Predicting Wages 2: Comparison of Different Models

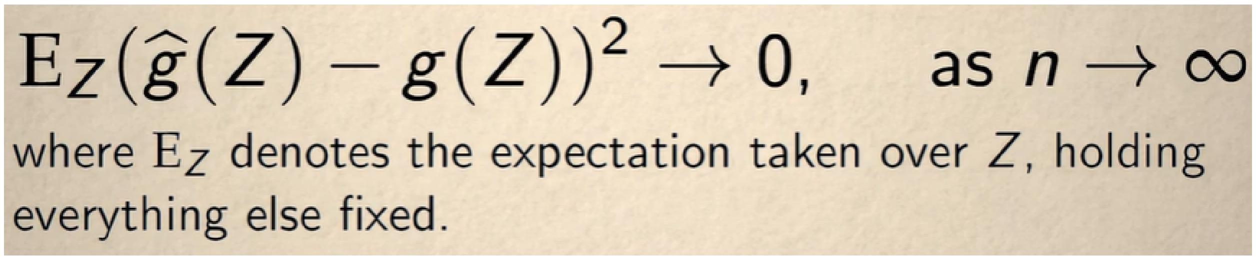
In this segment, we will discuss the quality of prediction that modern regression methods provide.

Recall that the best prediction rule for outcome Y using features or regressors Z is the function g(Z), equal to the conditional expectation of Y using Z:

$$g(Z) = E(Y \mid Z)$$

Modern regression methods, namely **Lasso, Random Forest, Boosted Trees and Neural Networks** when appropriately tuned, and under some regularity conditions, provide estimated prediction rules $\widehat{g}$ (Z) that approximate the best prediction rule g(Z) quite well.

Theoretical work demonstrates that under appropriate regularity conditions, and with appropriate choices of tuning parameters, the mean squared approximation error can be small once the sample size n is sufficiently large.



$$\mathrm{E}_Z(\widehat{g}(Z) - g(Z))^2 \to 0, \qquad \text{as } n \to \infty$$

where E_Z denotes the expectation taken over Z, holding everything else fixed.

These results do rely on various structural assumptions, such as sparsity in the case of Lasso and other such assumptions, to deliver this guarantee in modern, high-dimensional settings, where the number of features is large.

From a practical standpoint, we expect that under these conditions, the sample MSE, and R^2 will tend to agree with out-of-sample MSE and R^2.

We can measure the out-of-sample performance empirically by performing **data splitting**, as we did in classical linear regression.

Recall that:
1. We use a random part of the data (say half of the data) for **estimating or training the prediction rules.**
2. We use the other part of the data to **evaluate the predictive performance of the rule**, recording the out-of-sample MSE or R^2.

Accordingly, we call the first part of the data **the training sample** and the second part of the data **the testing or validation sample.**

Indeed, suppose we use ‘n’ observations for training and ‘m’ for testing or validation. And let, V denote the indices of the observations in the test sample. Then the out-of-sample or test MSE is defined as the average squared prediction error, where we predict Y_K in the test sample by $\hat{g}$ (Z_K), where the prediction rule $\hat{g}$ was computed on the training sample.

$$MSE_{test} = \frac{1}{m} \sum_{k \in V} (Y_k - \hat{g}(Z_k))^2$$

The out of sample R^2 is defined accordingly,

$$R^2_{test} = 1 - \frac{MSE_{test}}{\frac{1}{m} \sum_{k \in V} Y_k^2}$$

We next illustrate these ideas using a data set of 12,697 observations from the March Current Population Survey Supplement 2015.

In this data set, the outcome observations Y_i's are **log wages** (natural logarithms of the actual wages) of never-married workers living in the United States.

The feature Z_i's consist of a variety of worker characteristics including experience, race, education, 23 industry and 22 occupation indicators, and some other characteristics.

