Machine Learning + Trees

36-600

Fall 2021

What is Machine Learning?

The short version:

- Machine learning (ML) is a subset of statistical learning that focuses on prediction.

The longer version:

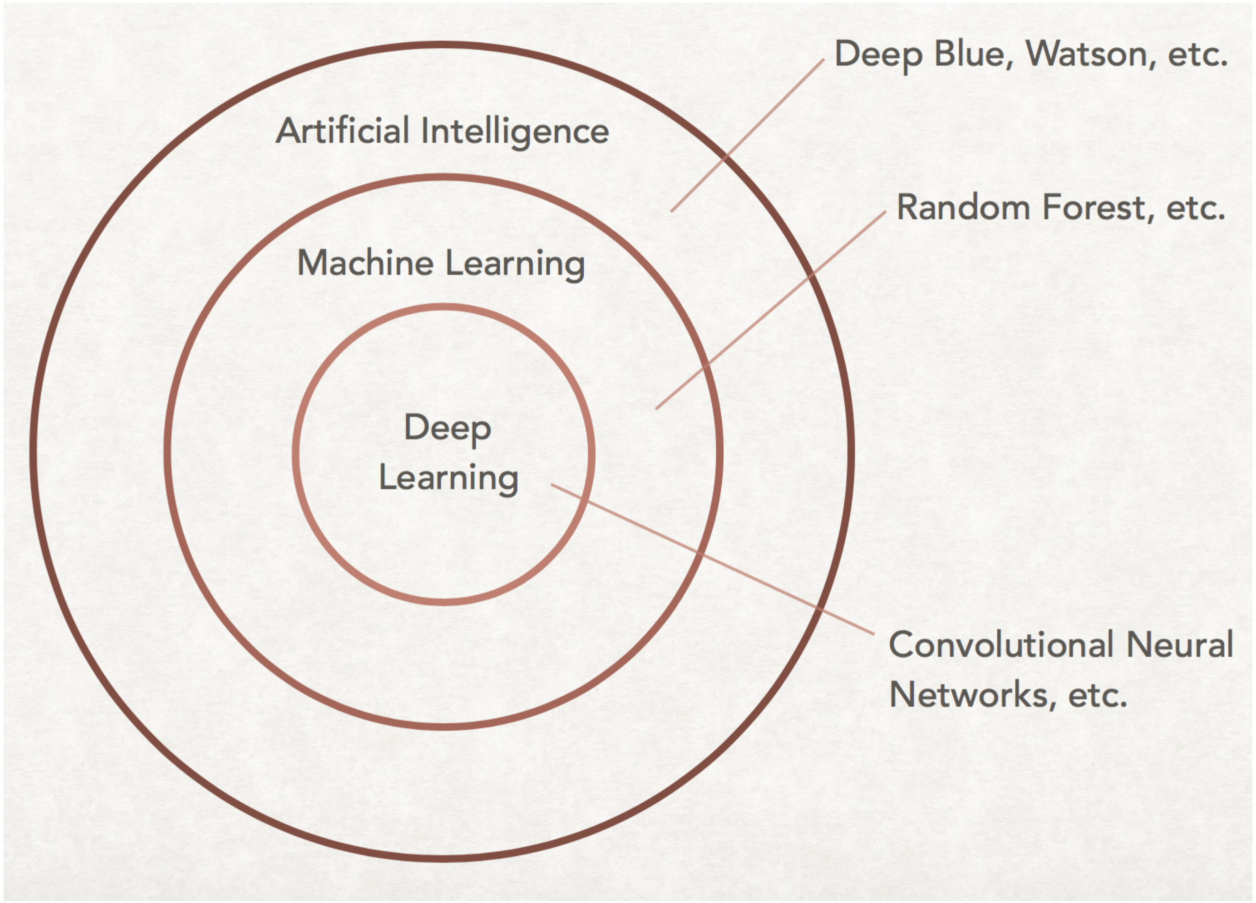
- ML is the idea of constructing data-driven algorithms that *learn* the mapping between predictor variables and response variable(s). Specifically, we suppose no parametric form for the mapping *a priori*, even if technically one can write one down *a posteriori* (for example, by translating a tree 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.

