Implementing machine learning methods in Stata

Austin Nichols

6 September 2018

Definitions

What are machine learning algorithms (MLA)?

- Methods to derive a rule from data, or reduce the dimension of available information.
- Also known as data mining, data science, statistical learning, or statistics.
- Or econometrics, if you are in my tribe.

Fundamental distinction: most MLA are designed to reproduce how a human would classify something, with all inherent biases. No pretension to deep structural parameters or causal inference—but this is changing.

Unsupervised MLA: no labels (no outcome data)

- Clustering: **cluster kmeans, kmedians**
- Principal component analysis: **pca**
- Latent class analysis: **gsem** in Stata 15

Supervised MLA: labels (outcome y)

- Regression or linear discriminants: **regress, discrim lda**
- Nonlinear discriminants: **discrim knn**
- Shrinkage: lasso, ridge regression, **findit lassopack**
- Generalized additive models (**findit gam**), wavelets, splines (**mkspline**)
- Nonparametric regress e.g. **lpoly, npregress**
- Support Vector Machines or kernel machines
- "Structural" Equation Models e.g. **sem, gsem, irt, fmm**
- Tree builders such as ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), CART (Breiman et al., 1984)
- Neural Networks (NN), Convolutional NN
- Boosting e.g. AdaBoost
- Bagging e.g. RandomForest

The big 3

These last 3 are what are usually meant by Machine Learning.

NN and Convolutional NN are widely used in parsing images e.g. satellite photos (see also Nichols and Nisar 2017).

Boosting and bagging are based on trees (CART), but Breiman (2001) showed bagging was consistent whereas boosting need not be.

Hastie, Tibshirani, and Friedman (2009; Sect. 10.7) outline some other advantages of bagging.

The Netflix Prize

The Netflix Prize was a competition to better predict user ratings for films, based on previous ratings of Netflix users.

The best predictor that beat the existing Netflix algorithm (Cinematch) by more than 10 percent would win a million dollars. There were also annual progress prizes for major improvements over previous leaders (one percent or greater reductions in RMSE).

The Netflix competition began on October 2, 2006, and 6 days later, one team had already beaten Cinematch. Over the second year of the competition, only three teams reached the leading position: BellKor, BigChaos, and BellKor in BigChaos, a joint team of the two other teams.

More exciting than the World Cup

On June 26, 2009, BellKor's Pragmatic Chaos, a merger of Bellkor in BigChaos and Pragmatic Theory, achieved a 10.05 percent improvement over Cinematch, making them eligible for the $1m grand prize. On July 25, 2009, The Ensemble (a merger of Grand Prize Team and Opera Solutions and Vandelay United) achieved a 10.09 percent improvement over Cinematch.

On July 26, 2009, the final standings showed two teams beating the minimum requirements for the Grand Prize: The Ensemble and BellKor's Pragmatic Chaos.

On September 18, 2009, Netflix announced BellKor's Pragmatic Chaos as the winner. The Ensemble had in fact matched the performance of BellKor's Pragmatic Chaos, but since BellKor's Pragmatic Chaos submitted their method in the final round of submissions 20 minutes earlier, the rules made them the winner.

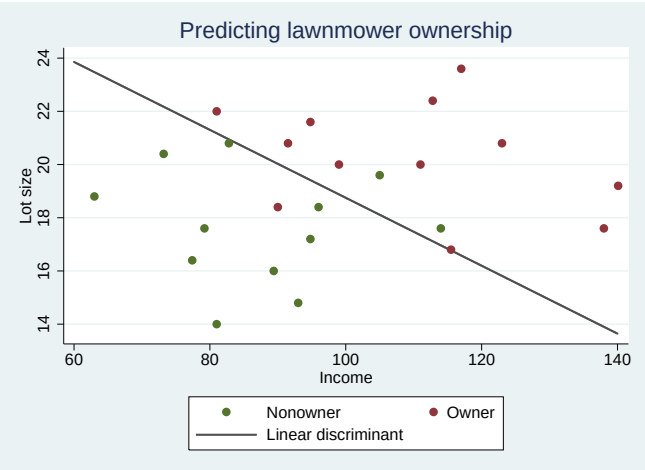
kaggle competitions

There are many of these types of competitions posted at kaggle.com at any given time, some with large cash prizes (active right now: Zillow home price prediction for $1.2m and Dept. of Homeland Security passenger screening for $1.5m).

