Extreme Gradient Boosting Regression to Predict Recent Estimated Glomerular Filtration Rate Results

**Christopher Haddad, Sanjay Roberts, Jeff Coady**

University of Denver Denver, CO, USA

“If linear regression was a Toyota Camry, then gradient boosting would be a UH-60 Blackhawk Helicopter,” Ben Gorman, A Kaggle Master Explains Gradient Boosting.

Abstract

In this study, the authors implement Extreme Gradient Boosting (xGboost) regression to predict the most recent estimated glomerular filtration rate (eGFR) from patient demographic, vitals, and lab-results. EGFR is a measure of the health of a patient’s kidneys, with a lower score indicating poor kidney health. There are many different versions of gradient boosting but what differentiates xGboost is its use of a more regularized model formalization to control over-fitting, which gives it better performance (Chen & Guestrin, 2016). We will put this concept to the test in two ways. The first is to leverage ‘importance’ from xGboost as it is a decision tree algorithm, to identify potential overfitting. We can then test whether xGboost has properly overcome this overfitting by iterating through all features, iteratively dropping the least important features, keeping track of the disparity between the training set accuracy (R-squared) and the validation set accuracy. Next, we’ll use the R package Caret’s implementation of grid search, called expand-grid, to tune the ratio of the attributes to be subsampled, the depth of the tree, and the learning rate. Tuning these parameters with a highly regularized function, one that limits model complexity, will result in a higher preforming model with little overfitting.

1 Introduction

How to model regression is a critical issue in many health care tasks. Estimated glomerular filtration rate (eGFR) as an indication of kidney health is of particular importance as kidney disease is the ninth leading cause of death in United States. The use of a highly regularized decision trees to predict a patient’s upcoming eGFR score would have far reaching implications when considering the large datasets coming from Electronic Medical Record companies like Allscripts. In the pages below we explore one-hundred thousand patients with eighteen consecutive eGFR scores and twenty-five explanatory variables for the purposes of testing the accuracy of a grid-search tuned xGboost regression model.

This high number of patients and explanatory variables would be ripe for overfitting. However, the creator of xGboost, Tianqi Chen, purports that the high regularization of the algorithm will result in a model that is not overfit. To put this concept to the test we set up two experiments.

The first experiment looks at overfitting and xGboost with respect to feature selection. We use an experiment to test the accuracy of a model iteratively trained on fewer and fewer features, dropping the least important features one-by-one, on both a training and validation set. We then explore the results graphically with an eye towards overfitting.

The second experiment explores tuning the model using parameters that we would expect to result in an overfit model:

1. Column-sample-ratio
2. Learning-rate
3. Maximum depth of a tree

Increasing any one of these parameters will increase the complexity of a model (Chen & Guestrin, 2016). Further, setting a high number of iterations to train the model (n\_rounds) would typically result in an overfit model with a typical decision tree model.

2 Data Source

For this study we are using de-identified patient data from Allscripts’ Central Data Warehouse (CDW). The data initially contained information on around one-hundred and fifty thousand patients but was reduced using the following methods during the exploratory data analysis (EDA) phase.

1. Patients with less than 18 consecutive months of eGFR data were dropped.
2. For those remaining patients that had greater than 18 months of data the oldest values were dropped to force all patients to have the same number of eGFR scores.
3. A polynomial of degree one was used to interpolate missing values.
4. Patients with leading and trailing eGFR scores equal to zero were dropped.
5. The following demographic data was joined to those remaining patients and those patients without demographic data were dropped:
   1. Gender
   2. Age
6. The following vitals and lab results were joined to those remaining patients and those patients without valid values were dropped:
   1. BMI
   2. Weight
   3. Cholesterol
   4. Systolic
   5. Diastolic

What remained was around one-hundred thousand patients in a flat-file populated with non-zero and no missing values.

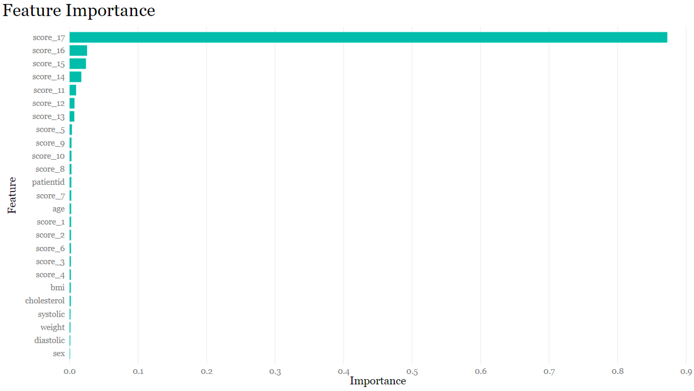
The dataset was then split 50/25/25 train, validation, test respectively.