We will estimate or train **two sets** of prediction rules, **linear and nonlinear models.**

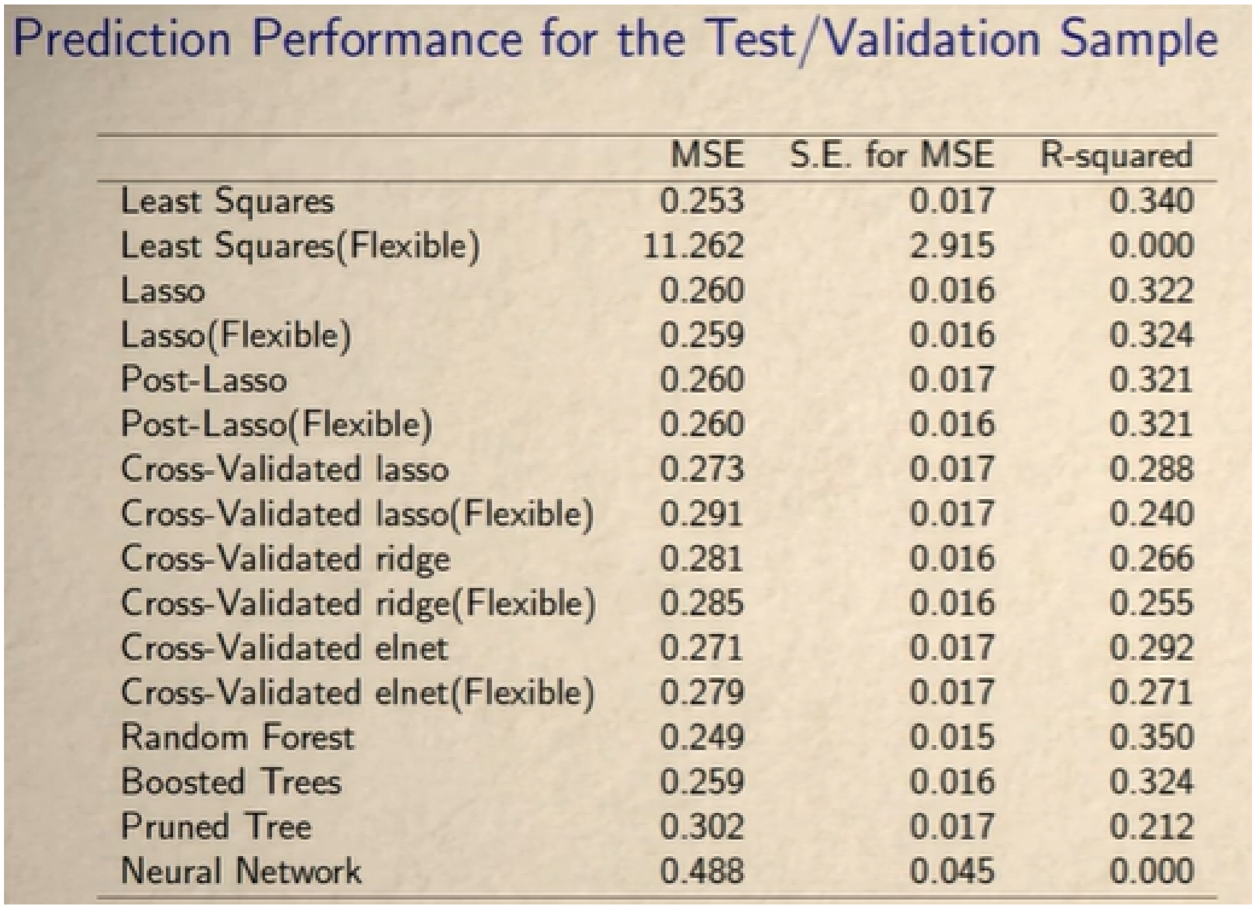
For the linear models, we estimate the prediction rule or the form $\widehat{g}(Z) = \widehat{\beta}' X$, with X generated in **two ways:**

1. **In the basic model**, X consists of 72 raw regressors (Z) and a constant.
2. **In the flexible model,** X consists of 2336 constructed regressors, which include the raw regressor Z, four polynomials in experience, and all two way interactions of these regressors.

We estimate $\widehat{\beta}$ by linear regression / least squares and by penalized regression methods: Lasso, Post-Lasso, Cross-Validated Lasso, Ridge, and Elastic Nets.