Virtually all of the development in this methods space is being done in R and Python (since Breiman passed away, there is less f77 code being written).

Discriminants

The linear discriminant method draws a line (hyperplane) between data points such that as many data points in group 1 are on one side and as many data points in group 1 are on the other as possible. For example, a company surveys 24 people in town as to whether they own lawnmowers or not, and wants to classify based on the two variables shown. The line shown separates "optimally" among all possible lines (Fisher 1934). A similar approach can classify mushrooms as poisonous or not. Or we can use a semiparametric version averaging over the k nearest neighbors (both subcommands of **discrim**).



A punny example

From the Stata manual:

Example 3 of [MV] discrim knn classifies poisonous and edible mushrooms. Misclassifying poisonous mushrooms as edible is a big deal at dinnertime.

...

You have invited some scientist friends over for dinner, including Mr. Mushroom ... a real "fun guy"

A punny example, cont.

From the Stata manual:

Because of the size of the dataset and the number of indicator variables created by xi, KNN analysis is slow. You decide to discriminate based on 2,000 points selected at random, approximately a third of the data.

...

In some settings, these results would be considered good. Of the original 2,000 mushrooms, you see that only 29 poisonous mushrooms have been misclassified as edible.

A punny example, cont.

[use priors to increase the cost of misclassifying poisonous mushrooms, then...]

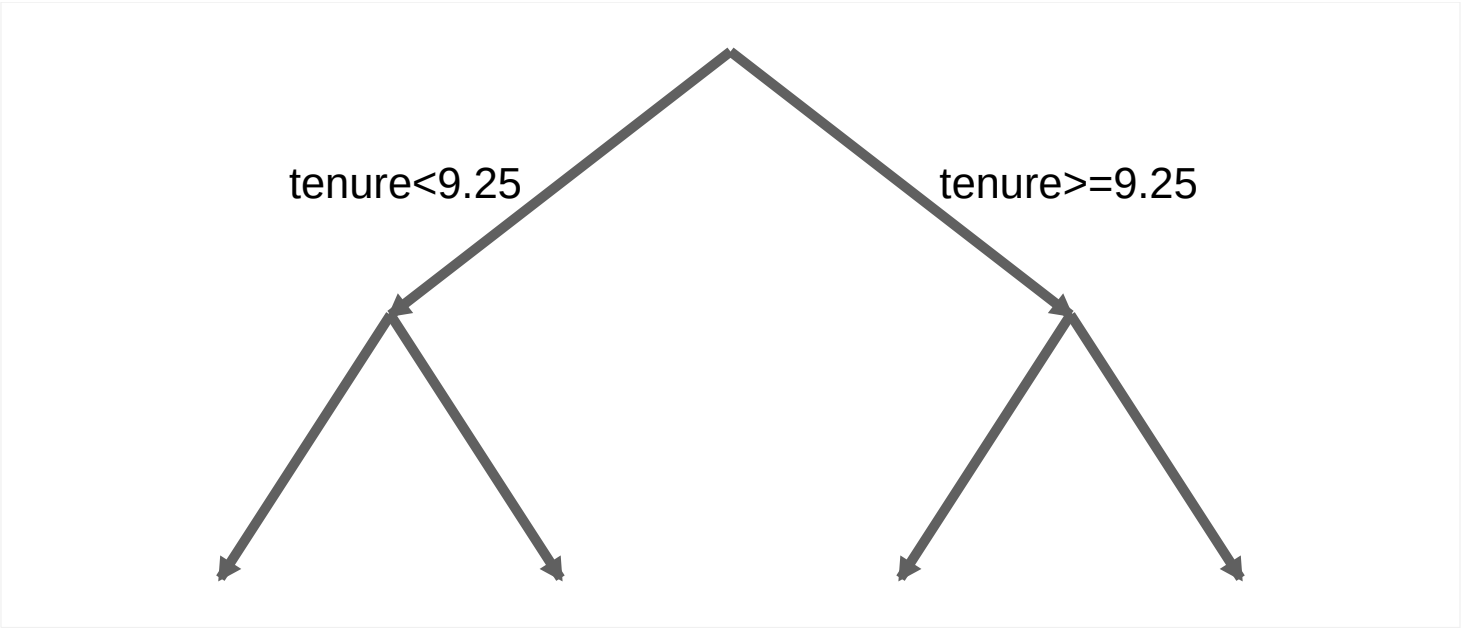
These results are reassuring. There are no misclassified poisonous mushrooms, although 185 edible mushrooms of the total 2,000 mushrooms in our model are misclassified.

