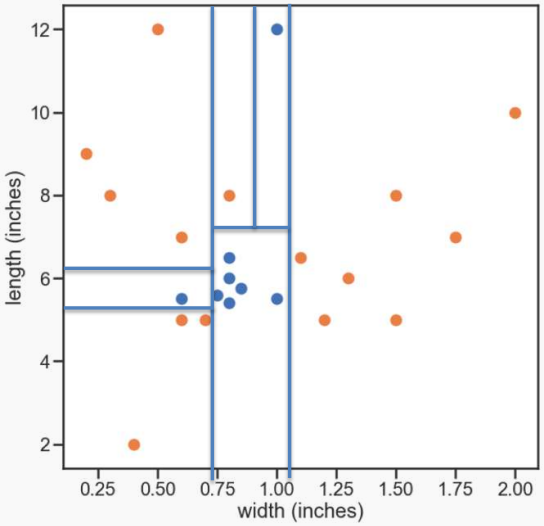


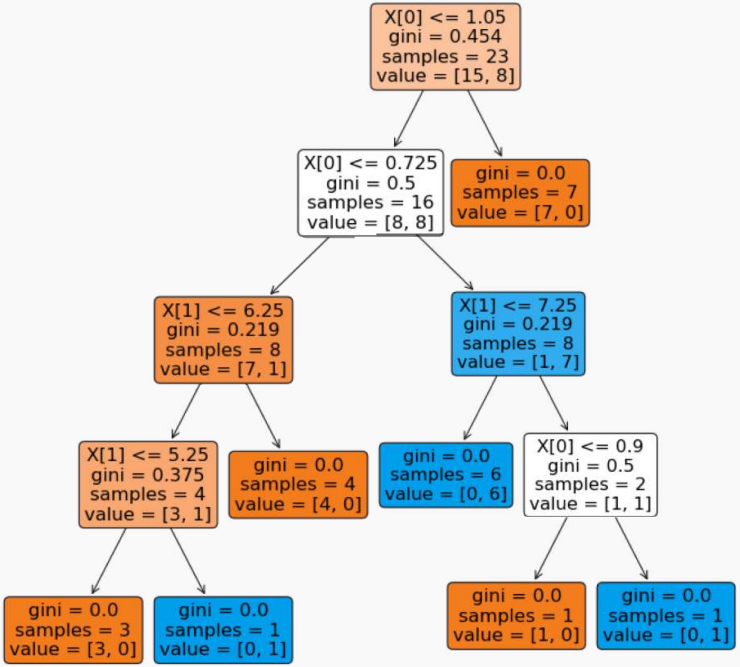


Motivation for Pruning

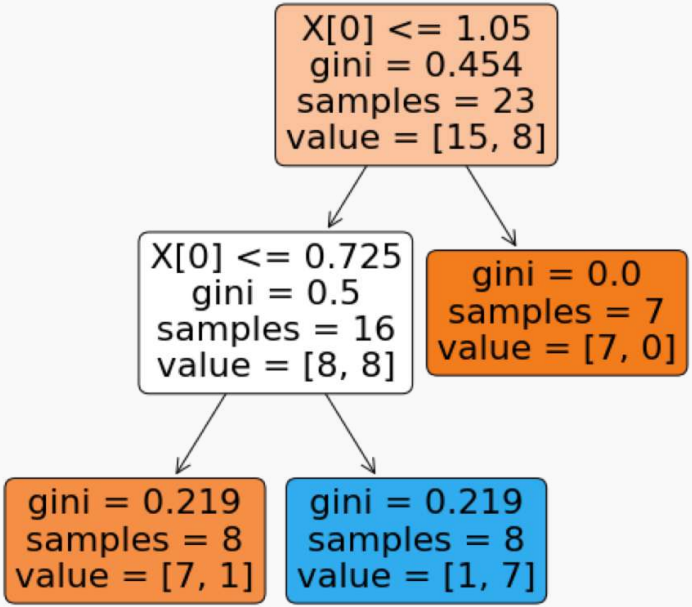


Hot Dog or Not

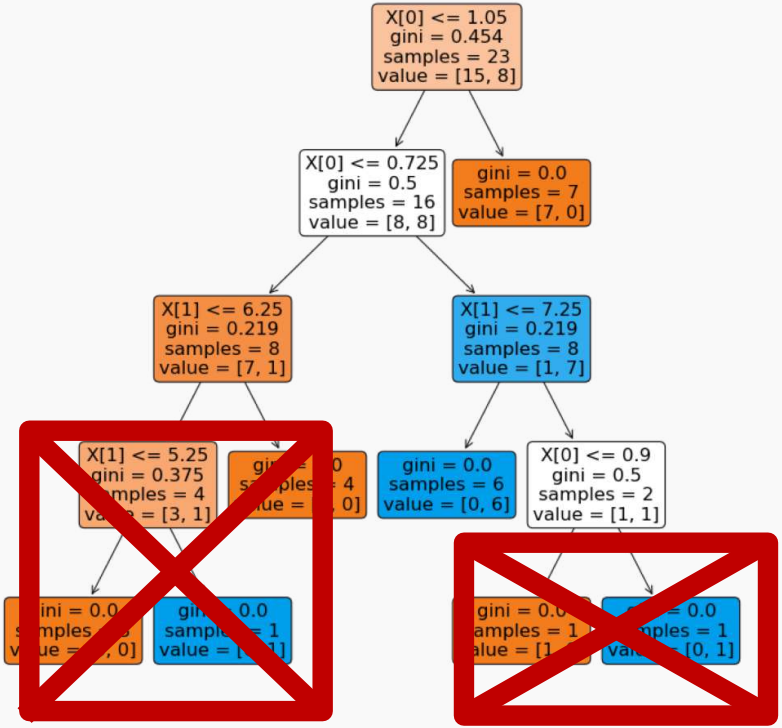




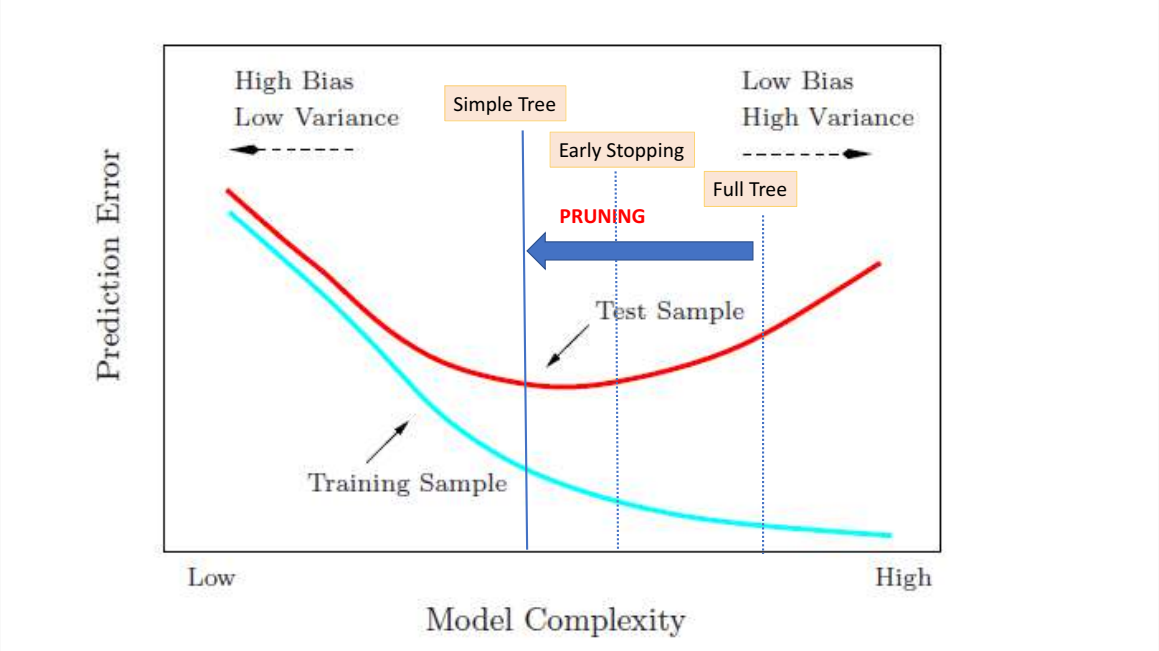
Motivation for Pruning



Motivation for Pruning



Motivation for Pruning



Pruning

Rather than preventing a complex tree from growing, we can obtain a simpler tree by ‘pruning’ a complex one.

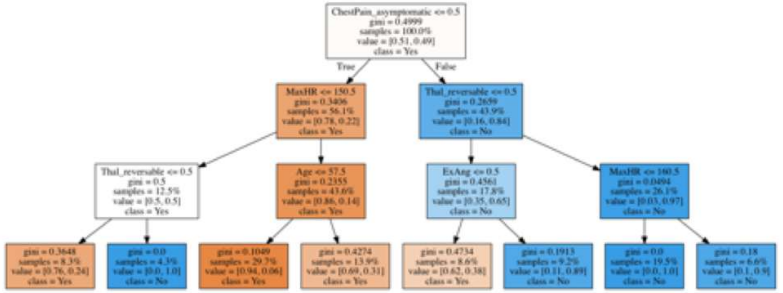
There are many method of pruning, a common one is ***cost complexity pruning***, where by we select from a array of smaller subtrees of the full model that optimizes a balance of performance and efficiency.

That is, we measure

$$C(T) = Error(T) + \alpha|T|$$

where *T* is a decision (sub) tree, $|T|$ is the number of leaves in the tree and α is the parameter for penalizing model complexity.

