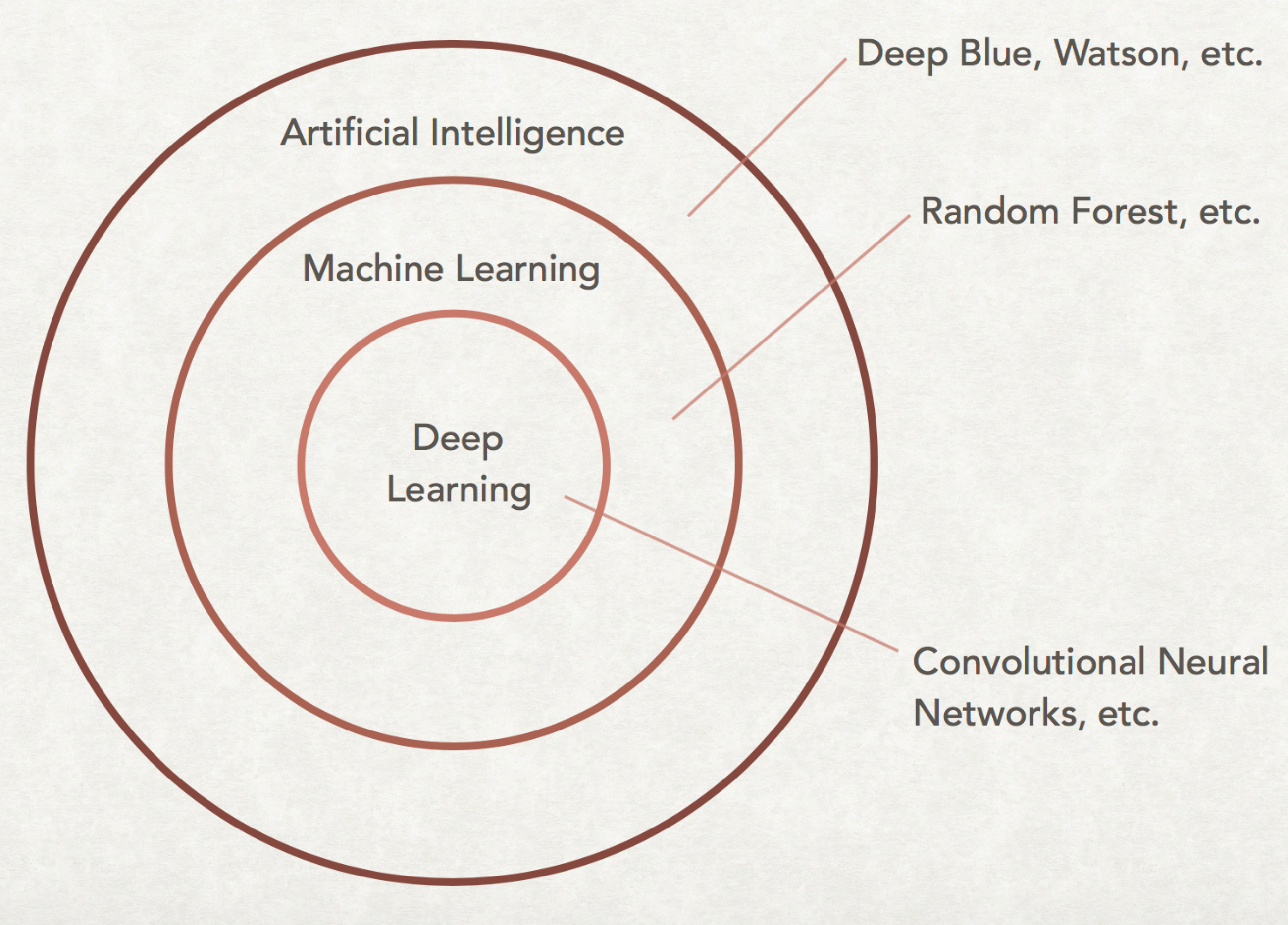
Machine Learning + Trees

36-600

What is Machine Learning?