...

This is altogether reassuring. Again, no poisonous mushrooms were misclassified. Perhaps there is no need to worry about dinnertime disasters, even with a fungus among us. You are so relieved that you plan on serving a Jello dessert to cap off the evening—your guests will enjoy a mold to behold. Under the circumstances, you think doing so might just be a "morel" imperative.

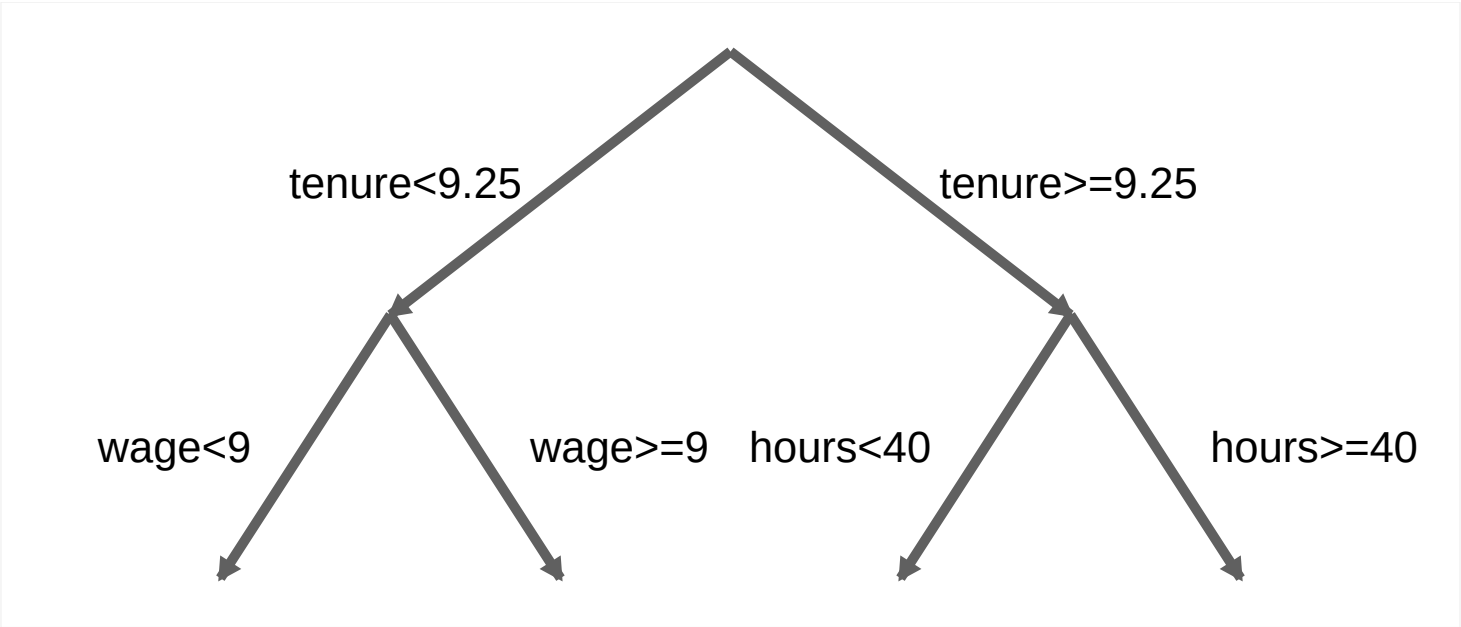
Trees

Ensembles can use a variety of models. A tree is one kind of model, shown classifying into two groups below.