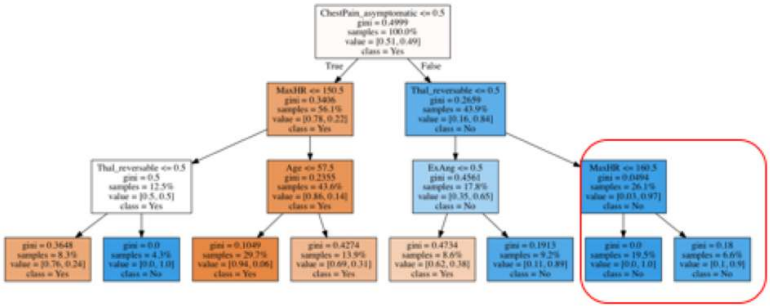
Pruning



$\alpha = 0.2$

Tree	Error	Num Leaves	Total

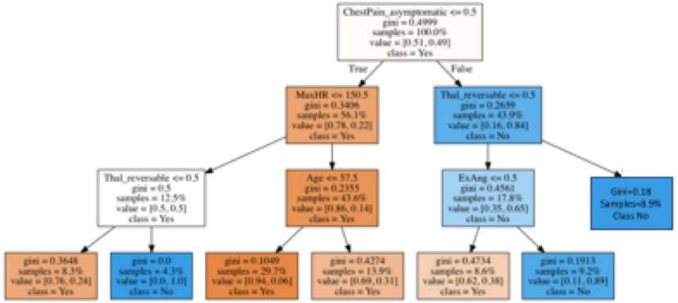
Pruning



$\alpha = 0.2$

Tree	Error	Num Leaves	Total
T	0.32	8	1.92

Pruning



$\alpha = 0.2$

Tree	Error	Num Leaves	Total
T	0.32	8	1.92
Tsmall	0.33	7	1.73

Smaller tree has larger error but less cost complexity score

Pruning

$$C(T) = Error(T) + \alpha|T|$$

1. Fix α.

2. Find best tree for a given α and based on cost complexity *C*.

3. Find best α using CV (what should be the error measure?)

Pruning

The pruning algorithm:

1. Start with a full tree T_0 (each leaf node is pure)
2. Replace a subtree in T_0 with a leaf node to obtain a pruned tree T_1. This subtree should be selected to minimize

$$\frac{Error(T_0) - Error(T_1)}{|T_0| - |T_1|}$$

3. Iterate this pruning process to obtain $T_0, T_1, \ldots, T_L$ where T_L is the tree containing just the root of T_0
4. Select the optimal tree T_i by cross validation.

Note: you might wonder where we are computing the cost-complexity $C(T_l)$. One can prove that this process is equivalent to explicitly optimizing *C* at each step.

Next

How can this decision tree approach apply to a ***regression problem*** (quantitative outcome)?

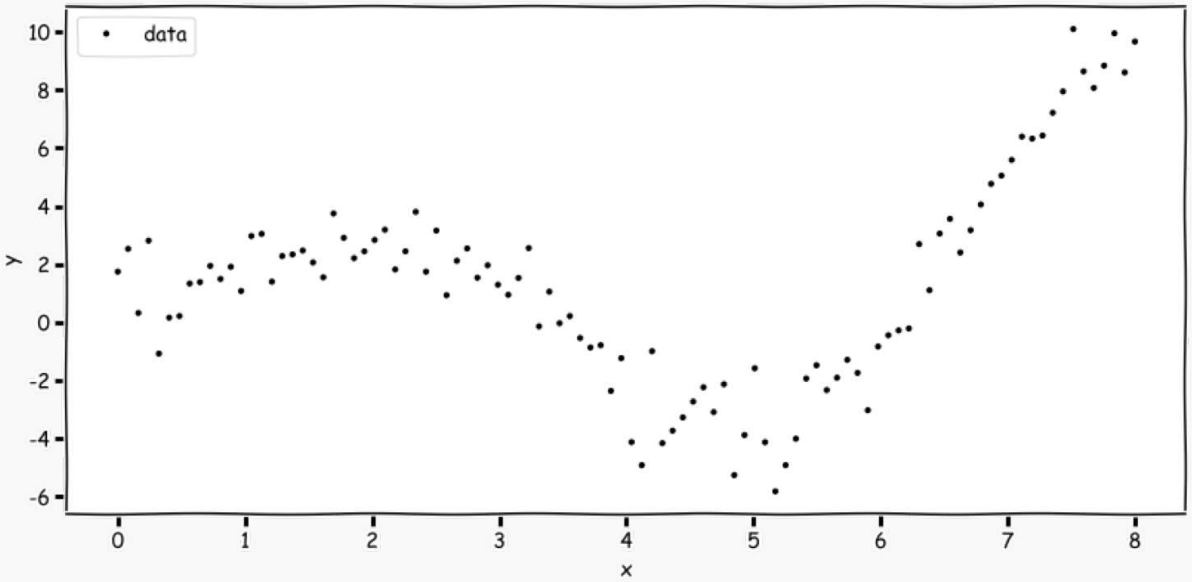
Questions to consider:

- What would be a reasonable loss function?
- How would you determine any splitting criteria?
- How would you perform prediction in each leaf?

A picture is worth a thousand words…

Regression Tree Example

How do we decide a split here?



Decision Trees for Regression

Adaptations for Regression

With just two modifications, we can use a decision tree model for regression:

1. The three splitting criteria we've examined each promoted splits that were pure - new regions increasingly specialized in a single class.

    A. **For classification**, purity of the regions is a good indicator the performance of the model.

    B. **For regression**, we want to select a splitting criterion that promotes splits that improves the predictive accuracy of the model as measured by, say, the MSE.

2. For regression with output in $\mathbb{R}$, we want to label each region in the model with a real number - typically the average of the output values of the training points contained in the region.

Learning Regression Trees

The learning algorithms for decision trees in regression tasks is:

1. Start with an empty decision tree (undivided features pace)
2. Choose a predictor j on which to split and choose a threshold value t_j for splitting such that the weighted average MSE of the new regions as smallest possible:

$$\underset{j,t_j}{\operatorname{argmin}} \left\{ \frac{N_1}{N} \operatorname{MSE}(R_1) + \frac{N_2}{N} \operatorname{MSE}(R_2) \right\}$$

   or equivalently,

$$\underset{j,t_j}{\operatorname{argmin}} \left\{ \frac{N_1}{N} \operatorname{Var}(y|x \in R_1) + \frac{N_2}{N} \operatorname{Var}(y|x \in R_2) \right\}$$

   where N_i is the number of training points in R_i and N is the number of points in R.

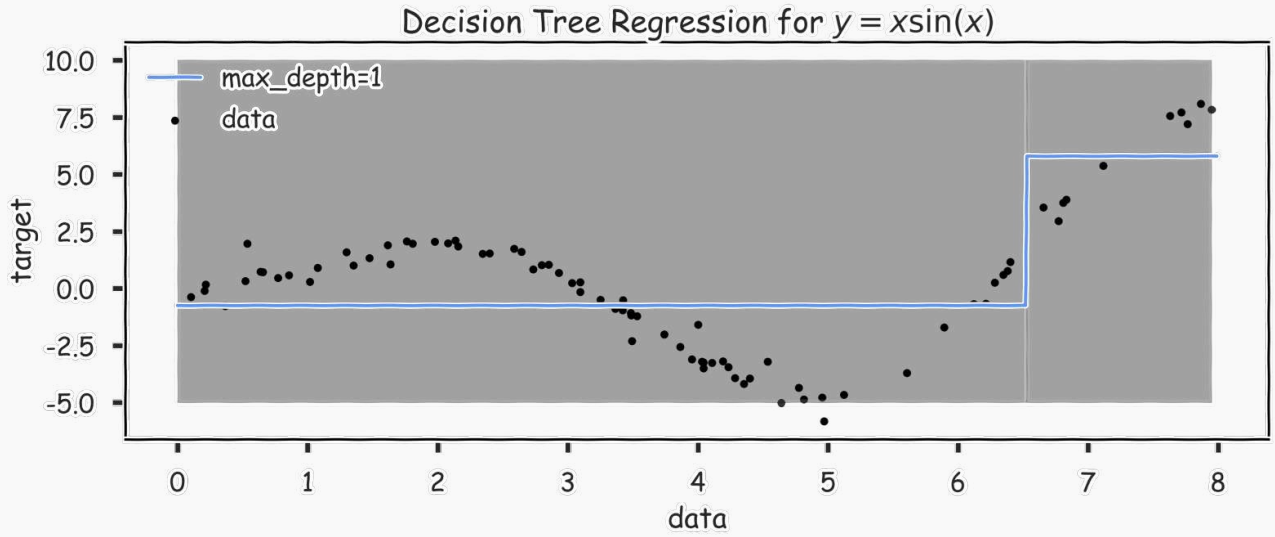
3. Recurse on each new node until ***stopping condition*** is met.

Regression Trees Prediction

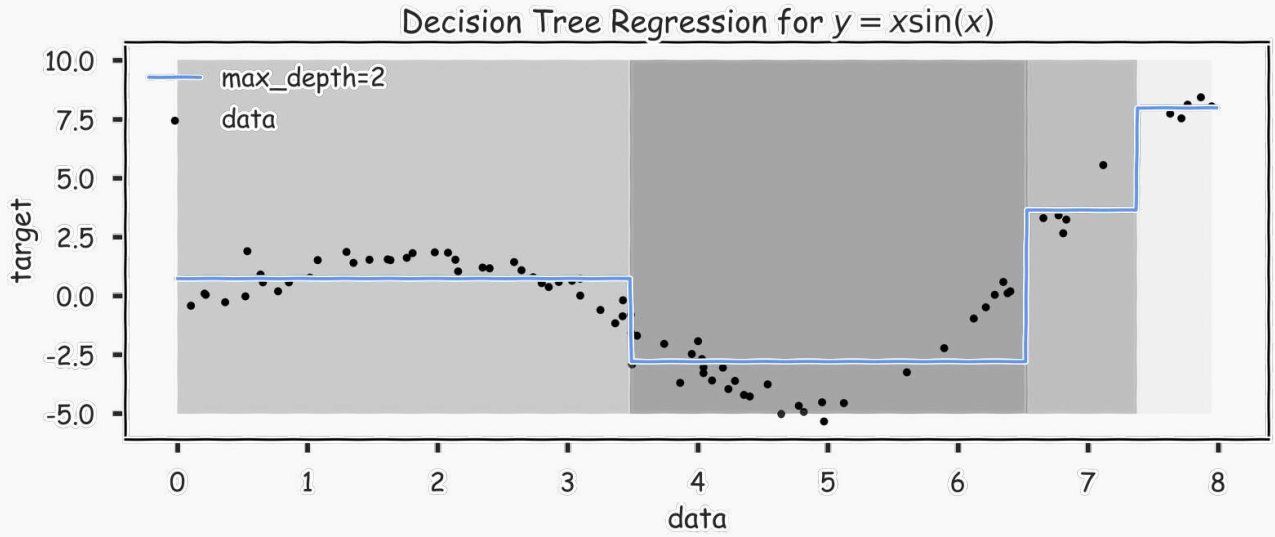
For any data point x_i

1. Traverse the tree until we reach a leaf node.
2. Averaged value of the response variable y's in the leaf (this is from the training set) is the $\widehat{y_i}$.