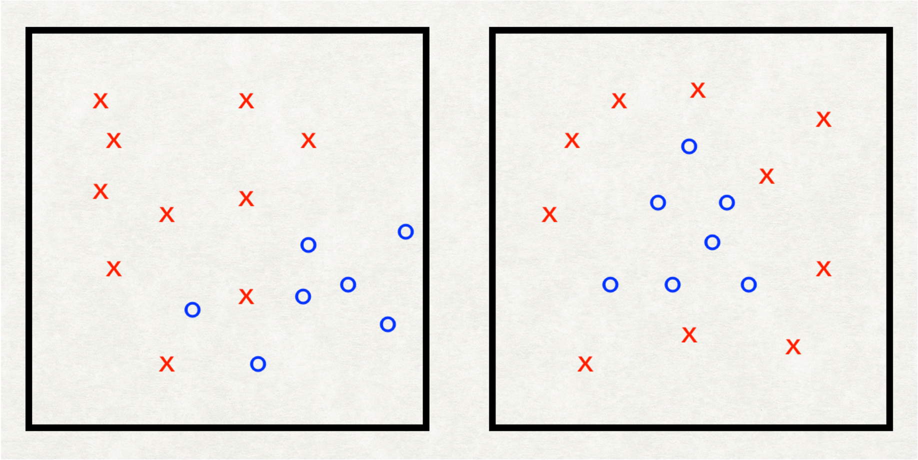
- The short version:
  - machine learning (ML) is a subset of statistical learning that focuses on prediction.
- The longer version:
  - ML is the idea of constructing data-driven algorithms that *learn* the mapping between predictor variables and response variable(s)
  - we suppose no parametric form for the mapping *a priori*, even if technically one can write one down *a posteriori* (for example, by translating a tree model to a indicator-variable-filled mathematical expression)
  - linear regression, for instance, is not a ML algorithm since we can write down the linear equation ahead of time, but random forest is a ML algorithm since we've no idea what the number of splits will end up being in each of its individual trees

Machine Learning: the Broader Context



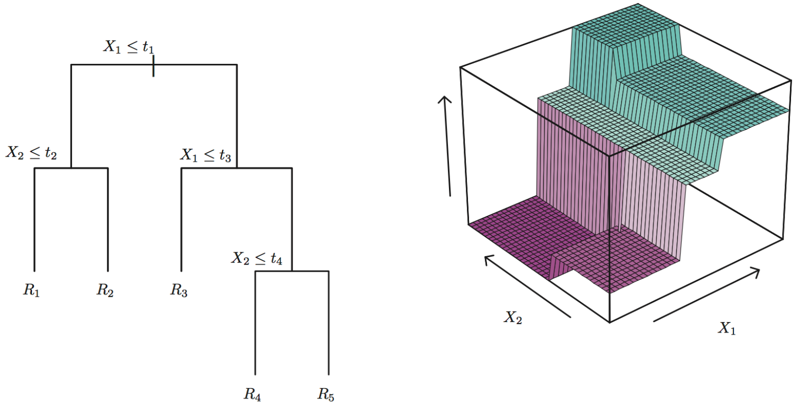
Machine Learning: Which Algorithm is Best?

- That's not actually the right question to ask
- The right question is ask is: why should I try different algorithms?
  - because you (generally) cannot visualize the distribution of predictor variables in their native space
  - the performance of different algorithms is predicated on how predictor data are distributed...
- The picture to the right shows data for which there are two predictor variables (along the x-axis and the y-axis) and for which the response variable is binary: x's and o's
  - an algorithm that utilizes linear boundaries or that segments the plane into rectangles will do well given the data to the left
  - an algorithm that utilizes circular boundaries will fare better given the data to the right
  - "do well/fare better": will do a better job at predicting whether a new datum is actually an x or an o



Your First ML Model: the Decision Tree

- A decision tree is a model that segments a predictor space into disjoint p-dimensional hyper-rectangles, where p is the number of predictor variables
  - for a regression tree, the predicted response in a hyper-rectangle is the average of the response values in that hyper-rectangle
  - for a classification tree, by default the predicted class in a hyper-rectangle is that class that is most represented in that hyper-rectangle
- Should I use this model?
  - yes: it is easy to explain to non-statisticians and easy to visualize/interpret
  - no: trees do not generalize as well as other models (i.e., they tend to have higher test-set MSEs)



(Figure 8.3, *Introduction to Statistical Learning* by James et al.)

