Cross Validation - not for Economists, but it will be.

Part I - Model Performance

Estimating model performance w/ cross validation [see scikitlearn for code]

Suppose you have a model and you think it's good. Economists would typically say that theory drives the model and so the model is right if the theory is right.

Say the model is:

```
wages ~ education + age + age^2
```

You want to know how good of a model is (this model is also known as the Mincer equation). Namely, a model explaining wages with education and age polynomial two.

Here economists would estimate the model on the data that they have, look at the Rsquared, see if the coefficients are significant. High Rsquared and significant (and unbiased) coefficients? Great!

Machine Learning (ML) specialists would say - wait a second - you're taking your estimate of fit (R2, MSE, RSE) from the data you estimated it on? That's not indicative of anything. Sure, if I applied this fitted model and its beta estimates to a nearly identical dataset then my R2 and my RSE = Sum (y-yhat)^2/N-1 would be great (high R2, low RSE and MSE). But ML-ers want to know how well the model does in prediction data out of sample - i.e. in predicting the y outcomes for another sample. For example, we estimate a model for purchasing goods on amazon today. How well does it do in predicting what goods are purchased tomorrow?

What is prediction?

Prediction is simple.

Suppose we want to estimate a linear model using OLS

$$y = b + a * X + \epsilon$$

We obtain point estimates of $\widehat{a}$ and $\widehat{b}$. Now we would like make a prediction about how well this linear model would predict the outcome variable given another sample. Let's say that new sample data are y_new, X_new.

We can compute $\widehat{y\_new} = \widehat{a} + \widehat{b} * X\_new$ and compare these values to the true y_new. To measure our out-of-sample prediction accuracy we compute:

MSE =

$$\frac{1}{N}\sum_{i=1}^{N}(y\_new_i - \widehat{y\_new_i})^2$$

Here is where cross validation comes in. To test our model's predictive abilities we want to know how well the model does in predicting y_new on average - not just for one single case as we've described thus far.

Test and train, test and train, test and train ---> kfold CV

We'll now split our data from the ***outset*** into folds (partitions), say, in the simplest case, 2 partitions - training and testing data. This is justwhat was described above using X, y and testing on X_new, y_new, but we intentionally divide our data up apriori.

We now estimate the model on the training data, leaving out the other partition to test on. We predict out-of-sample using the left out (test) data (i.e. multiply the estimated coefficients times the new data X_test in a Linear case), and compute a score (R2, MSE, RSS). If the MSE is jumping all over the place or is high, that's no good. We'd want it to be consistent and consistently low.

We can generalize this procedure to more than 2 folds - say 5, or 10, or just k. This procedure is called kfold cross validation (cv), depending on the number of k paritions. In kfold cv, we have k partitions. We estimate the model on k-1 partitions, and test it on the kth partition.

Economists don't always check for out-of sample prediction - macro economists do so more than micro economists, though macro time series can be quite short if they're only reported annualy. Applied microeconomists tend to focus on causal estimates of particular parameters given the data. Structual econometricians do look at how well their model fits the data - but rarely in the iterative approach of cross validation (CV). Part of this discrepancy in approach can be attributed to the fact that applied microeconomists use policy to change outcomes; They care about being able to effect change through a particular policy variable (often a program) given today's data. But the other part of this is data frequency. ML generally requires long time series that are updated frequently, else, we won't improve much on out-of-sample prediction (high and variable MSEs). Exceptions include finance where there's high frequency price data, or perhaps store data.

Bayesian statisticians do do out-of-sample predictions (though, that paradigm is a bit different from the ML paradigm. Bayesians will repeatedly sample from a posterior distribution of their parameters and then simulate data from those distributions. They'll compare their averaged simulations to true historical data.)

Part II - Model Selection

Tuning your hyperparameters using cross validation. see skitlearn

Perhaps Part I didn't go well. You're out of sample predictions are poor: High MSE, Low R2. You're wondering if you specified the "right" model.