Trees level 2

At each node, we can then classify again; note that the feature (variable) used to classify can differ across nodes at the same level.



Trees, branches, leaves

Can select branches optimally according to some criterion at each branching point, or can select a random cut point of a randomly selected variable. Can have multiple branches from each node or only two (we will focus on these binary splits).

It's very easy for even such a simple model to produce some complex computations. With 10 levels of nodes with binary splits, a tree has $2^{10} = 1,024$ terminal nodes ("leaves" at the ends of branches).

Forests

An ensemble method constructs many models on subsets of variables and data (sampling with replacement) and averages across them. The key developments are described in Breiman (1999): bootstrap then look across random subsets of features at each node. This type of stochastic ensemble method adds randomness to the choice of models, rather than constructing optimal models on each subset. This has the advantage of "de-correlating" models, which can reduce total variance.

A side benefit of bootstrapping is that the out-of-sample error is predicted by estimating the model in each randomly selected subset (sometimes called the "bag"), and using the balance of the data ("out of bag") to assess error. Because we use a random subset, the measured out-of-sample error is an unbiased estimator of true out-of-sample prediction error.

Estimating out-of-sample

The ensemble of models can predict for any new observations with the same set of variables defined.

In McBride and Nichols (2016), it turns out that choosing the parametric model that performs best out of sample also dramatically improves prediction. It's really the holdout observations, and prioritizing out-of-sample performance, that drives the improvement. A k-fold cross-validation approach would also do this.

Causal Inference

Nichols and McBride (2017) make the point that prediction is exactly the target for a propensity score model (as in **teffects ipw** or **teffects ipwra** etc.), though better predictions are not always better! In particular, if one estimates the probability of treatment as a function of excluded instruments, and not every confounder, a better predicted probability of treatment can lead to worse inference.

Comparing across many of these methods, bagging (RandomForest) worked best, in the sense that it had the lowest MSE for the true treatment effect.

Direction

For the rest of this talk, we will focus on the winner in that prior work, but the goal is to implement a stochastic ensemble method from scratch, with an eye toward tweaks in the method that can improve causal inference.

The code is not public yet, but email me if you'd like to be a beta tester. It is currently called **stens**, for "stochastic ensemble" method.

Overview

Basic method uses CART: binary splits that minimize "impurity" (entropy/gini/twoing). In a regression tree, the split is based on sum of squared residuals, which is the default in **stens**. Note that the sum of squared residuals for a binary outcome is just the number of observations misclassified.

Each leaf in a complete tree is captured by a single dummy built of interaction terms. The prediction is either a classification (predicted class) for that leaf, or an average outcome $\bar{y}$ for that leaf.

Breiman et al. (1984) advocate pruning a complete tree and using cross-validation. Pruning in such a system means combining dummies via an OR operation.

Breiman (1996) instead advocates no pruning and instead using bootstrap aggregation.

Outline of code

1. Bootstrap data to create matrix d
2. Randomly choose columns of d on which to split at current node
3. Compute optimal splits over all choices, pick best to create d0 and d1 submatrices
4. Store best choice (dummy syntax and predicted value) in prediction matrix
5. Repeat steps 2-4 in submatrices until stoppingrule met in all
6. Collect results in prediction matrix for this tree (number of leaves by 2)
7. Repeat steps 1-6 until treelimit met

Along the way, compute proximities between each pair of observations as the fraction of the time they fall in the same leaf. Also compute "variable importance" by permuting each feature used in the tree in the out-of-bad sample and computing the difference in prediction error.

Step 1. Bootstrap

Approximately 63.212 percent of observations are used for each tree: these are the "bag," in Breiman's parlance.