However, note that despite the statement that ML "focuses on prediction," there are those (e.g., Cynthia Rudin) who are leading an effort towards developing "interpretable ML," for instance through the construction of globally optimal logical models (models akin to trees). (See, e.g., the `corels` package and its documentation.)

- Note: do not confuse "interpretable ML" (which focuses on constructing interpretable models) with "explainable ML" (which are post-hoc attempts at explaining the results of black-box algorithms).

Machine Learning: the Broader Context



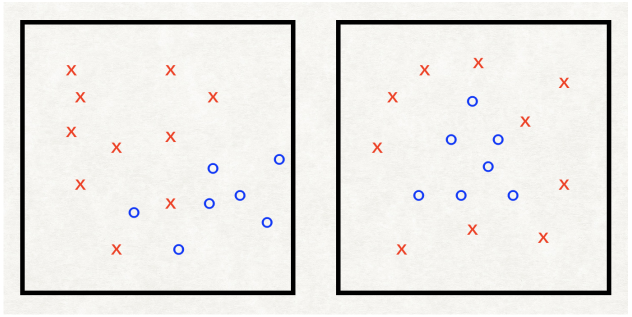
Machine Learning: Which Algorithm is Best?

That's not actually the right question to ask.

(And the answer is *not* deep learning. Because if the underlying relationship between your predictors and your response is truly linear, *you do not need to apply deep learning*! Just do linear regression. Really. It's OK.)

The right question is ask is: why should I try different algorithms?

The answer to that is that without superhuman powers, you cannot visualize the distribution of predictor variables in their native space. (Of course, you can visualize these data *in projection*, for instance when we perform exploratory data analysis.) And the performance of different algorithms will be predicated on how predictor data are distributed...



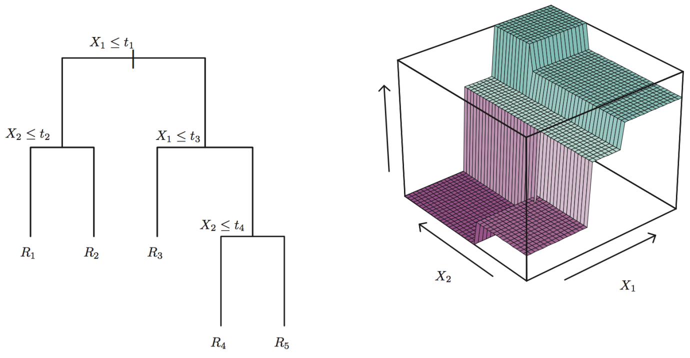
The picture above shows data for which there are two predictor variables (along the x-axis and the y-axis) and for which the response variable is binary: x's and o's. An algorithm that utilizes linear boundaries or that segments the plane into rectangles will do well given the data to the left, whereas 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.