For the non-linear models however, we estimate the prediction rule of the form $\widehat{g}(Z)$. We estimate it through Regression Trees, Random Forests, Boosted Trees and Neural Networks.

We present the results in the following table, reporting the results for a single equal split of data into the training and testing part:



Prediction Performance for the Test/Validation Sample

	MSE	S.E. for MSE	R-squared
Least Squares	0.253	0.017	0.340
Least Squares(Flexible)	11.262	2.915	0.000
Lasso	0.260	0.016	0.322
Lasso(Flexible)	0.259	0.016	0.324
Post-Lasso	0.260	0.017	0.321
Post-Lasso(Flexible)	0.260	0.016	0.321
Cross-Validated lasso	0.273	0.017	0.288
Cross-Validated lasso(Flexible)	0.291	0.017	0.240
Cross-Validated ridge	0.281	0.016	0.266
Cross-Validated ridge(Flexible)	0.285	0.016	0.255
Cross-Validated elnet	0.271	0.017	0.292
Cross-Validated elnet(Flexible)	0.279	0.017	0.271
Random Forest	0.249	0.015	0.350
Boosted Trees	0.259	0.016	0.324
Pruned Tree	0.302	0.017	0.212
Neural Network	0.488	0.045	0.000

This table shows the Test MSE in the first column, the Standard Error for the MSE in the second column, and the Test R^2 in column three.

We see that the **prediction rule produced by Random Forest performs the best** for this example, giving the lowest out-of-sample or test MSE and the highest test R^2.

Other methods, for example, Lasso, Cross-Validated Elastic Net, and Boosted Trees perform nearly as well. They all perform similarly, with the Test MSEs being within one standard error of each other.

Remarkably, the simple classical OLS on a simple model with 72 regressors, performs extremely well in comparison - almost as well as the Random Forest.

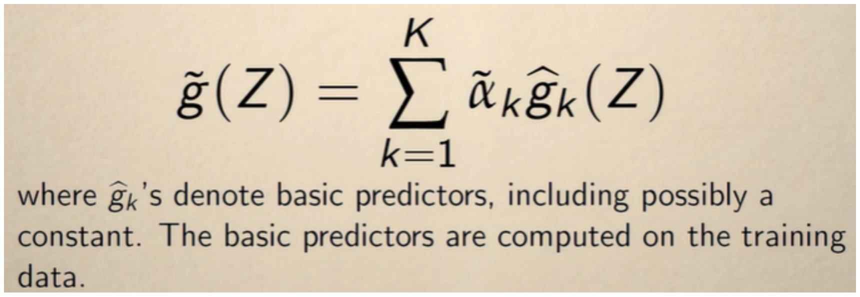
The performance of OLS, a simple linear model, is **statistically indistinguishable** from the performance of the Random Forest. The additional non-linear complexity of the Random Forest regressor has not made a significant improvement on the prediction performance of this task, and hence, we may choose **the simple OLS method to be the winner here** as it has achieved the best balance between complexity and performance.

On the other hand, classical OLS done on a flexible model with 2335 regressors performs very poorly. It provides a prediction rule that is very noisy and imprecise.

Penalized regression methods on the flexible model, such as Lasso, do much better because they are able to reduce the imprecision while leveraging the low approximation error/bias of the flexible model.

Pruned Regression Trees and simple Neural Networks with a small number of neurons don't perform well here. This is because these methods provide an approximation to the best prediction rule that is too crude, resulting in too much bias relative to other methods.

Given the results presented, we can choose one of the best-performing prediction rules. For example, we can select the prediction rule generated by least-squares on the simple model or the prediction rule generated by Lasso on the flexible model, or even the prediction rule generated by the Random Forest. We can also consider aggregations or ensembles of prediction rules which combine several prediction rules into one. Specifically, we can consider an aggregated prediction rule of the form:



$$\tilde{g}(Z) = \sum_{k=1}^{K} \tilde{\alpha}_k \hat{g}_k(Z)$$

where $\hat{g}_k$'s denote basic predictors, including possibly a constant. The basic predictors are computed on the training data.