Decision Tree: Detail

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

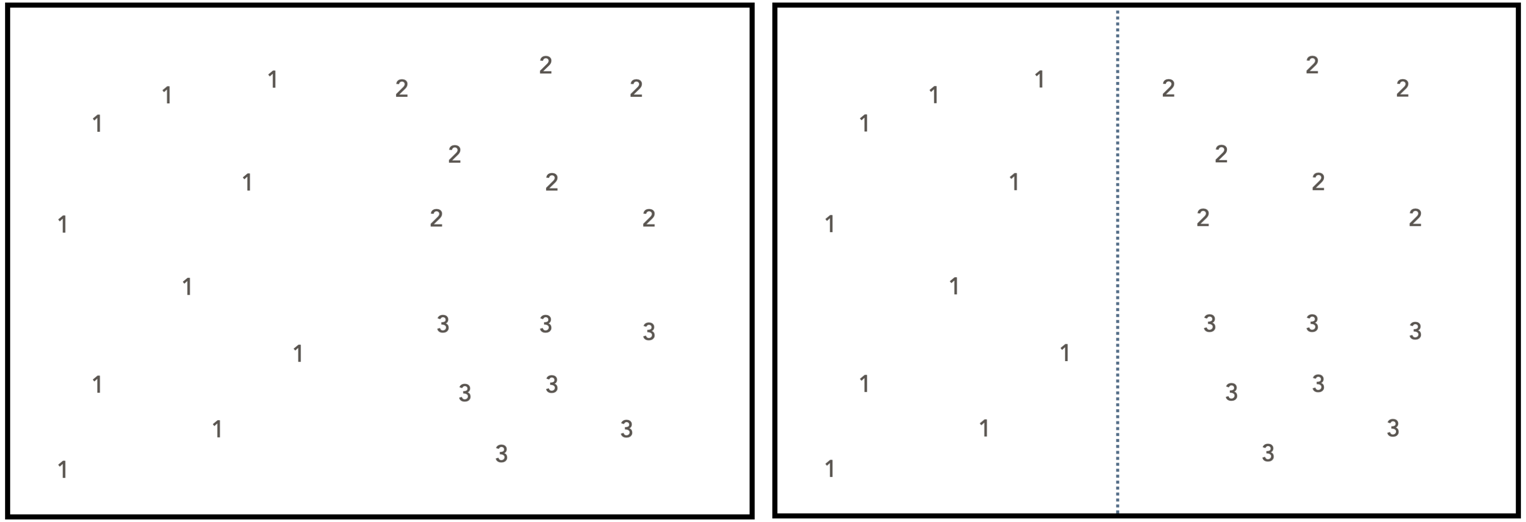
   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

(Algorithm 8.1, *Introduction to Statistical Learning* by James et al.)

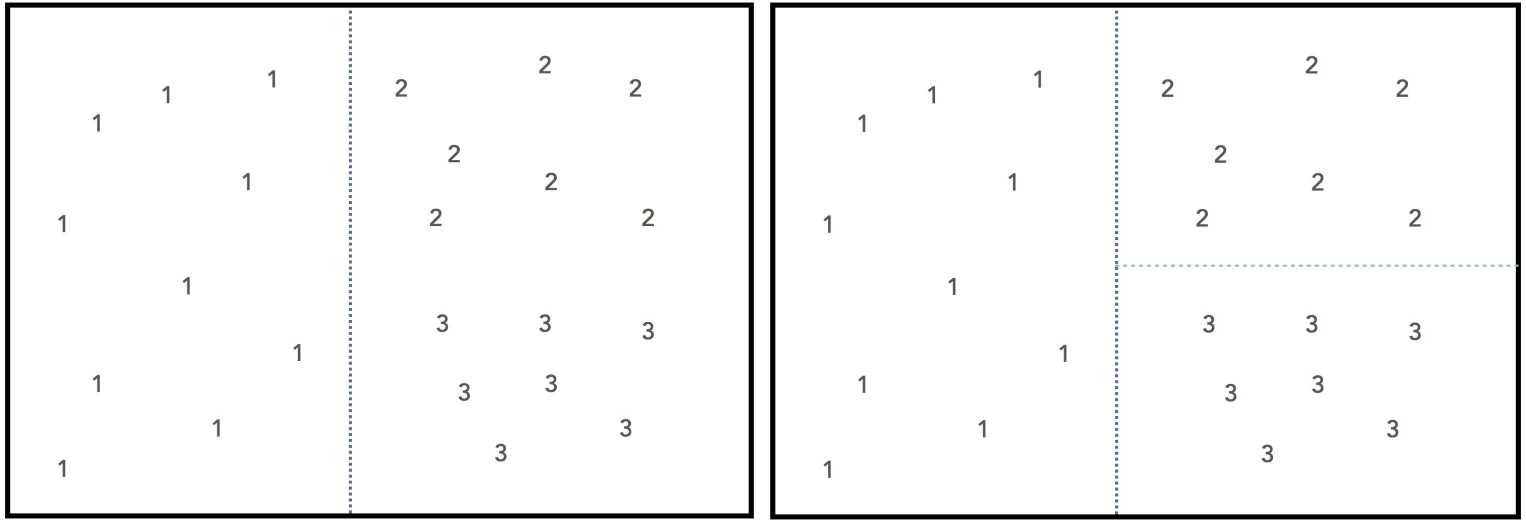
- The classification tree algorithm is similar: instead of splits based on reduction of the residual sum-of-squares (RSS), the splits would be based on, e.g., reduction of the Gini coefficient, a metric whose value becomes smaller as each node becomes more "pure," i.e., populated more and more by objects of a single class