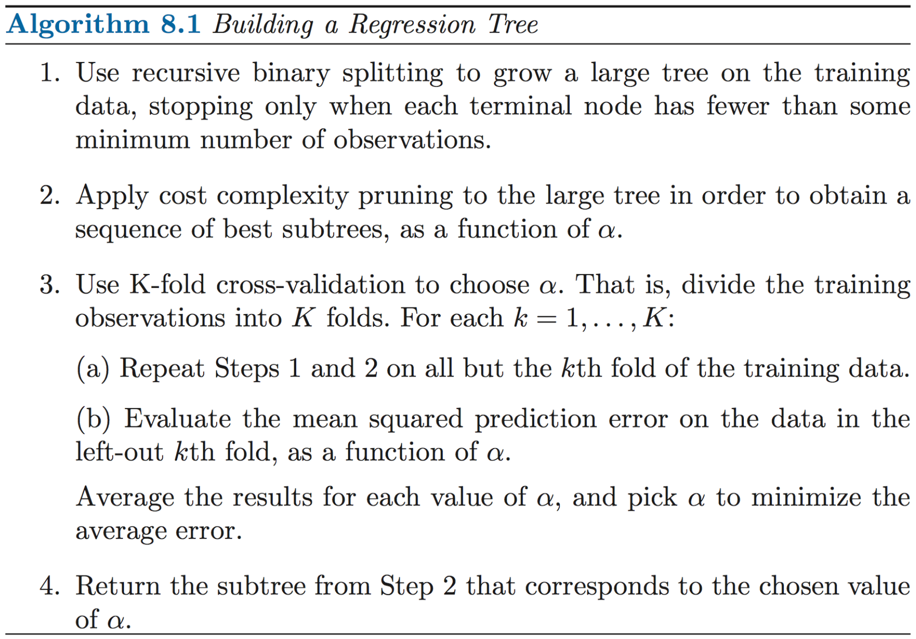


(Figure 8.3, *Introduction to Statistical Learning* by James et al.)

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).

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.)

While the algorithm given above is for a regression tree, 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, which is a metric that becomes smaller as each node becomes more "pure," i.e., populated more and more by objects of a single class.

Decision Tree: Caveats

- In a perfect world, one would systematically try all combinations of hyper-rectangles to see which combination minimizes values of RSS/Gini. However, our world is imperfect; the decision tree algorithm is a *greedy algorithm* which utilizes top-down *recursive binary splitting* to build the model: while each split is "locally optimal" (i.e., it causes the largest reduction in RSS or Gini), the final model may not be "globally optimal" (i.e., it may not have the smallest possible overall RSS or Gini value).
- When building a decision tree, one must guard against overfitting. For instance, 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. One strategy for dealing with overfitting is to grow a large tree, then apply *cost complexity* (or *weakest link*) pruning (as described in Steps 2-4 above).

Decision Tree: Example

Here we will look at an example of a dataset with 25 predictor variables and whose response variable is object type: 385 galaxies and 243 stars. We do a 70-30 split of the dataset: 70% for training and 30% for testing.

```r
library(rpart)
rpart.out = rpart(class~.,data=df.train)
class.prob = predict(rpart.out,newdata=df.test,type="prob")[,1]
class.pred = ifelse(class.prob>(385/(385+243)),"GALAXY  ","STAR    ")
round(mean(class.pred!=df.test$class),3)
```

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

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

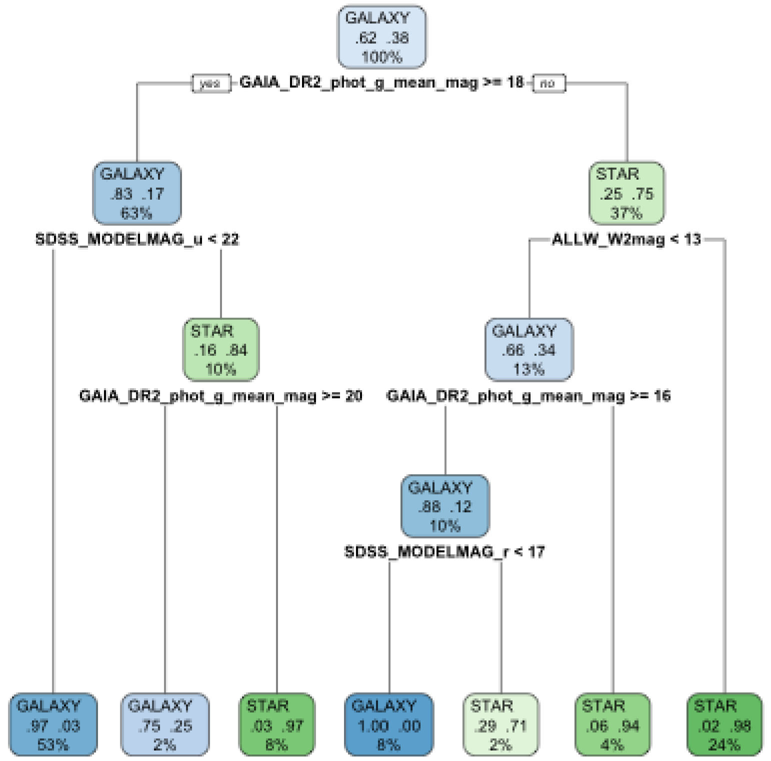
```
##
## class.pred GALAXY   STAR
##   GALAXY        109      9
##   STAR            4     66
```

We see that a classification tree does much better than simply classifying all the data as galaxies (null MCR: 0.387).