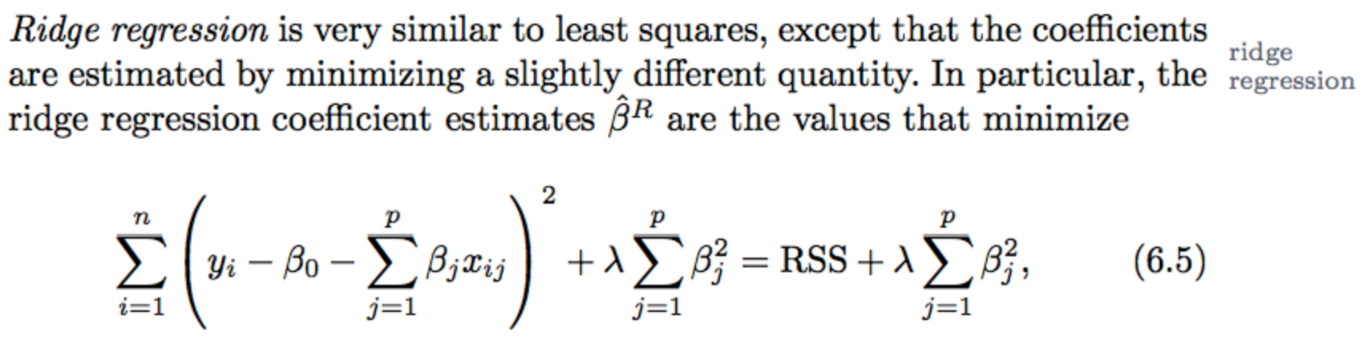
What do we mean by right model? For economists, the right model is driven by theory. An economist would re-write their theoretical model and determine whether the equation they're estimating changes. For example, perhaps an interaction effect between age and education falls out of their rewritten model:

```
    wages ~ education + age + age^2 + education*age
```

For ML - the right model means selecting the best values for the hyperparameters. What are hyperparameters? They are parameters that are not estimated (i.e. NOT the coefficients (aka weights) in the model) but rather, chosen by the researcher that put constraints on the model. Hyperparameters could include: the polynomial degree of a variable (i.e. how many polynomial degreess should there be for age?), the shrinking factor of all the variables (lambda in ridge or lasso), or the number of trees.

One question you might have here is if this tuning procedure will actually choose how many covariates(aka features) to include in the model? Answer: Not directly - ML WON'T tell you exactly which covariates to use. But what ML can do is downselect (shrink) from a large number of covariates. To accomplish this we'll add on a penalty term to our regression. This penalty term dampens the effect of covariates that aren't contributing much towards prediction (or are highly colinear with other variables). These models are called regularizaiton models (Ridge and Lasso, Elastic Net fall into this class of models.). Ridge is similar to OLS but adds on a penalty term to minimizing the sum of squared errors, lambda, to dampen the effects of covariates that contribute nothing (are colinear). Lambda is a hyperparameter.

(From ISLR textbook on ML:)



Ridge regression is very similar to least squares, except that the coefficients are estimated by minimizing a slightly different quantity. In particular, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

ridge regression

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \text{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2, \qquad (6.5)$$

If we are estimating lasso or ridge to penalize covariates, then we are implicitly choosing which variables to keep or which to quiet. Economists tend not to use lasso/ridge in model selection because 1) they believe that economic theory should dicatate the model we estimate, not the data itself and 2) the coefficient estimates from lasso/ridge do not follow a distribution and, therefore, we cannot perform inference on them (there are ways around this called bootstrapping). There is, however, an economist, Andrii Babij! no less, who is making headway on finding distributions for beta estimators from regularization. It's some really heady math.