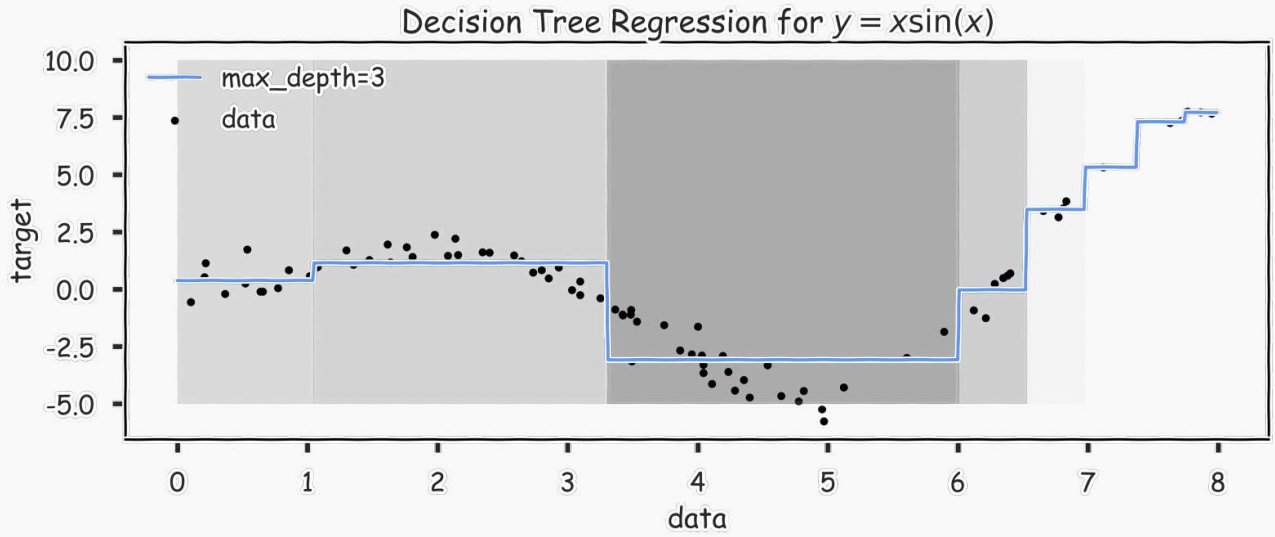
Regression Trees Prediction (grey scale represents MSE)



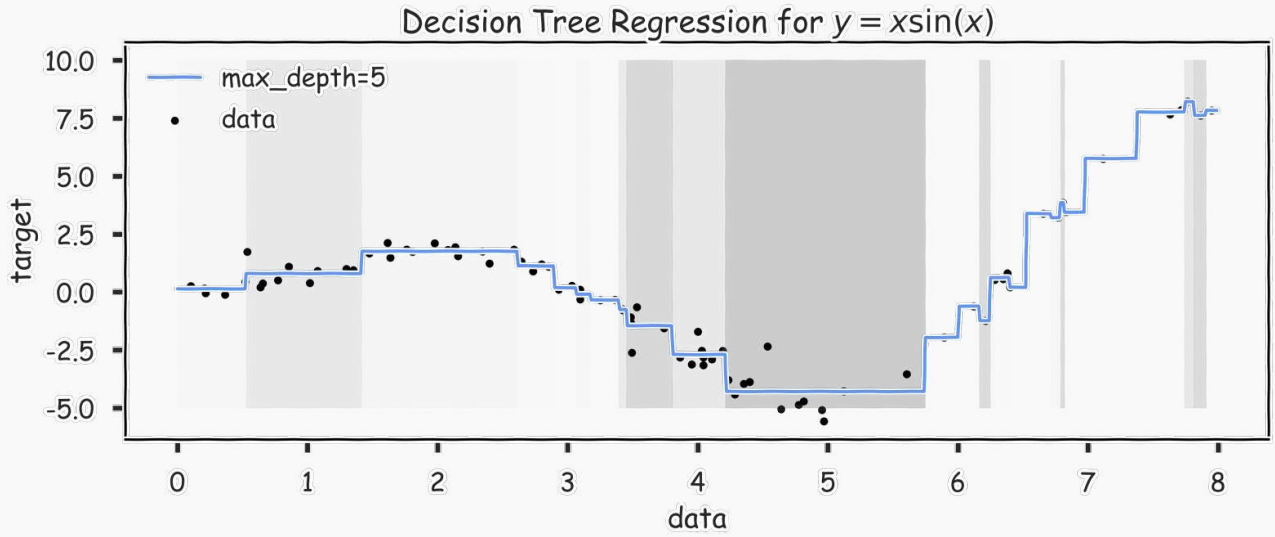
Regression Trees Prediction (grey scale represents MSE)



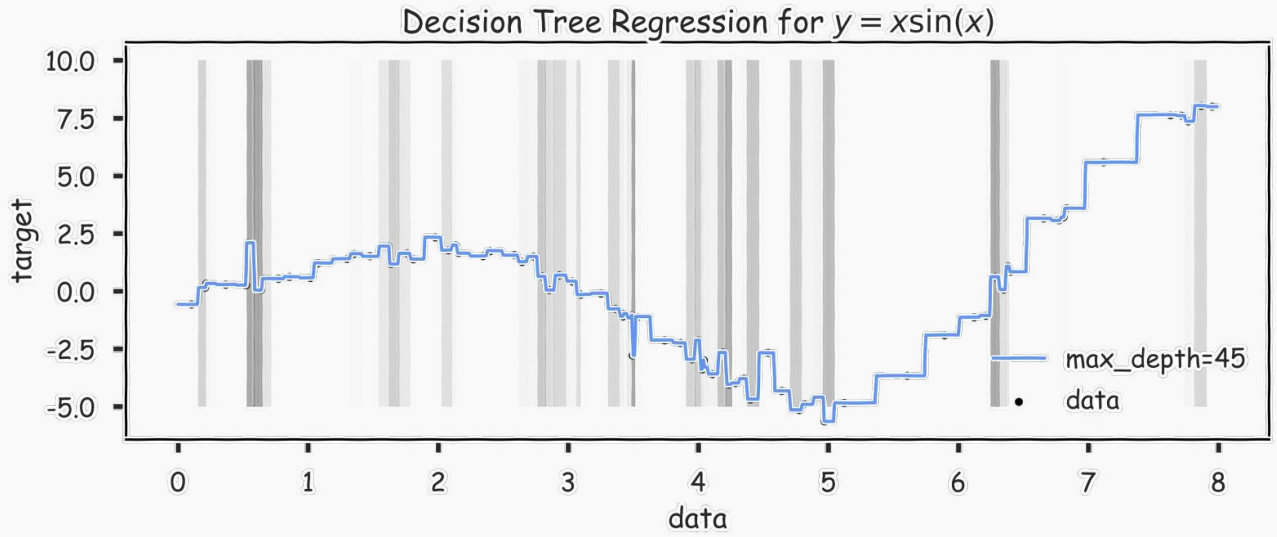
Regression Trees Prediction (grey scale represents MSE)

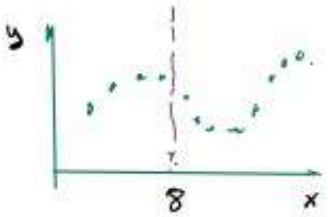


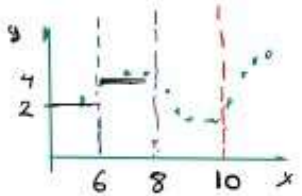
Regression Trees Prediction (grey scale represents MSE)

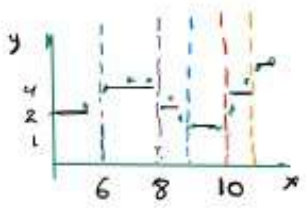


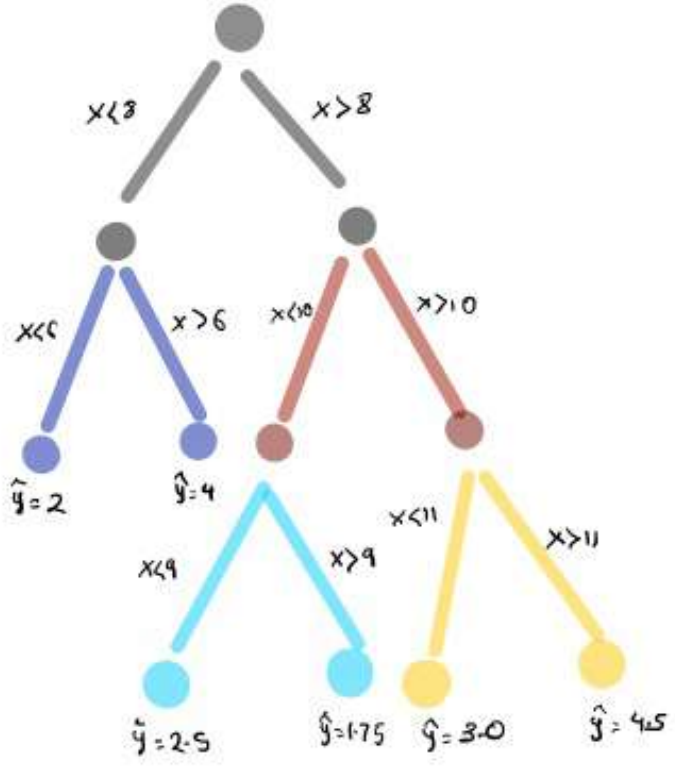
Regression Trees Prediction (grey scale represents MSE)




y
8
x


y
4
2
6
8
10
x


y
4
2
1
6
8
10
x


x<8
x>8
x<6
x>6
x<10
x>10
ŷ=2
ŷ=4
x<9
x>9
x<11
x>11
ŷ=2.5
ŷ=1.75
ŷ=3.0
ŷ=4.5

Stopping Conditions

Most of the stopping conditions, like maximum depth or minimum number of points in region, we saw for classification trees can still be applied.