Decision Tree: Example

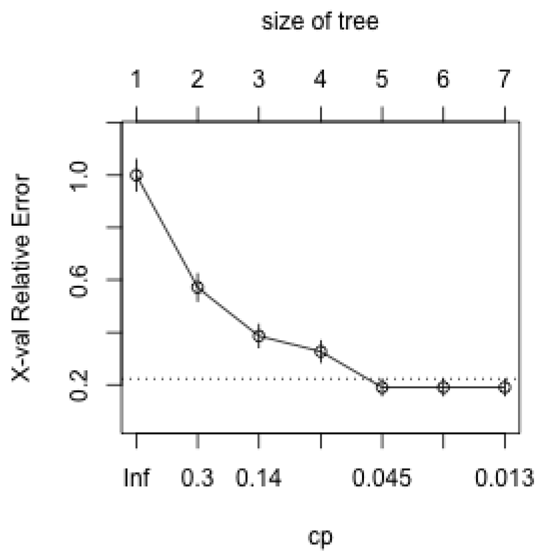
Inference is done via examination of the tree.

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



Decision Tree: Example

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



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.

Decision Tree: Example

```
rpart.pruned <- prune(rpart.out,cp=0.045)
class.prob <- predict(rpart.pruned,newdata=df.test,type="prob")[,1]
class.pred <- ifelse(class.prob>(385/(385+243)),"GALAXY  ","STAR    ")
round(mean(class.pred!=df.test$class),3)
```

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

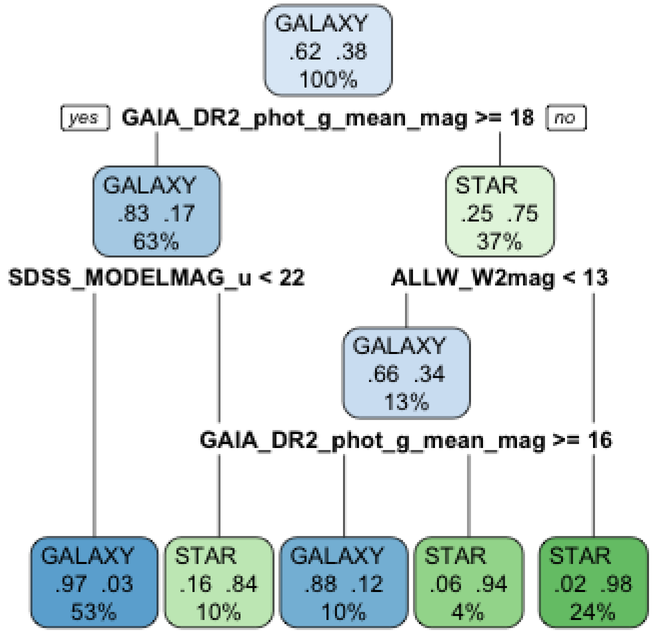
```
table(class.pred,df.test$class)
```

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

We note that 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. (Good!) In general, explore pruning the tree to different extents if it is clear that pruning is a viable option (multiple points below the red line), but if it (markedly) increases the test-set MCR, ignore the pruning option.

Decision Tree: Example

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


GALAXY .62 .38 100%
yes
GAIA_DR2_phot_g_mean_mag >= 18
no
GALAXY .83 .17 63%
STAR .25 .75 37%
SDSS_MODELMAG_u < 22
ALLW_W2mag < 13
GALAXY .66 .34 13%
GAIA_DR2_phot_g_mean_mag >= 16
GALAXY .97 .03 53%
STAR .16 .84 10%
GALAXY .88 .12 10%
STAR .06 .94 4%
STAR .02 .98 24%