The Art of Statistics, Chapter 6: Algorithms, analytics, and prediction
Take-Away Notes

This chapter gives us the lay of the land when it comes to using algorithms to do statistical modeling.
- Supervised learning = using an algorithm to make predictions when you know in advance what you're looking for. To do it properly, we need to *strictly* train the algorithm on a subset of the data (the **training set**, and test it on another (the **test set).**
- Unsupervised learning = using an algorithm to find patterns in data, when you don't know in advance what you're looking for.
- Sometimes, dimensionality reduction (or reduction in the number of columns) is a necessary prerequisite to using a classification algorithm; this is known as **feature engineering.**

A **binary classification tree** asks a sequence of yes/no questions, examining features in sequence until a classification is made. Such trees are evaluated with **error matrices** which represent accuracy (% correctly classified) sensitivity (% of those with the feature of interest correctly classified) and specificity (% of those without that feature correctly classified) done on both the training and test set.

- Algorithms that output a probability (or a number) rather than a simple yes/no binary classification are often compared using **Receiver Operating Characteristic (or ROC) curves** , which pick a threshold value above which a data-point is classified as a 'yes' and below which as a 'no'. Different choices of value here change the sensitivity and specificity values of the algorithm.

An algorithm also needs **calibration**: a check that the observed frequencies of events match those expected by the probabilistic predictions of the outcome. *E.g*: if some event is given the probability of 0.85, we need to check that that event actually occurs roughly 85% of the time.

The **mean-squared error** measures the performance of our algorithm, and is the average of the squares of the errors; the average mean-squared error is known as the **Brier score**.

Excessively adapting a classification tree to the training data makes its predictive power decline, and is known as **over-fitting**. We overfit when we take into account all the available information (reducing **bias**) but thereby increasing **variation** in our estimates. Since too much bias is bad, we need to make a judgment on the **bias/variance trade-off**. We can use repetitions of **cross-validation** (removing a percentage of the training data, developing the algorithm on the remaining data, and then testing it on the removed data) to help counter this. We typically build a classification tree by deliberately overfitting, and then pruning the tree back to something simpler and more robust.