In the place of purity gain, we can instead compute accuracy gain for splitting a region R

$$\text{Gain}(R) = \Delta(R) = MSE(R) - \frac{N_1}{N} MSE(R_1) - \frac{N_2}{N} MSE(R_2)$$

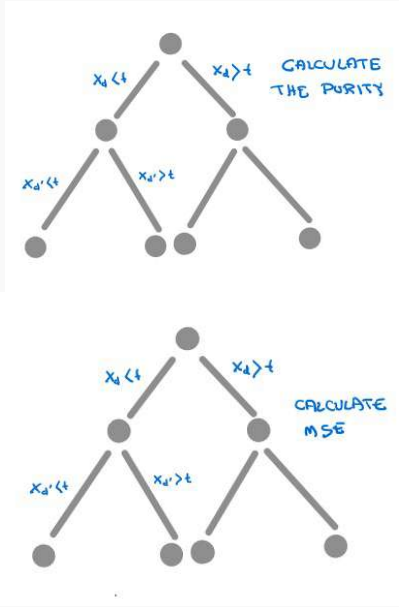
and stop the tree when the gain is less than some pre-defined threshold.

Decision Trees Summary: Splitting Criteria

Splitting Criteria:

For classification, purity of the regions is a good indicator the performance of the model. Entropy as a splitting criterial minimizes the cross-entropy (greedy). Gini is also a splitting criteria.

For regression, we want to select a splitting criterion that promotes splits that improves the predictive accuracy of the model as measured by the MSE

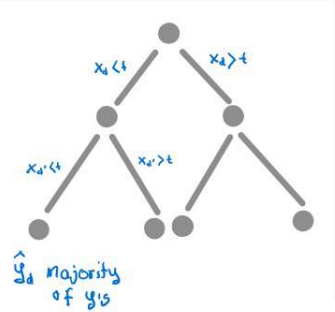


Decision Trees Summary : Prediction

Prediction:

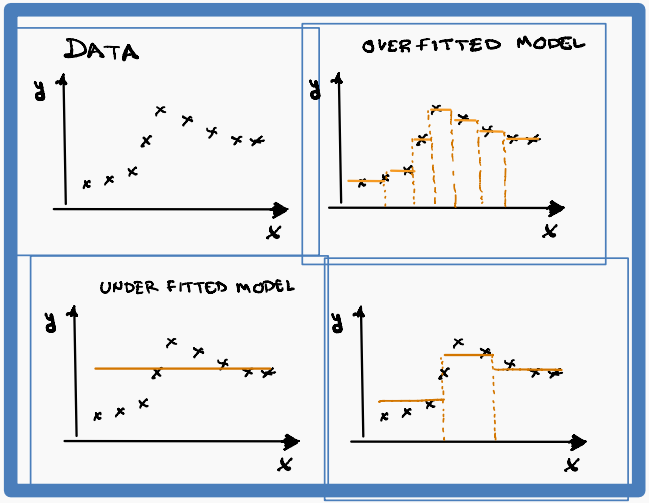
For classification, we label each region in the model with the label of the class to which the **plurality** of the points within the region belong.

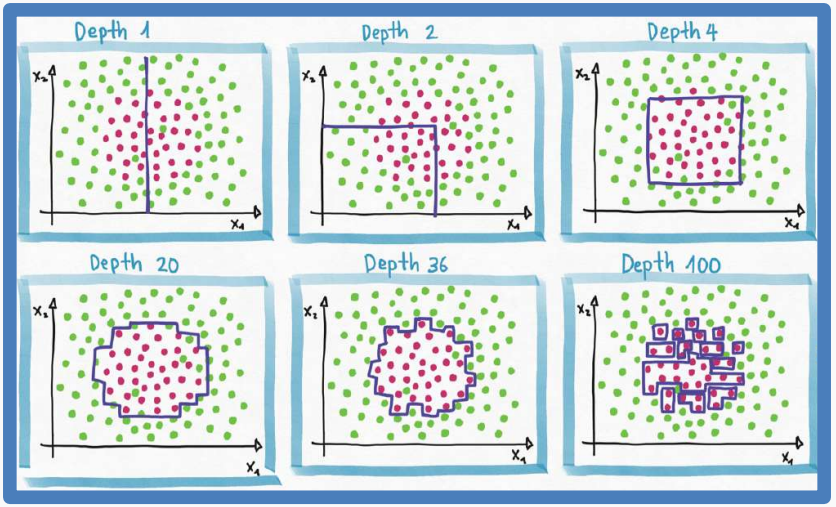
For regression, we predict with the **average** of the output values of the training points contained in the region.



Decision Trees Summary : Overfitting

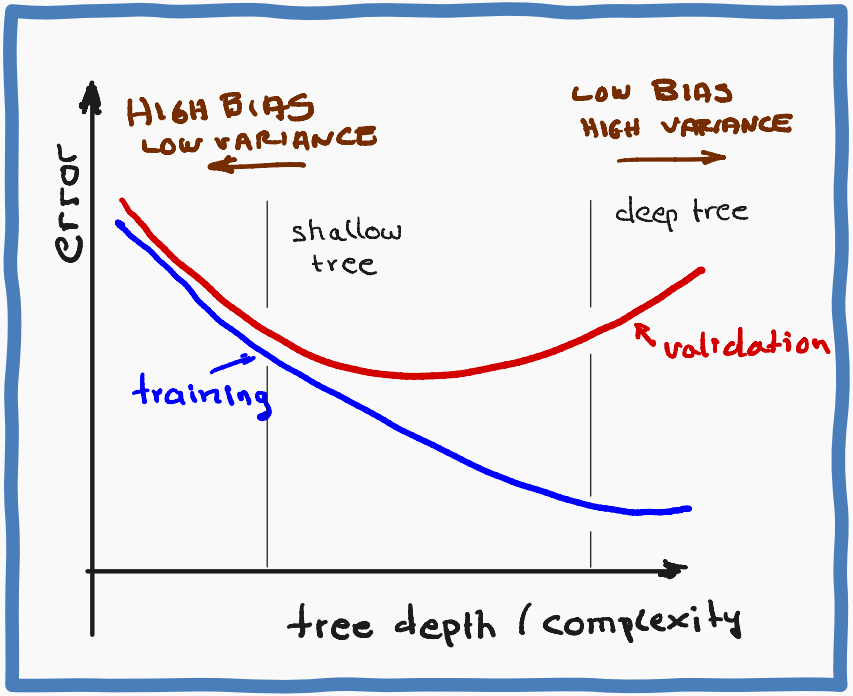
When tree is too shallow, it cannot divide the input data into enough regions, so the model underfits. When the tree is too deep it cuts the input space into too many regions and fit to the noise of the data -> overfits.



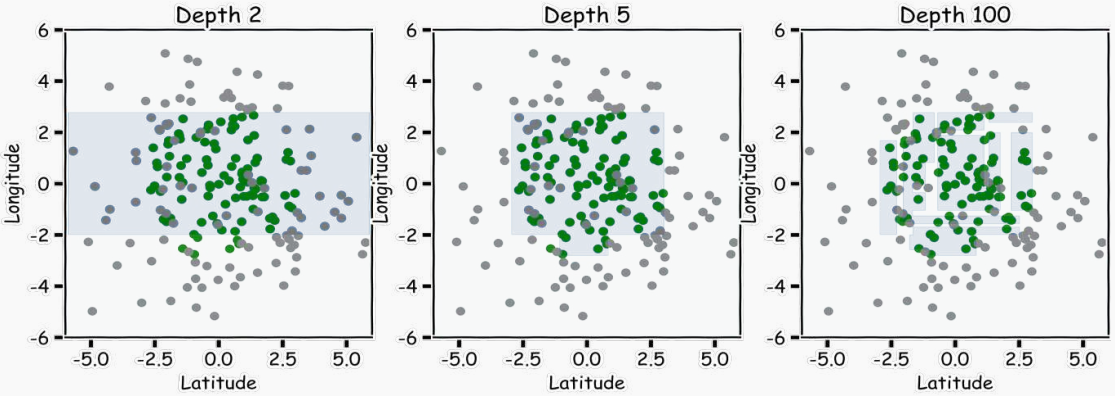


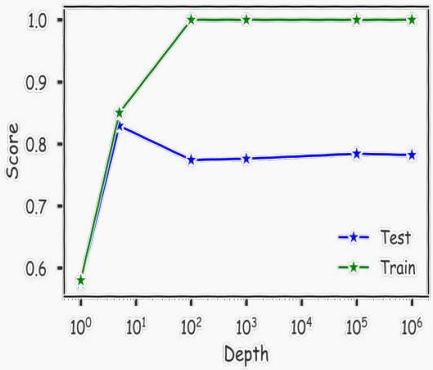
Decision Trees Summary : Overfitting

Avoid overfitting by pruning or limiting the depth of the tree and using CV.



Reduce the variance: Depth of the tree





We've seen that large trees have high variance and are prone to overfitting.

Use train/validation or cross validation to estimate the best depth.

Limitations of Decision Tree Models

Decision trees models are highly interpretable and fast to train, using our greedy learning algorithm.

However, to **capture a complex decision boundary** (or approximate a complex function), we need to use a large tree (since each time we can only do axis-aligned splits).

We've seen that large trees have high variance and are prone to overfitting.

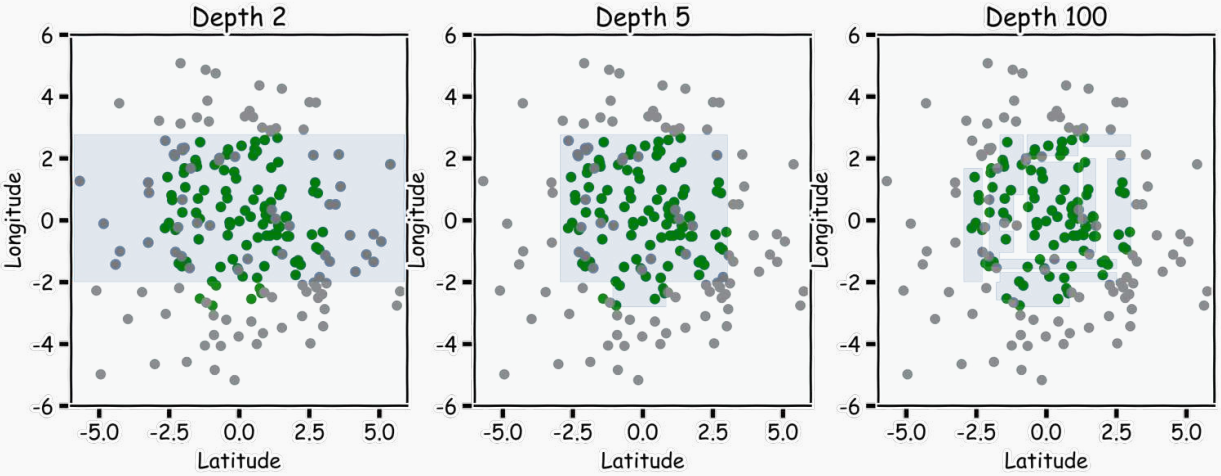
For these reasons, in practice, decision tree models often underperform when compared with other classification or regression methods.