3 Method

Gradient boosted trees compose a machine learning algorithm used for classification and regression tasks (Friedman, 2001; Mason, Baxter, Bartlett & Frean, 1999). It is an ensemble method, meaning it generates many weak prediciton models that are combined to create a single model. Whereas bagging methods like Random Forest generate a number of independent predictive models, boosting methods like Gradient Boosting generate a sequence of models, each dependent on previous models, culminating in an optimal predictive model.

The algorithm starts by generating a tree with just a root, effectively predicting the dependent measure’s group mean as the value for each case. The model then calculates (pseudo-) residuals and builds a second tree based on those residuals. At this step, the model begins to predict residuals instead of observed measures. Individual residuals are shunted to their appropriate leaves, and because there should be more observations than leaves, a mean is calculated for each leaf. To avoid overfitting, residuals are scaled by a learning rate between 0 and 1. The prediction at this step is then the original prediciton (the group mean) plus the residual scaled by the learning rate. This process continues until the residuals stabilize, or the model reaches a previously-specified max number of iterations. The final predicted value is the group mean plus the residuals at each step scaled by the learning rate.

The ‘gradient boosting’ part refers to the process of gradient descent, which is an iterative process where model weights are incremented gradually. Friedman pointed out that many small steps in the right direction give optimal predictions (2001). The ‘trees’ part refers to the decision trees generated at each iteration. These are the weak predictive models that are combined to form an optimal model. In practice, simple regression on residuals can replace decision trees.

Because this is a machine learning algorithm, there are a number of parameters that can be controlled by the user. Common ones to control are learning rate, number of trees, tree depth, and regularization. First, learning rate directly affects the loss function, *L(yobsv, ypred),* typically MSE. It takes values between 0 and 1, and as described above, ensures gradual incrementation toward an optimal solution. Second, number of trees (iterations) specifies how long the model will continue learning. Setting a low number of iterations has the effect of limiting weights, similar to a Ridge regression. Third, tree depth limits the number of branchings for each decision tree, thereby also limiting overfitting. Finally, regularization is specific to xGboost, and limits the construction of decision trees at each step. Trees are generated using a category purity score such as a Gini index, and regularization accepts a not-so-stringent level of impurity, which also serves to limit tree depth.

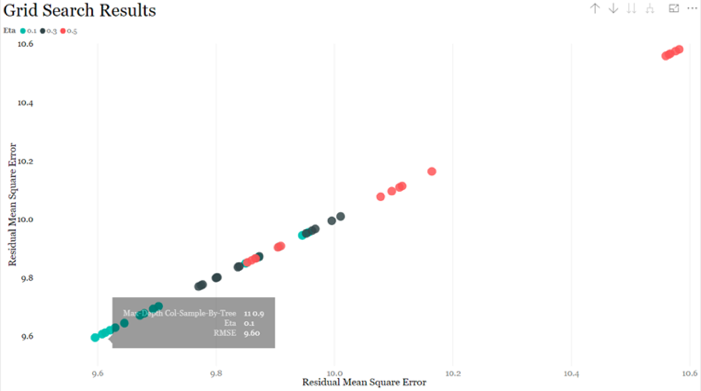
A strength of gradient boosting is that it is an all-purpose algorithm that can handle a variety of functions and distributions. It typically succeeds at finding complex patterns in the data because weights are incremented iteratively. It suffers, though, in that it can overfit training data, especially noisy data, also because of gradual gradient descent. It can also take longer to train because trees are created sequentially (Natekin & Knoll, 2013).

4 Application

One output of decision trees is feature importance, or rankings of the most important features. XGboost is no exception, providing a dataframe of feature importance for each attribute. We first used the default parameters of xGboost to fit a model on our training data only. We used this feature importance as a baseline for our autocorrelation and model complexity, both of which we aimed to reduce (Figure 1).

Figure 1. Default Model Feature Importance

We then leveraged the feature importance output of xGboost to run an experiment which looked at the accuracy of our model trained on fewer and fewer features, dropping the least important features one-by-one according to the output of the xGboost model with default parameters.

For example, the first iteration (looking at figure 1) would train a model on ‘score\_17’ through ‘diastolic’ dropping ‘sex.’ The next iteration would train a model on ‘score\_17’ through ‘weight’ dropping ‘diastolic’ and ‘sex,’ and so on.

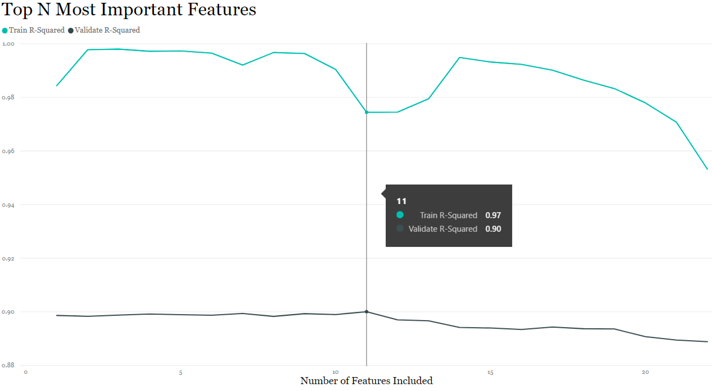
We then graphed the accuracy of our model prediction on training data and on validation data, recording the values of each seperately. The output of this experiment can be seen in Figure 2.

