DECISION TREES

You're right that they can tend to suffer from over-fitting unless we're careful about tuning them. As you probably remember from the model validation project,the max\_depthparameter can be a useful starting point for reducing overfitting.  
Another popular method is pruning:  
<https://en.wikipedia.org/wiki/Pruning_(decision_trees>)

Although pruning is not strictly supported in sklearn, tuning the min\_impurity\_splithyper-parameter serves is a great replacement.  
<http://blog.nelsonliu.me/2016/08/05/gsoc-week-10-scikit-learn-pr-6954-adding-pre-pruning-to-decisiontrees/>

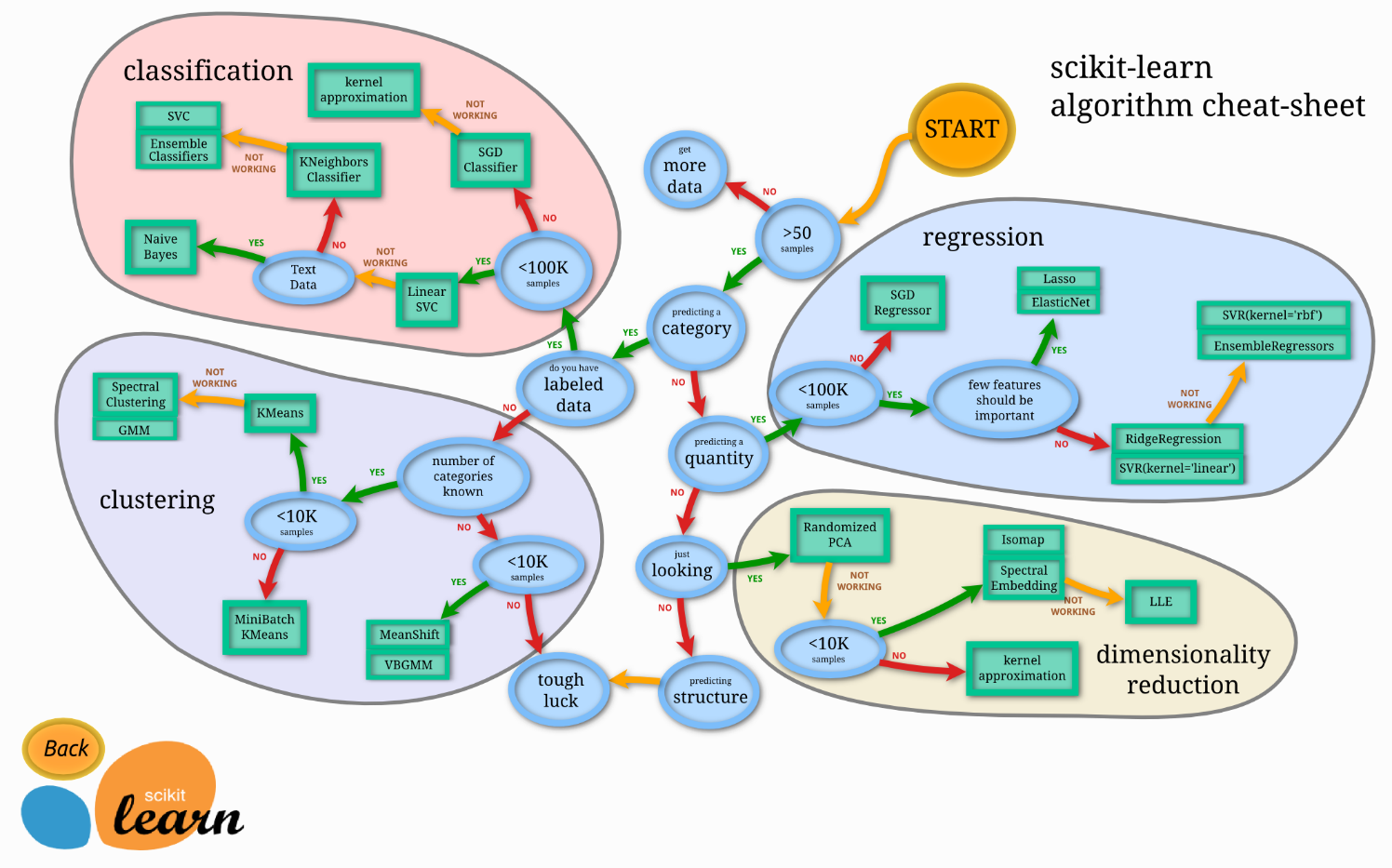
LOGISTIC REGRESSION

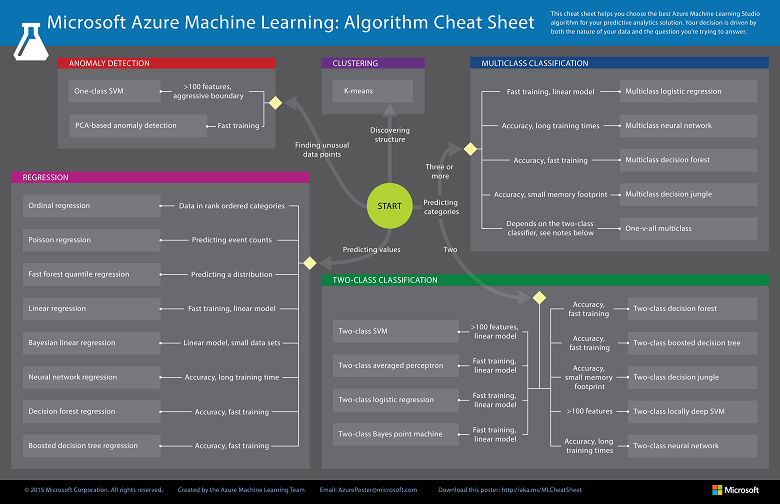
You might find that it's difficult to get extra performance out of this model in the hyperparameter tuning phase. This is a consequence of the high-bias nature of the model. One workaround we can use is to perform some clever *feature engineering*. By creating new features that are non-linear combinations of existing features, we can allow logistic regression to learn more complex relationships in the data.  
You can use the [PolynomialFeatures](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html" \t "_blank) class in sklearn to do the legwork here.

GAUSSIAN NAIVE BAYES

It turns out that this model can actually work optimally even when the independence assumption doesn't hold. The caveat is that in these cases, the feature dependence must be distributed evenly across features. In practice this is just as rare as pure independence, but knowing about it does provide another lens through which to examine whether this model is appropriate for a given problem.  
If you'd like to look at the proof of this, check out the paper below. (Warning - it's *very* heavy on the math).  
<http://www.aaai.org/Papers/FLAIRS/2004/Flairs04-097.pdf>

MODEL SELECTION

Even if we carefully analyze our data, we'll likely still have some level of trial and error involved in our model selection process. Unfortunately, there's no one-size-fits-all solution either. However, there are a few great heuristics we can use to really help narrow down the process! Check out the cheat sheets posted below.  
**Sklearn**  
[](https://udacity-reviews-uploads.s3.us-west-2.amazonaws.com/_attachments/32786/1520528637/ml_map.png)

**Azure**  
[](https://udacity-reviews-uploads.s3.us-west-2.amazonaws.com/_attachments/32786/1520528646/machine-learning-algorithm-cheat-sheet-small_v_0_6-01.png)