Bagging

Outline

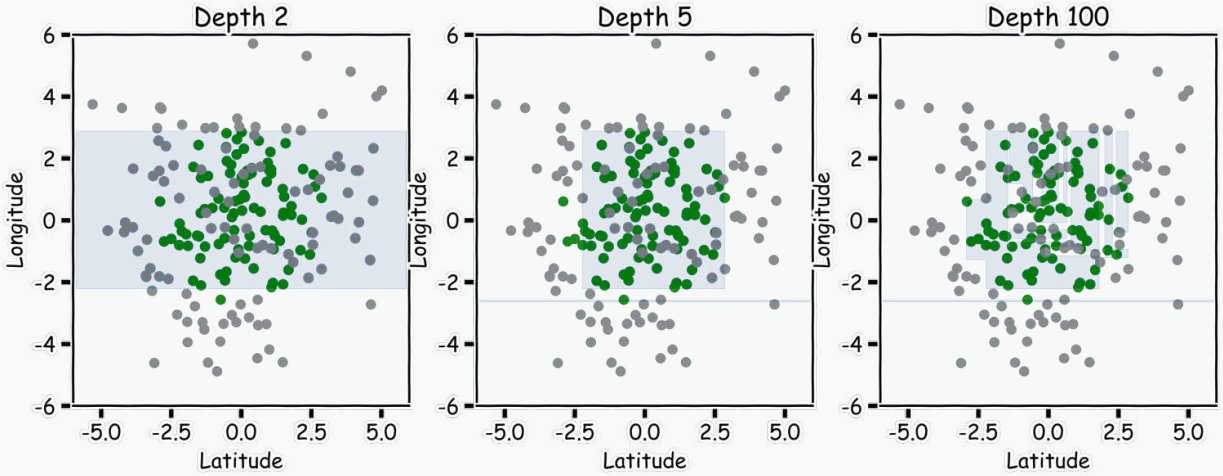
- Review of Decision Trees
- **Bagging**
- Out of Bag Error (OOB)
- Variable Importance

Bagging

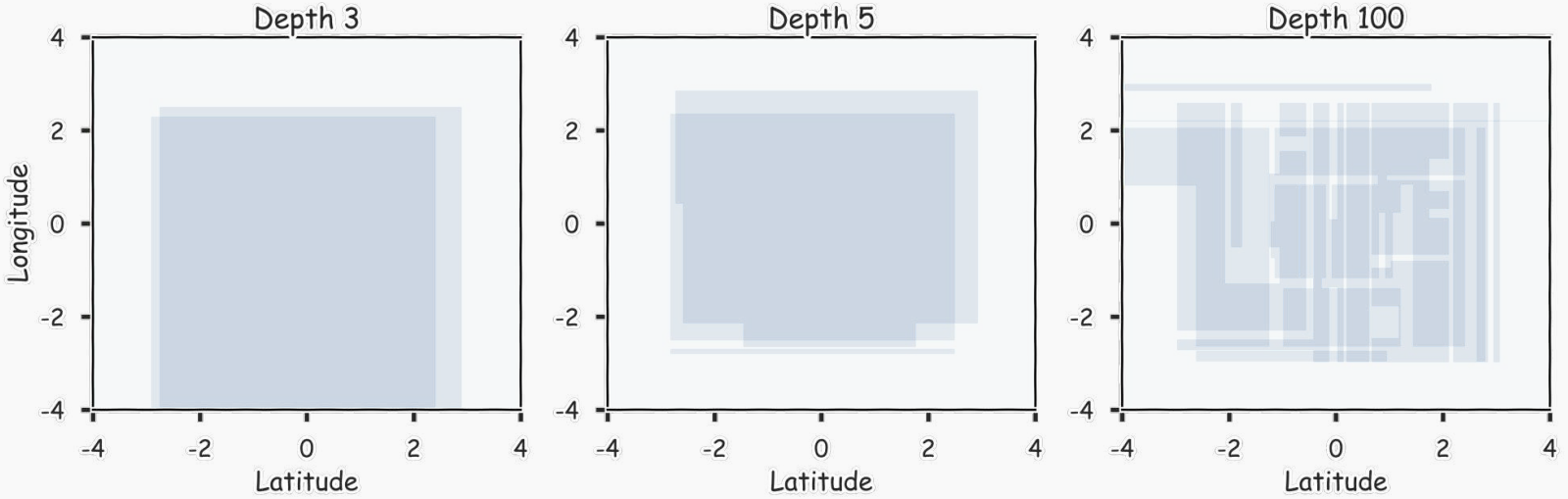

Depth 2
Depth 5
Depth 100
Longitude
Latitude
6
4
2
0
-2
-4
-6
-5.0
-2.5
0.0
2.5
5.0