Bias Variance Tradeoff

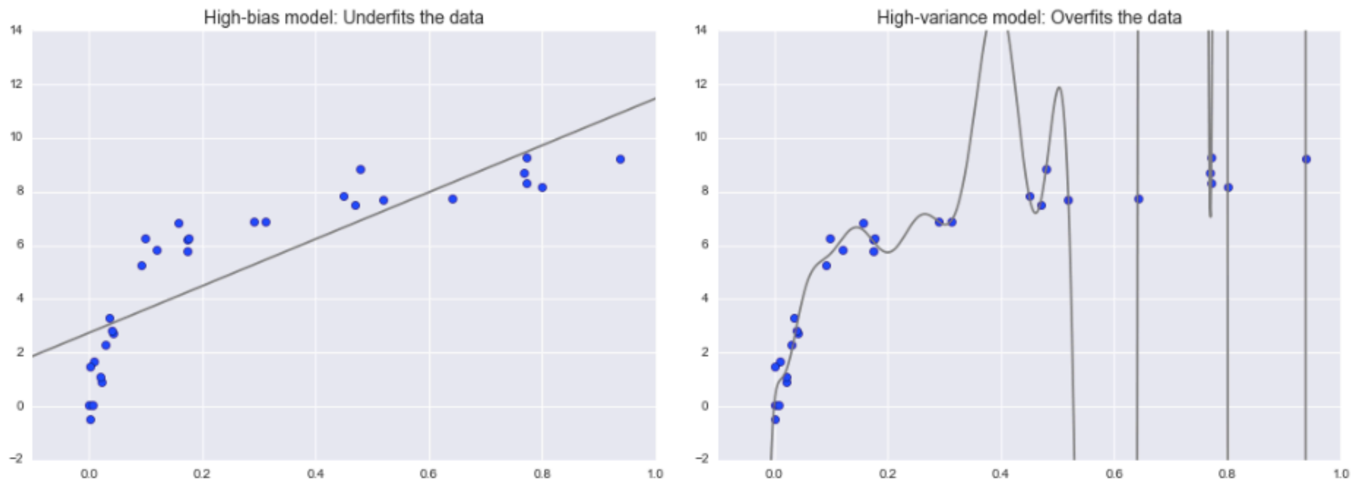
In tuning your hyperparamters comes a concept called bias-variance tradeoff. Our Vanderplas textbook (Ch 5 Hyperparameters and Model Validation) discusses this. Predicted values can be decomposed into the bias (how far away are the predicted y's from the true y?) and the variance (how much do our predicted y's vary?). We can't reduce bias without increasing variance, and vice versa.

$$E(y - \widehat{f(x)})^2 = Var(f(\hat{x})) + [Bias(f(\hat{x}))]^2 + Var(\epsilon)$$

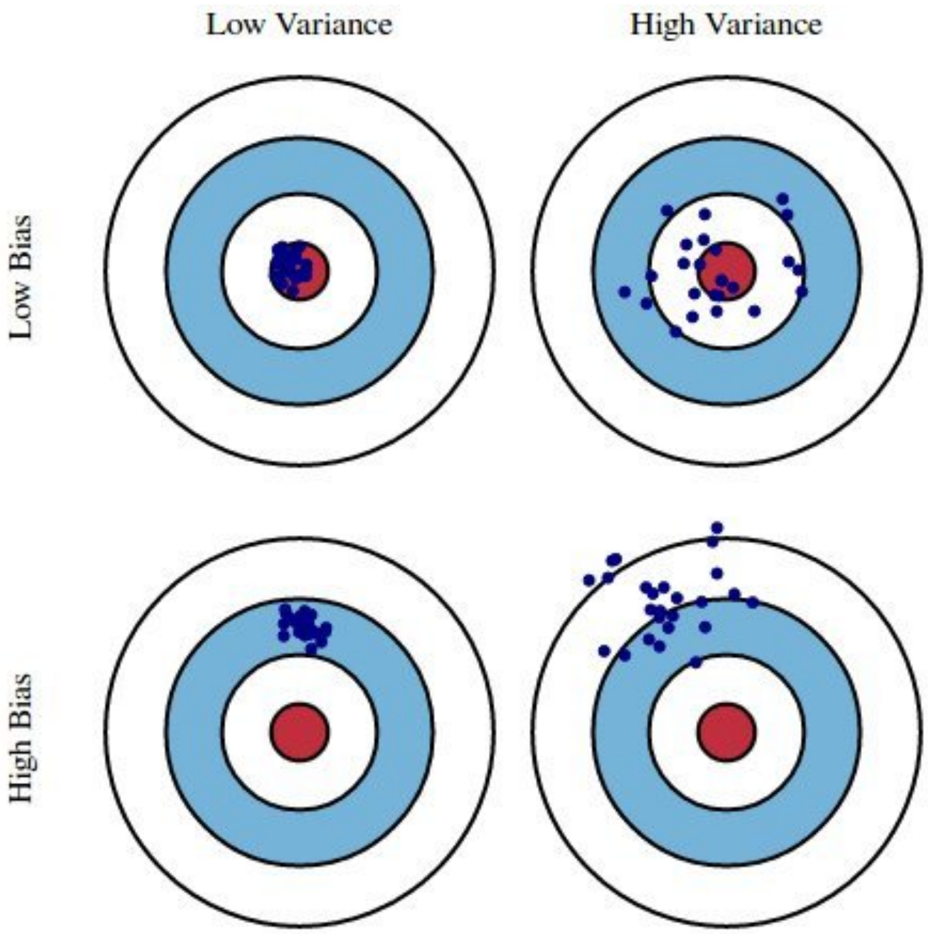
(where $f(\hat{x}) = \hat{y}$)