Decision Tree: Regression Example



- The dataset above has two predictor variables and a quantitative response (1, 2, or 3)
    - pre-split: $\hat{Y} = \bar{Y} = 1.875$, with RSS = 16.625
    - post-split: $\hat{Y}_{\text{left}} = 1$ and $\hat{Y}_{\text{right}} = 2.5$ (total RSS = 3.5): the change in RSS is sufficient to accept the proposed split

Decision Tree: Regression Example



- The splitting process continues separately in each new segment
    - no further splits are possible to the left
    - the proposed split to the right reduces the overall RSS to zero
    - no further splits can be made at upper right or lower right: the tree-building is done

Decision Tree: Caveats

- The decision tree algorithm is a *greedy algorithm* which utilizes top-down *recursive binary splitting* to build the model, so the final model may not be "globally optimal" (i.e., it may not have the smallest possible overall RSS or Gini value)
- Overfitting is an issue: a tree that places a hyper-rectangle around each datum will be highly flexible (with training set MSE or MCR equal to zero!) but will not generalize well
    - a strategy to mitigate overfitting is to grow a large tree, then apply *cost complexity* (or *weakest link*) pruning

Decision Tree: Example

- Our dataset has 25 predictor variables and a response variable that is an object type: 385 galaxies and 243 stars
    - we do a 70-30 split of the dataset to create training/test sets

```
library(rpart)
rpart.out  <- rpart(class~.,data=df.train)
class.prob <- predict(rpart.out,newdata=df.test,type="prob")[,2]
class.pred <- ifelse(class.prob>0.5,"STAR","GALAXY")
round(mean(class.pred!=df.test$class),3)
```

```
## [1] 0.069
```

```
table(class.pred,df.test$class)
```

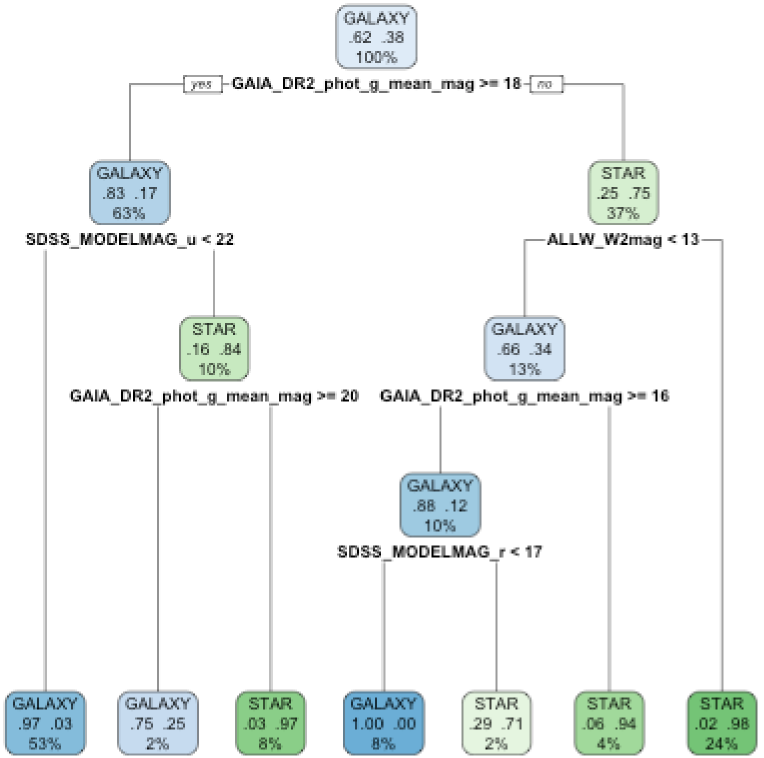
```
##
## class.pred GALAXY STAR
##     GALAXY    109    9
##     STAR        4   66
```