Bagging

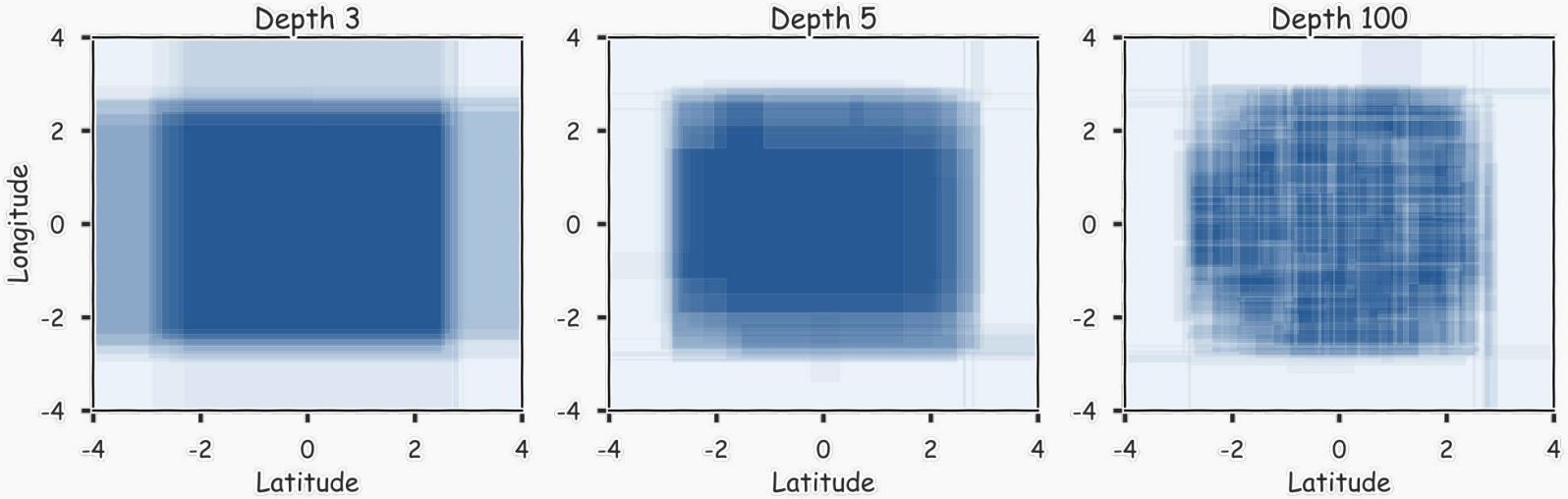
Magic realism -> bootstrap



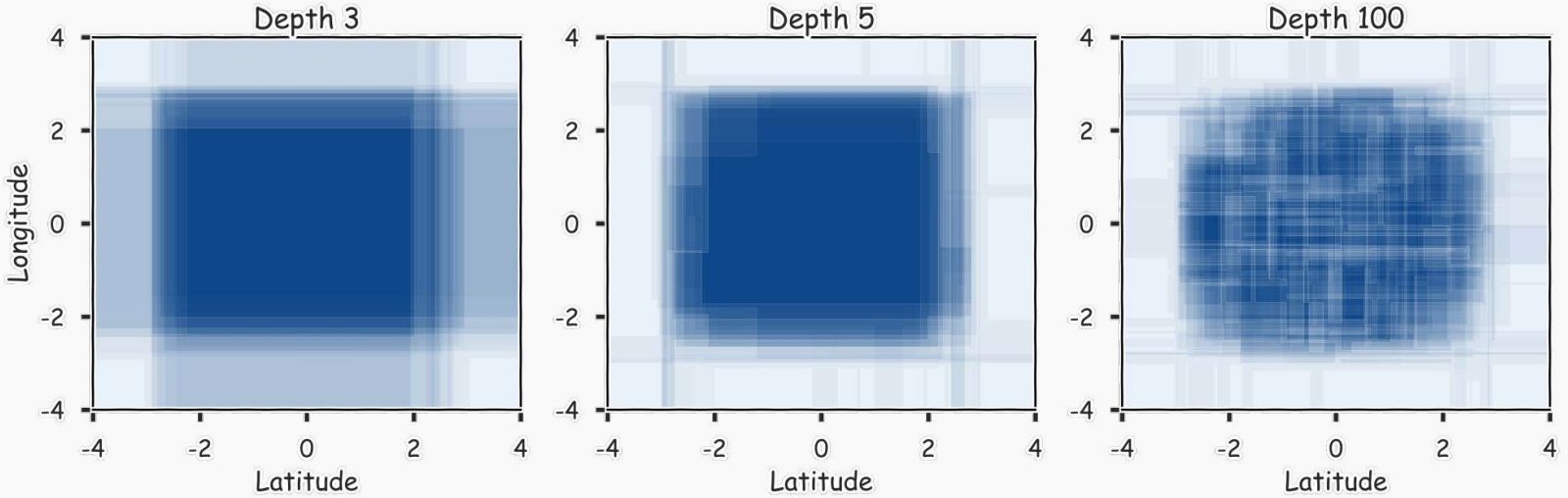
Combine them? 2 magic realisms



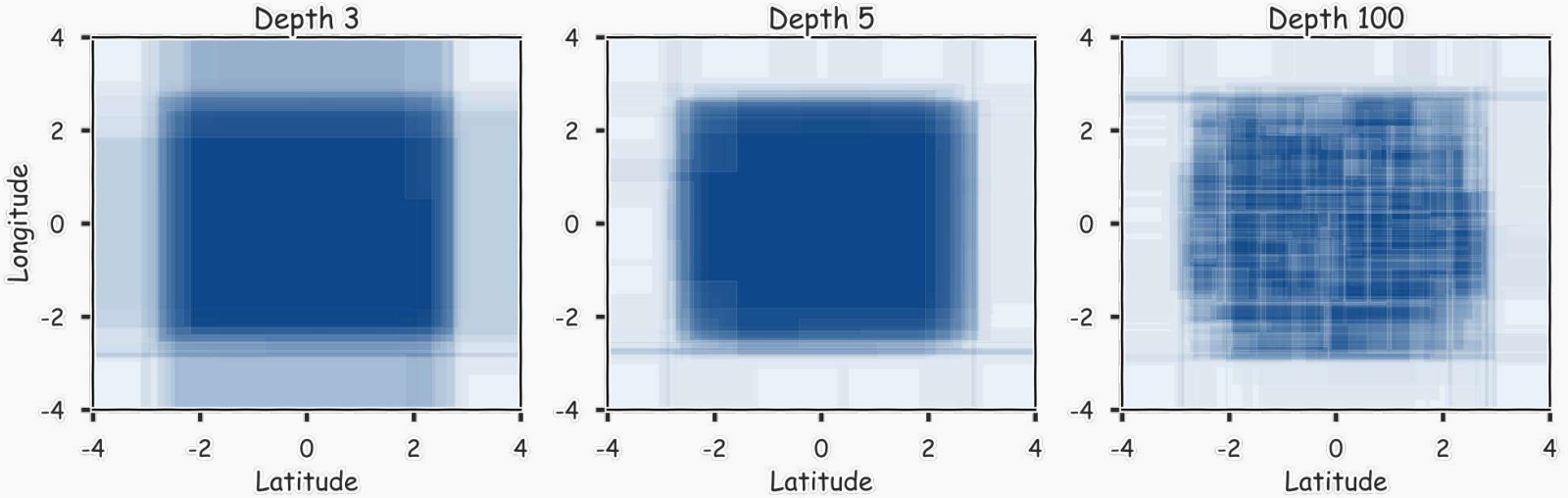
20 magic realisms



100 magic realisms



300 magic realisms



Bagging

One way to adjust for the high variance of the output of an experiment is to perform the experiment multiple times and then average the results.

The same idea can be applied to high variance models:

1. **Bootstrap:** we generate multiple samples of training data, via bootstrapping. We train a deeper decision tree on each sample of data.
2. **Aggregate:** for a given input, we output the averaged outputs of all the models for that input.

This method is called ***Bagging*** (Breiman, 1996), short for, of course, **B**ootstrap **Agg**regating.

For classification, we return the class that is outputted by the plurality of the models. For regression we return the average of the outputs for each tree.

Bagging

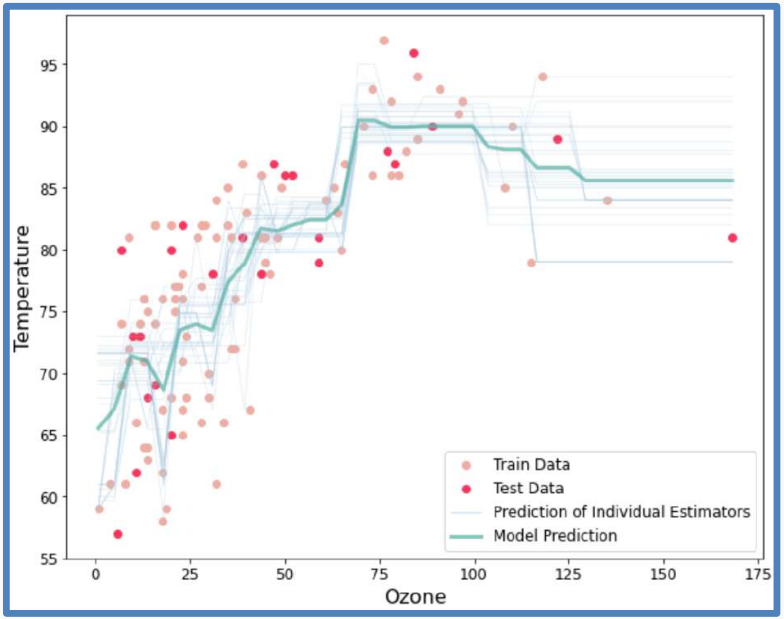
Bagging enjoys the benefits of:

1. High expressiveness - by using deeper trees each model is able to approximate complex functions and decision boundaries.

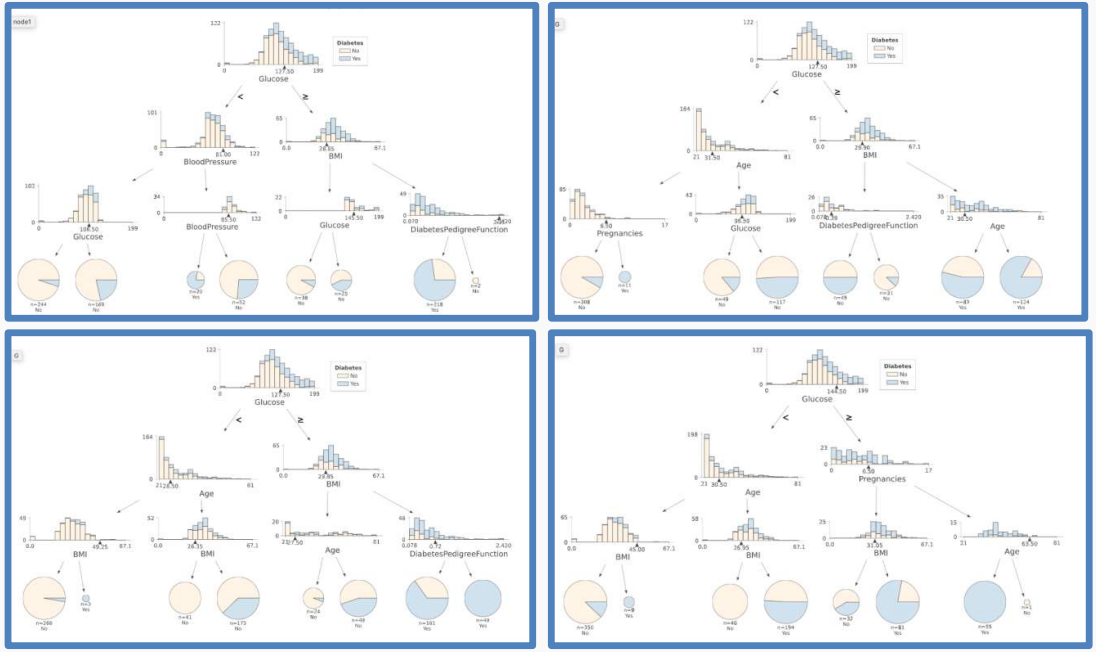
2. Low variance - averaging the prediction of all the models reduces the variance in the final prediction, assuming that we choose a sufficiently large number of trees.

Bagging (regression)

The resulting tree is the average of all tree (estimators).



Bagging (classification)



For each bootstrap, we build a decision tree. The results is a combination (majority) of the predictions from all trees.

Bagging

Question: Do you see any problems?

Bagging

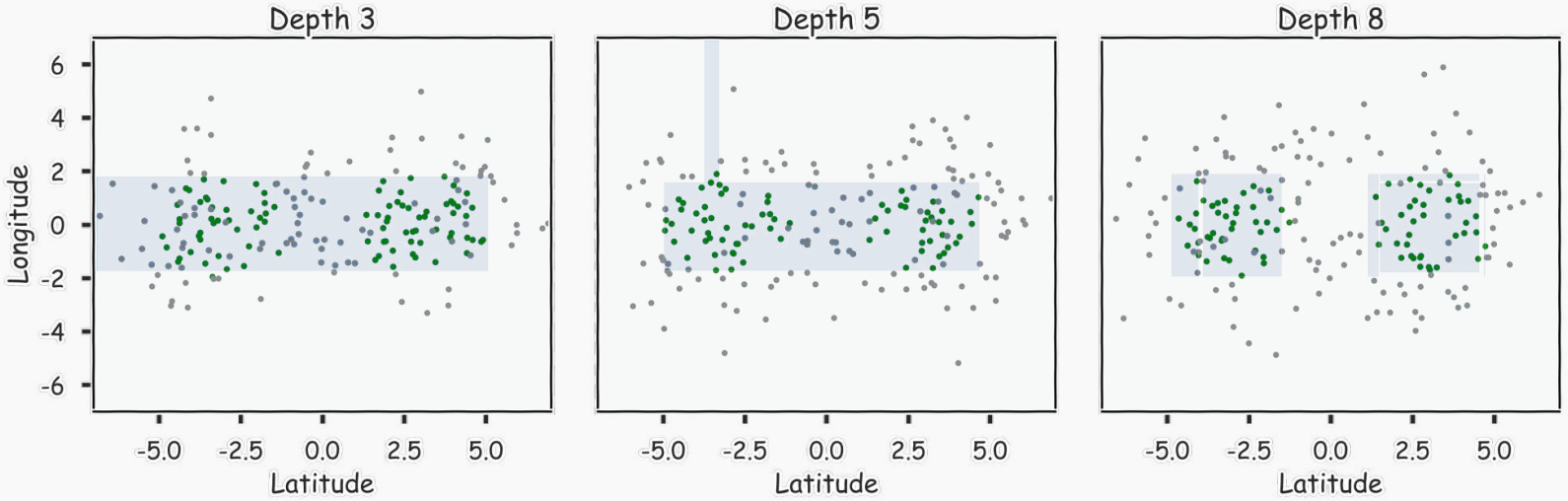
Question: Do you see any problems?