If we overfit the model to the data, we'll have low bias (perfect fit) but high variance (predicted y's are jumping all over the place). An overfitted model won't perform well out-of-sample, because it caters to one dataset, and an underfitted model won't peform well because it doesn't capture the data at all.

Here the graph on the left tries to predict data that clearly follow a polynomial shape with a line. A linear function is an underfit of these data - we'll have low variance but high bias. Conversely, the graph on the right fits a high degree polynomial to the data. The variance of $\hat{y}$ is high, but the bias of the model is low.



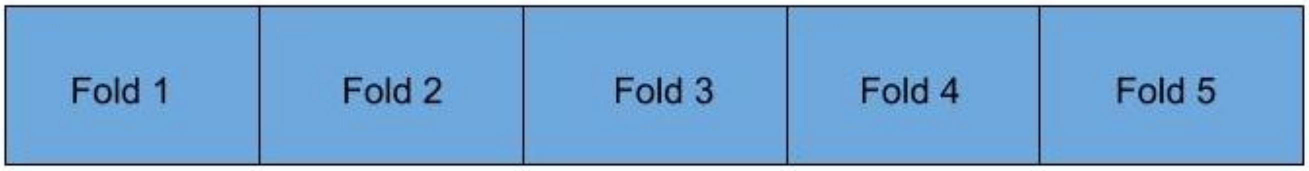
Here's another nice figure that shows the bias/variance tradeoff. The concentric circles describe the true underlying data generating process. In the NW corner the model use to predict with exhibits low variance and low bias for these test data. All the predicted data points fall tightly within the target bull's eye. In the SW corner the model exhibits high bias but low variance - the $\widehat{y}$'s are tightly clustered, but don't match the target. In the NE corner the $\widehat{y}$'s have low bias - i.e. most of the predicted points fall loosely around the target ,- but high variance. It is the case that we may prefer a model that has higher bias but lower variance (e.g. ridge or L2 norms will always have higher bias that OLS, but lower variance).