$\tilde{\alpha}_k$ Here, tilde α_K is the coefficient assigned to the different prediction rules.

We can build K prediction rules from the training data, we can figure out the aggregation coefficients tilde α_K 's using the test and validation data.

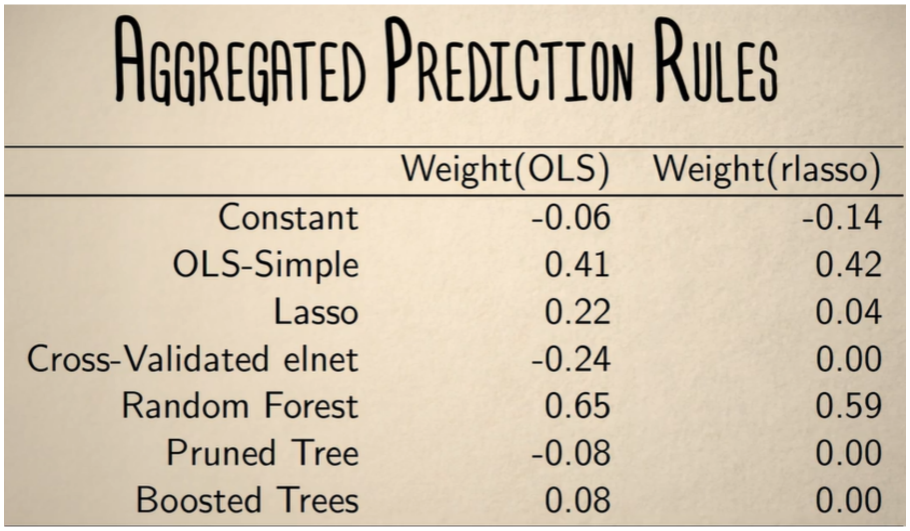
If the number of prediction rules K is small, we can figure out the aggregation coefficients using test data by simply running least-squares of outcomes on the predicted values in the test sample, where we minimize the sum of squared prediction errors on predicting Y_i by the linear combination of prediction rules in the test sample.

$$\min_{(\alpha_k)_{k=1}^{K}} \sum_{i \in V} \Big(Y_i - \sum_{k=1}^{K} \alpha_k \widehat{g}_k(Z_i)\Big)^2 .$$

If K is large however, we can do **Lasso aggregation** instead, where we minimize the sum of squared prediction errors of predicting Y_i by the linear combination of prediction rules in a test sample **plus a penalty term which penalizes the size of the coefficient.**

$$\min_{(\alpha_k)_{k=1}^{K}} \sum_{i \in V} \Big(Y_i - \sum_{k=1}^{K} \alpha_k \widehat{g}_k(Z_i)\Big)^2 + \lambda \sum_{k=1}^{K} |\alpha_k|$$

Let's illustrate this idea in our case study. Here we consider aggregating prediction rules based on Post-Lasso, Elastic Net, Random Forest, Boosted Trees, and Neural Networks. We estimated the aggregation coefficients using least squares and Lasso.



AGGREGATED PREDICTION RULES

	Weight(OLS)	Weight(rlasso)
Constant	-0.06	-0.14
OLS-Simple	0.41	0.42
Lasso	0.22	0.04
Cross-Validated elnet	-0.24	0.00
Random Forest	0.65	0.59
Pruned Tree	-0.08	0.00
Boosted Trees	0.08	0.00

From the estimated coefficients reported in this table:

We see that **most of the weight goes to the prediction rules generated by least squares on a simple model, and by the Random Forest**. Other prediction rules receive considerably less weight. Moreover, **the adjusted R^2 for the test sample gets improved** from 35% obtained by the random forest **to about 36.5% obtained by the aggregated methods.**

So let us summarize the above ideas. We discussed the assessment of the predictive performance of multiple linear and non-linear regression methods using the splitting of data into training and testing samples. The results could be used to pick the best prediction rule or to aggregate prediction rules into an ensemble, which can also result in some improvement in predictive performance.

We have also illustrated all these ideas in a case study above, using a recent wage data set from the March Current Population Survey Supplement 2015.