- If trees are too shallow it can still underfit.
- Still some overfitting if the trees are too large.
- Interpretability:

  The **major drawback** of bagging (and other ***ensemble methods*** that we will study) is that the averaged model is no longer easily interpretable - i.e. one can no longer trace the 'logic' of an output through a series of decisions based on predictor values!

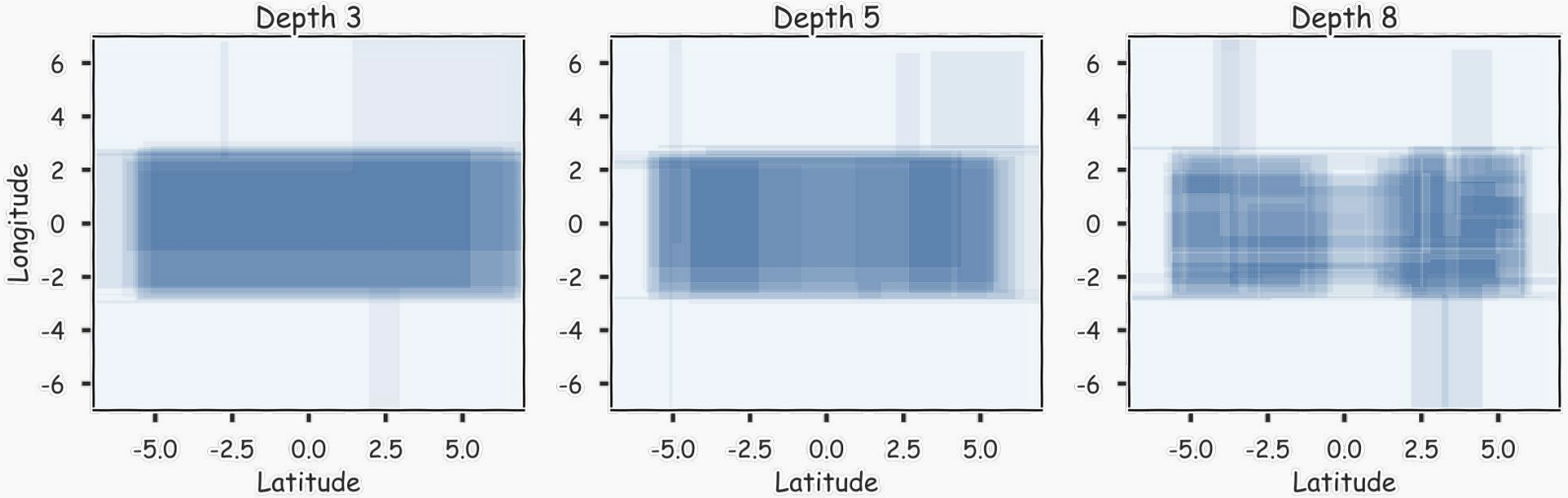
Case of underfitting

Consider the dataset below. To capture the pattern we need deeper tree.



Case of underfitting

Here we fit 100 trees using bootstrapped samples. Even with multiple estimators, the shallow tree will not be able to capture the real pattern.



Question: Do you see any problems?

- If trees are too shallow it can still underfit.
- Still some overfitting if the trees are too large.

Cross Validation

Outline

- Review of Decision Trees
- Bagging
- **Out of Bag Error (OOB)**
- Variable Importance

Bagging

Original Data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
⋮	⋮
X_n	y_n

Bootstrap Sample 1

X	Y
X_4	y_4
X_{14}	y_{14}
X_{11}	y_{11}
X_2	y_2
X_{35}	y_{35}
⋮	⋮
X_k	y_k

Decision Tree 1

Used and unused data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
⋮	⋮
X_n	y_n

Bagging

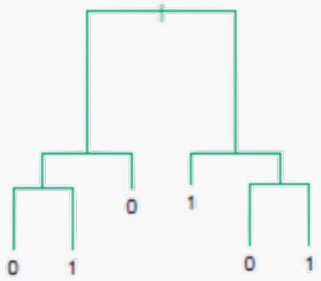
Original Data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
⋮	⋮
X_n	y_n

Bootstrap Sample 2

X	Y
X_5	y_5
X_3	y_3
X_{12}	y_{12}
X_{43}	y_{43}
X_1	y_1
⋮	⋮
X_k	y_k

Decision Tree 2



Used and unused data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
⋮	⋮
X_n	y_n

Bagging

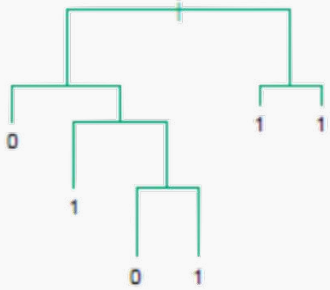
Original Data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
⋮	⋮
X_n	y_n

Bootstrap Sample 3

X	Y
X_9	y_9
X_4	y_4
X_1	y_1
X_1	y_1
X_{65}	y_{65}
⋮	⋮
X_k	y_k

Decision Tree 3



Used and unused data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
⋮	⋮
X_n	y_n

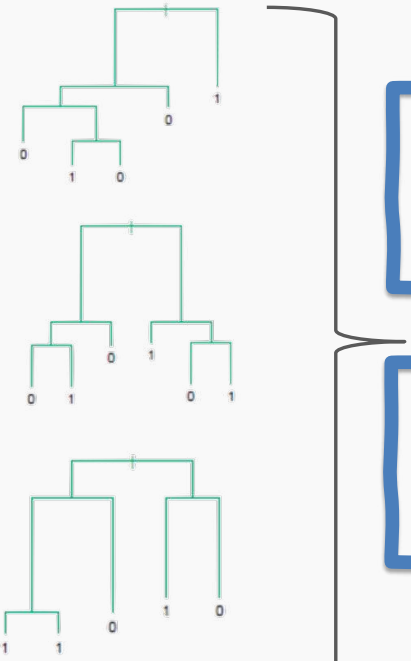
Point-wise out-of-bag error

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
⋮	⋮
X_i	y_i
⋮	⋮
X_n	y_n

Point-wise out-of-bag error

B Trees that did not see $\{X_i, y_i\}$

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
⋮	⋮
X_i	y_i
⋮	⋮
X_n	y_n



Classification

$$\hat{y}_{i,pw} = \text{majority}(\hat{y}_i) \qquad e_i = \mathbb{I}(\hat{y}_{i,pw} \neq y_i)$$

Regression

$$\hat{y}_{i,pw} = \frac{1}{B}\sum_{j \in B} \hat{y}_{i,j} \qquad e_i = \left(y_i - \hat{y}_{i.pw}\right)^2$$

OOB Error

We average the point-wise out-of-bag error over the full training set.

Classification

$$Error_{OOB} = \frac{1}{B}\sum_{i}^{B} e_i = \frac{1}{B}\sum_{i}^{B} \mathbb{I}(\hat{y}_{i,pw} \neq \mathrm{y_i})$$

Regression

$$Error_{OOB} = \frac{1}{B}\sum_{i}^{B} e_i = \frac{1}{B}\sum_{i}^{B} \left(y_i - \hat{y}_{i,pw}\right)^2$$

Out-of-Bag Error

Bagging is an example of an ***ensemble method***, a method of building a single model by training and aggregating multiple models.

With ensemble methods, we get a new metric for assessing the predictive performance of the model, the ***out-of-bag error***.

Given a training set and an ensemble of models, each trained on a bootstrap sample, we compute the ***out-of-bag error*** of the averaged model by

1. For each point in the training set, we average the predicted output for this point over the models whose bootstrap training set excludes this point. We compute the error or squared error of this averaged prediction. Call this the point-wise out-of-bag error.

2. We average the point-wise out-of-bag error over the full training set.

Bagging

Question: Do you see any problems?

- If trees are too shallow it can still underfit.
- Still some overfitting if the trees are too large.
- Interpretability:

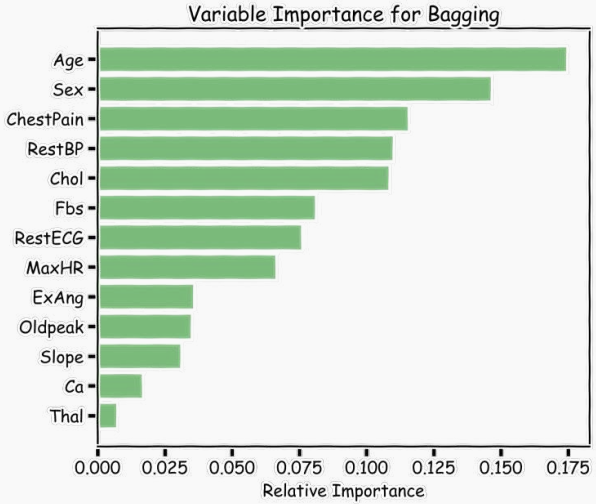
  The **major drawback** of bagging (and other ***ensemble methods*** that we will study) is that the averaged model is no longer easily interpretable - i.e. one can no longer trace the 'logic' of an output through a series of decisions based on predictor values!

Outline

- Review of Decision Trees
- Bagging
- Out of Bag Error (OOB)
- **Variable Importance**

Variable Importance for Bagging

Calculate the total amount that the MSE (for regression) or Gini index (for classification) is decreased due to splits over a given predictor, averaged over all B trees.



100 trees, max_depth=10

Improving on Bagging

In practice, the ensembles of trees in Bagging tend to be highly correlated.

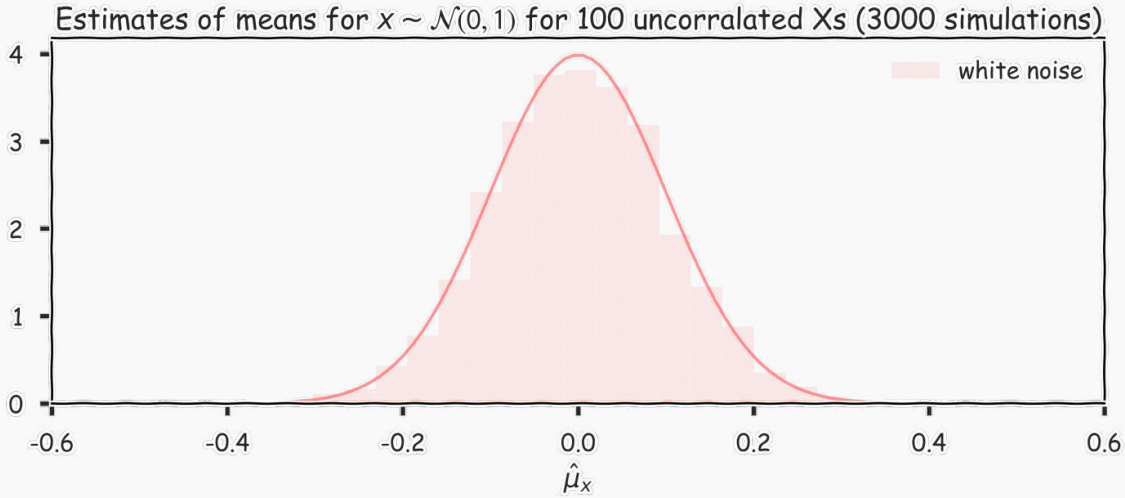
Suppose we have an extremely strong predictor, x_j , in the training set amongst moderate predictors. Then the greedy learning algorithm ensures that most of the models in the ensemble will choose to split on x_j in early iterations.