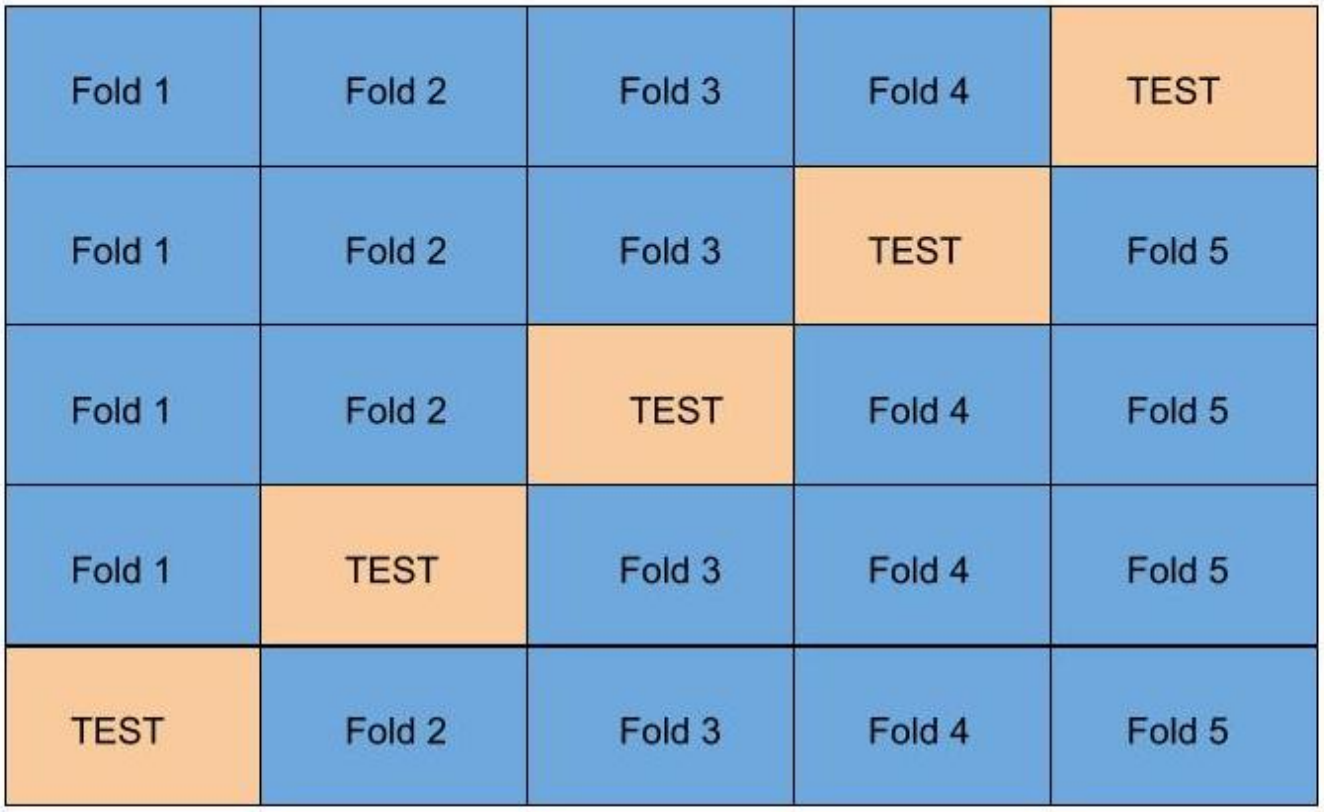
In ML, to get a good model fit we need to choose our hyperparameters wisely. How do we this??.....**kfold cross validation**!!!! with **grid search**. What does this mean?! Grid search of the hyperparameters means we try out many possible values for the hyperparameter. kfold cross validation means for each hyperparameter we train the model on the training data and then test it on a validation fold. So if we try out 10 values for the hyperparameter (which could be the polynomial degree of a variable: 1, 2, 3, 4 .... polynomials) and perform 5 fold cross validation for each paramater, that means we're estimating the model 50 times. For each hyperparmater we'll obtain a score (R2, MSE or RSE as we like). The hyperparamter with the best score wins!

We then have one last estimation to perform. The fifty-first estimation will be to fit the model using the best parameter from hypertuning exercise. We'll do this final fit on the training data. The we'll perform one last out-of-sample score calculation using the left out test data.

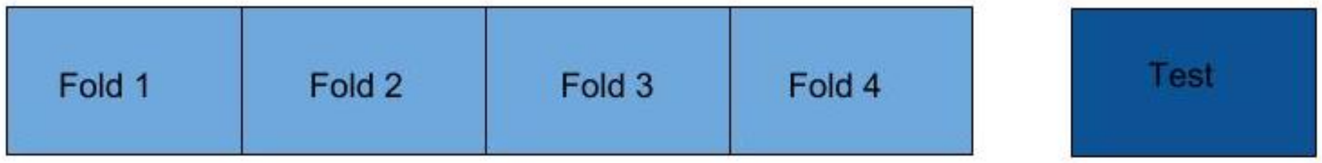
Note the main differences in how data are partitioned between **Part I** and **Part II**. In Part I, we partition the data into five folds:

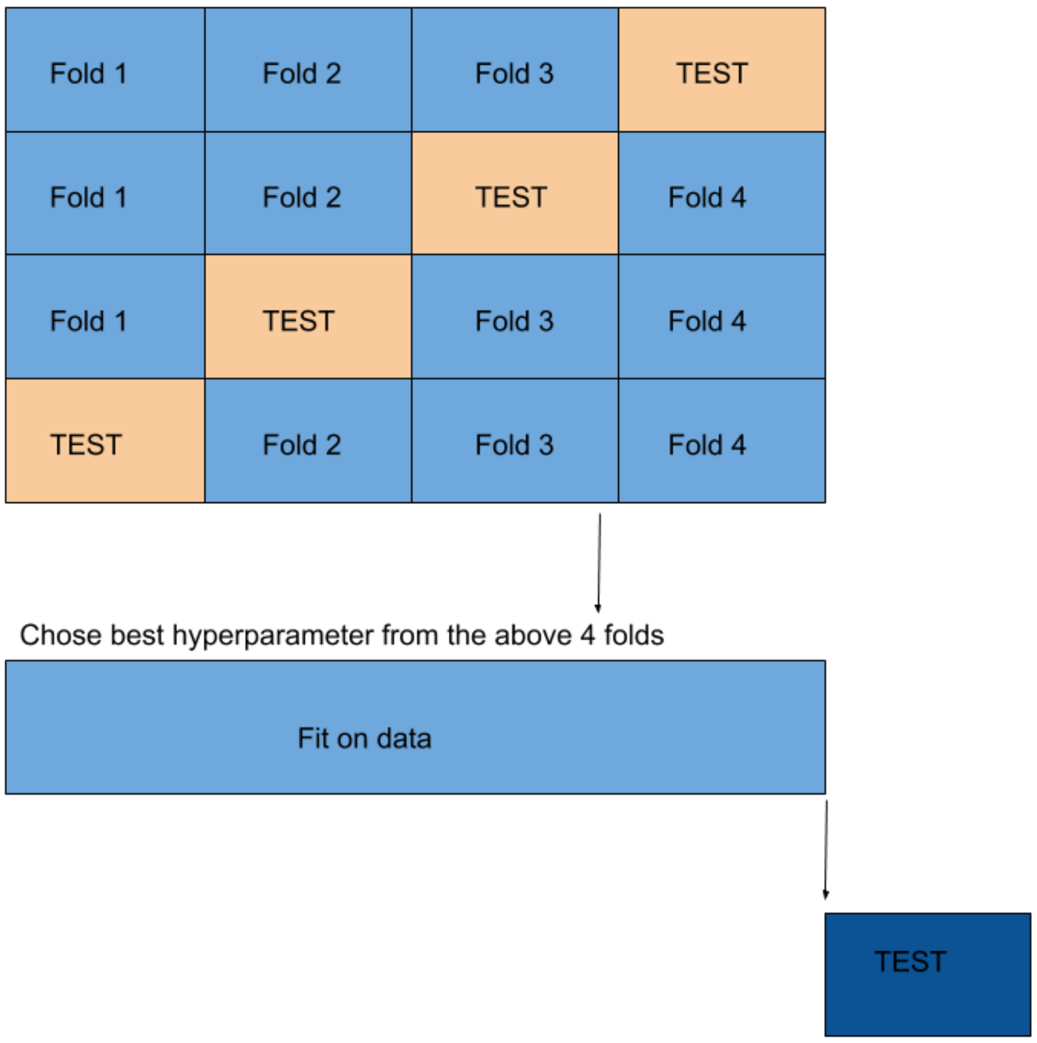


Where each fold will be potential a test set while the n-1 folds are used to train:



In Part II, we partition the data into train/validation data (on which we do kfold CV, here k = 4) and one final test data set.





There are, in fact, many more complex variations on this theme of kfold cross validation. For example, in the grid search we could pick a random values for our hyperparameter. We could also do kfold within each fold for hypertuning! Very meta.

Example

Suppose you want to test the model for a lambda of .1, 1 and, 1.1. Here are the steps to take:

a. You estimate the model with lambda = .1 on k - 1 folds -- > get the test error using the validation set --> repeat k times, average the test error rate say it's a_hat

b. You estimate the model with lambda = 1 on k - 1 folds -- > get the test error using the validation set --> repeat k times, average the test error rate say it's b_hat

c. You estimate the model with lambda = 1.1 on k - 1 folds -- > get the test error using the validation set --> repeat k times, average the test error rate say it's c_hat

You pick the lambda with the lowest error rate from validation. Say ahat was lower than bhat and chat. So you pick lambda = 0.1

Combine training and validation into one dataset - estimate the model with the best hyperparameter, and test it on the held out test data.

- **Repeat.**

(For the complete overview of model selection and model evaulation see SkikitLearn. For more information on machine learning, ISLR is a great book and what is used for Cal Poly's Statistical Learning course (Stat 434)).

- **More Reading.**

We have talked about ML not being a main paradigm for data analysis in Economics, yet. However, that is definitely changing. These two articles by very well known economists Hal Varian (Google and UC Berkeley) and Susan Athey (Microsoft and Standford) are excellent summaries about how ML is changing economics.