- A classification tree does much better than simply classifying all the data as galaxies!...(the "null MCR" is 0.387)

Decision Tree: Example

- Inference is done via examination of the tree
    - test-set prediction is done by sending each test-set datum "down" the displayed tree and seeing where it ends up

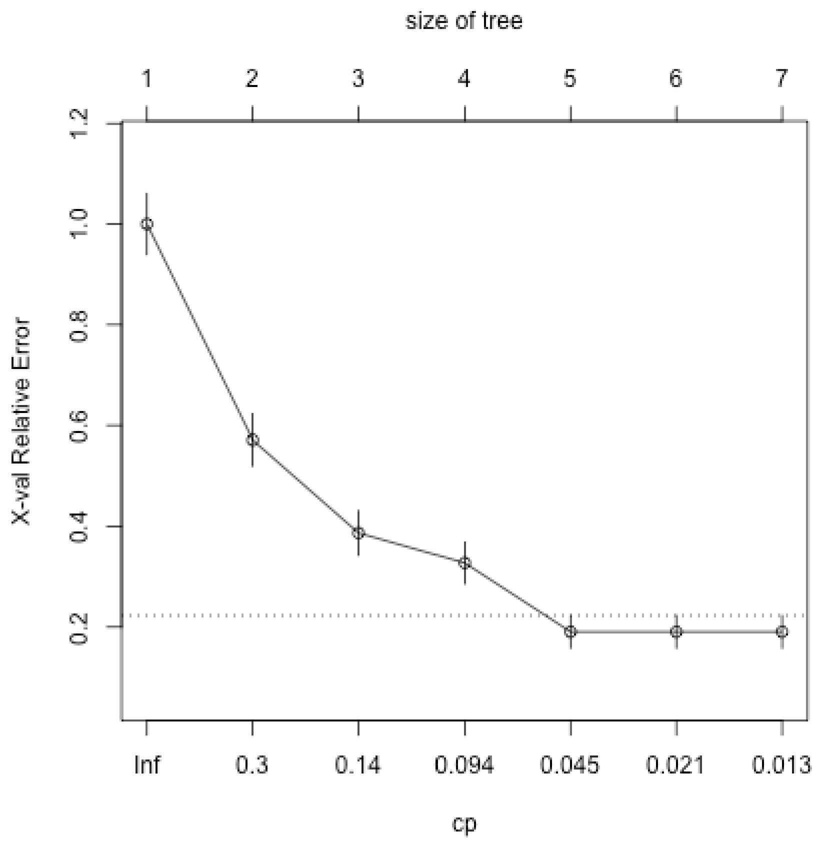
```
library(rpart.plot)
rpart.plot(rpart.out,extra=104)
# "extra": see the rpart.plot documentation
```



Decision Tree: Example

- From the `plotcp()` documentation: "[a] good choice of `cp` for pruning is often the leftmost value for which the mean lies below the horizontal line"
    - here, that would be 0.045, which corresponds to 5 leaves

```
plotcp(rpart.out)
```



Decision Tree: Example

```
rpart.pruned <- prune(rpart.out,cp=0.045)
class.prob   <- predict(rpart.pruned,newdata=df.test,type="prob")[,2]
class.pred   <- ifelse(class.prob>0.5,"STAR","GALAXY")
round(mean(class.pred!=df.test$class),3)
```

```
## [1] 0.069
```

```
table(class.pred,df.test$class)
```

```
##
## class.pred GALAXY STAR
##     GALAXY    108    8
##     STAR        5   67
```

- The pruned-tree MCR is the same than the unpruned-tree MCR, so in this case pruning did not adversely impact the model's generalizability

Decision Tree: Example

```
rpart.plot(rpart.pruned,extra=104)
```