About 36.788 percent are the out-of-bag sample, a randomly selected sample that can be used to assess out-of-sample performance of the built tree.

Can also draw clusters for each sample rather than observations.

Step 2. Sample features

At each node, of the m features (predictor variables), randomly select $k << m$ variables to assess candidate splits; default is k=floor(sqrt(m)).

Can also check linear combinations of those k features; default is to check each pair by computing a linear discriminant. For 10 randomly chosen features, this implies 55 comparisons; if we instead compute all possible combinations, we have to compute at least 1,023 comparisons.

Step 3. Compute Impurity

Candidate splits are judged based on an impurity measure (how unalike are the resulting two branches).

Impurity measure options include Gini, entropy, twoing, squared prediction error.

Default is squared prediction error (the "regression" option of a Classification and Regression Tree).

Step 4. Store choice for each node

At each node, store the predictions for each branch together with the syntax for a dummy that generates it e.g. "(tenure>=9.5)*(hours<40)" in the example 2-level tree.

Step 5. Loop until stop

Repeat this process for subnodes unless a *stoppingrule* condition is met:

If all observations at a node are in one class (or have zero variance), or too few observations (below a user-specified limit), do not split the node.

If no nodes remain eligible for splitting, or the maximum tree depth (a user-specified limit) is reached, this tree is finished.

Step 6. Store predictions

For a categorical outcome: at terminal nodes, all observations are in same class unless a stopping rule is reached: maximum depth—by default, the minimum of ten splits for 1,024 leaves and floor($\log_2(N)$), or all nodes have reached the minimum number of observation to split (default is 4). With a continuous outcome and regression tree, the stopping rule plays a larger role—but can be useful even for a binary outcome. For the binary outcome case, can predict Pr(T=1) using mean, but by default predict as $(N_T + 1)/(N + 2)$. Classify by default using highest probability, or for binary case the Boolean $Pr(T = 1) > 0.5$.

Can also construct ROC curves.

Can also choose priors to weigh different classification errors differently.

Step 7. Repeat

Now, bootstrap again and build a new stochastic tree. Keep doing this either until maximum number of trees is reached (a user-specified limit with a default of 500) or a "change in prediction" limit is reached (e.g. predictions have changed less than c(epsdouble) over the last 10 new trees).

Caveats

Currently does not handle missing values except via a trick proposed by Breiman:

1. impute randomly via hotdeck,
2. run **stens** and predict proximity,
3. impute using nearest cases (using proximity and kNN), then
4. rerun **stens**.

Does not handle nonbinary splits except through repeated splits.

Currently a mix of ado and Mata code. Needs to be made faster—several ways forward here.

Horizon

Big innovation to come: Estimate a causal model in each tree

- Predict (noisily) the probability of treatment in each tree,
- Reweight within tree for ATE or ATT, then
- Average over trees for overall average impact estimate.

But also, assess dependence of impact estimate on:

- out of sample prediction error rate,
- distance from average prediction, or
- permutations in one feature.

Breiman, Leo, Jerome Friedman, Richard A. Olshen, and Charles J. Stone. 1984. *Classification and Regression Trees.* Wadsworth, New York.

Breiman, Leo. 1996. "Bagging predictors." *Machine Learning,* 24(2): 123140.

Breiman, Leo. 2001. "Random forests." *Machine Learning,* 45(1): 532.

Fisher, R. A. 1936. "The use of multiple measurements in taxonomic problems." *Annals of Eugenics,* 7: 179–188.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer, New York.

McBride, Linden, and Austin Nichols. 2016. "Retooling Poverty Targeting Using Out-of-Sample Validation and Machine Learning." The World Bank Economic Review.

Nichols, Austin, and Linden McBride. 2017. "Propensity scores and causal inference using machine learning methods." https://www.stata.com/meeting/baltimore17/

Nichols, Austin, and Hiren Nisar. 2017. "Analyzing satellite data in Stata." https://www.stata.com/meeting/baltimore17/