However, we assumed that each tree in the ensemble is independently and identically distributed, with the expected output of the averaged model the same as the expected output of any one of the trees.

Improving on Bagging

Recall, for B number of identically and independently distributed variable, X, with variance σ^2, the variance of the estimate of the mean is :

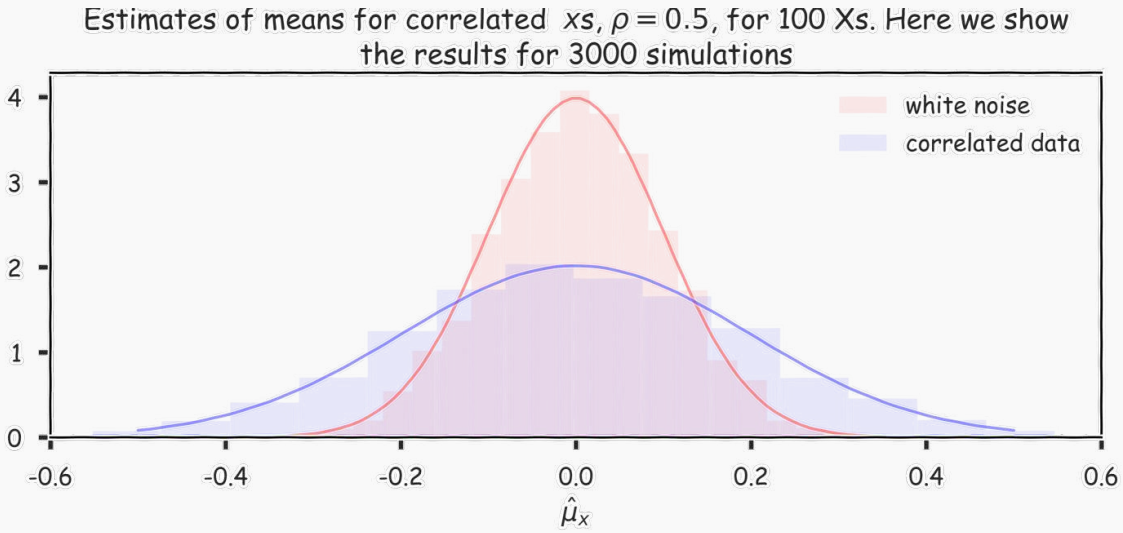
$$\text{var}(\hat{\mu}_x) = \frac{\sigma^2}{B}$$



Improving on Bagging

For B number of identically but not independently distributed variables with pairwise correlation ρ and variance σ^2, the variance of their mean is

$$\text{var}(\hat{\mu}_x) \propto \sigma^2(1+\rho^2)/B$$



Random Forests

Random Forests

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

To decorrelate 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.

3. 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 hyperparameters to tune:

1. The sampling scheme: number of predictors to randomly select at each split : `max_features {“auto”, “sqrt”, “log2”}, int or float, default=”auto”.`

2. The total number of trees in the forest: `n_estimators, int, default=100`

3. The complexity of each tree: stop when a leaf has <= min_samples_leaf samples or when we reach a certain max_depth.

4. 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 is not unusual.

Tuning Random Forests

When the number of predictors is large, but the number of relevant predictors is small, you need to set `max_features` to a larger number.

Question: Why?

If chosen features is small, in each split, the chances of selecting a relevant predictor will be low and hence most trees in the ensemble will be weak models.

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 the use of out-of-bag (**OOB)** (making them data and problem dependent).

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

– $\sqrt{N_j}$ for classification

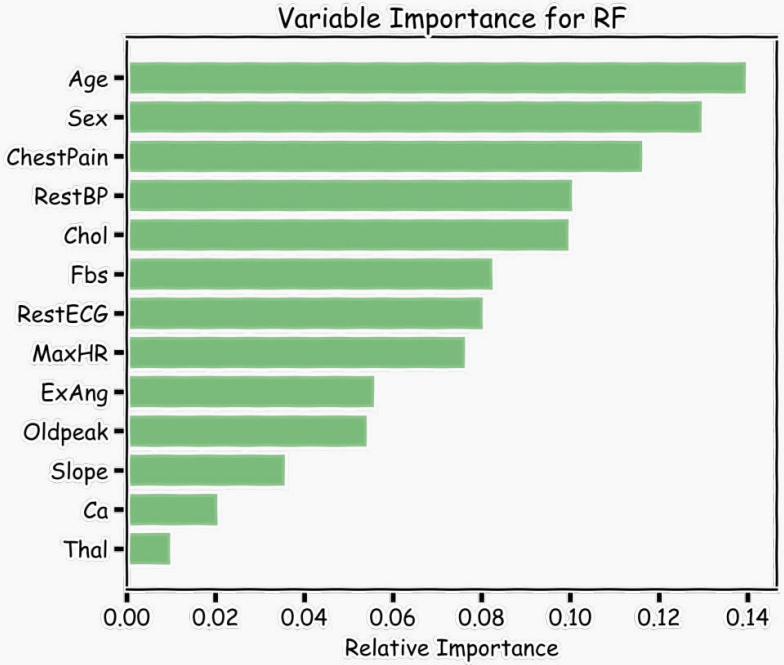
– $\frac{N}{3}$ for regression

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

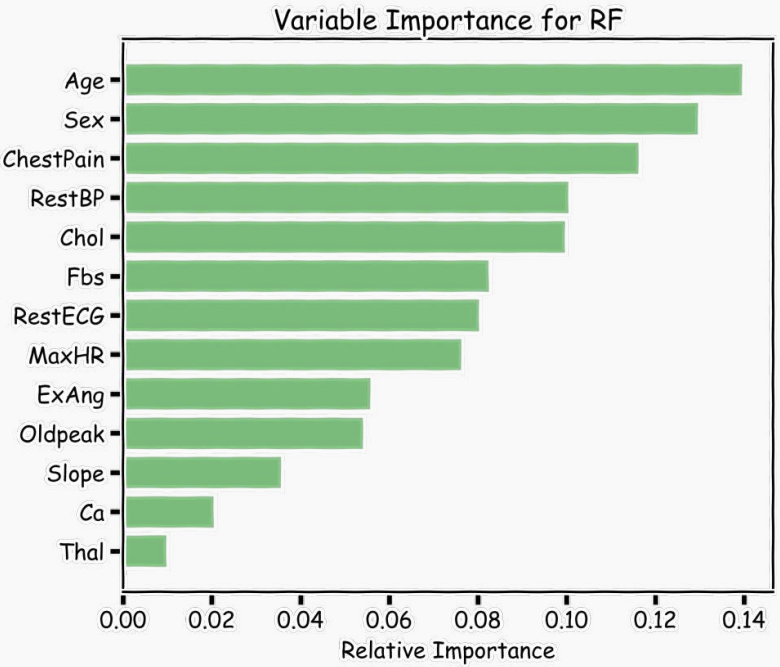
Variable Importance for RF

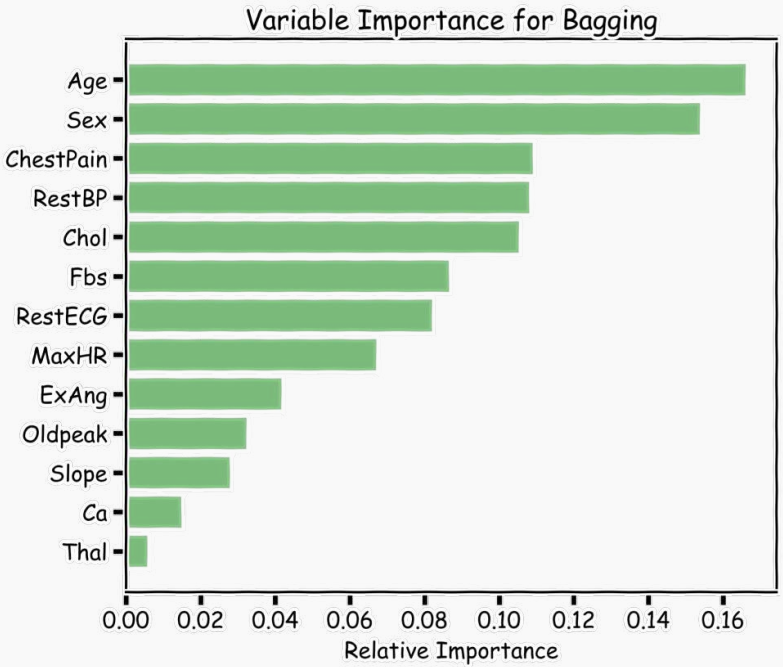
Explaining predictions from tree models is always desired; the patterns uncovered by a model are, in some applications, more important than the model's prediction performance.

A drawback of RF, Bagging, and other ***ensemble methods,*** is that the averaged model is no longer easily interpretable - i.e. one can no longer trace the *logic* of an output through a series of decisions based on predictor values!



Variable Importance for RF





100 trees, max_depth=10

Variable Importance for RF

1. Mean Decrease in Impurity (MDI)

- Same as Bagging.
- Calculate the total amount that the RSS (for regression) or Gini index (for classification) is decreased due to splits over a given predictor in different nodes (weighted by the probability of reaching that node (which is approximated by the proportion of samples reaching that node)).
- The weighted decrease in purity as a result of the splits over a given predictor 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 or Mean Decrease in Accuracy (MDA)

- 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.

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.