Figure 3 Results of Grid Search, eta, colSamp, maxDepth

Figure 2. Train vs Validation Accuracy, Feature Selection

As expected, when we passed all of these features to the model we see some apparent overfitting as indicated by the large gap between train and validation. Particularly around smaller x values (where x represents the number of features dropped in descending order of importance) we see that the prediction accuracy on the training data is at nearly one-hundred percent.

The optimal point that we were looking for, as indicated by the tooltip box, is where the accuracy on the training data and the accuracy on the validation data come closest to converging. This is the point of least overfitting with respect to our feature selection. The number of explanatory variables to keep is thus:

Once the number of features had been selected according to importance and reduction in overfitting, we then began our grid search using parameters expected to result in an overfit model. We did so with the expectation that the regularaization effect of xGboost would counter the overfitting parameters.

Figure 3 shows the results of our grid search. We tested the effect of the three parameters on our accuracy and we identified that learning rate ( had the greatest overall effect on accracy. We then graphed etaversus residual mean squared error on both the x and y axis. The result was a grouping by the three different eta values indicated by green, black, and red along the diagonal of Figure 3. The green grouping represents where , its lowest value. The transparent black box also shows a max-depth of eleven and column-sampling-ratio of 0.9 as the parameters to result in the lowest residual mean squared error. We therefore selected these parameter values for our final experiment, testing the model on unseen data.

4 Test on Unseen Data

Our final experiment was to test our optimal features as indicated during the first experiment with our optimal parameters selected from grid search on unseen data. The results of this experiment can be seen in Figure 4. This graph coalesces the R-squared

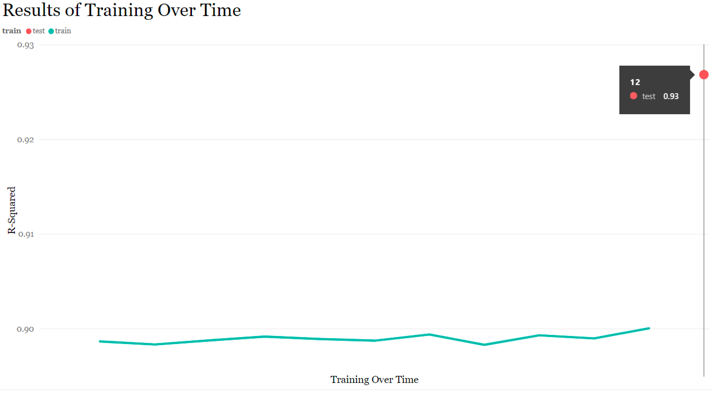
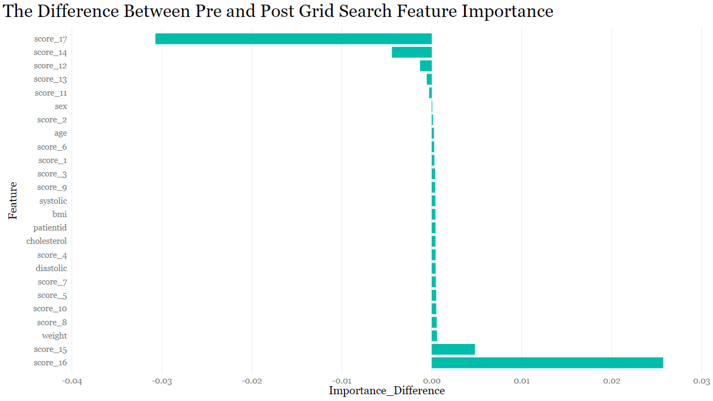
values from the above experiments with the final R-squared from the test of the optimized model on unseen data.

Figure 5 Difference in Importance Default v Final Model

Figure 4. Results of Model Training Over Time

The experiment shows that indeed xGboost’s regularaization resulted in a model that was not overfitted. This is indicated by the accuracy of the model on unseen data (the red dot in Figure 4 with the value 93%) being greater than any accuracy we had seen throughout our experiments on the validation data (the green line in Figure 4 which hovers around 90%). In other words, our optimized model performs exeptionally well on unseen data which suggest that the model did not over-learn distinct aspects of the training data and instead was able to extract the general patterns from that data to produce a model with a high prediction accuracy on unseen data. In the end, age and previous eGFR scores paired with a learning rate of 0.1 and max number of trees set to 100 provided the best predictive fit for time-18 eGFR scores.

Again we are able to examine importance from our optimized model and compare that ouput with our default model’s importance (Figure 5). This graph shows us that when we tuned out model we reduced the importance of our most recent eGFR score (‘score\_17’) not including our target (‘score\_18’) and increased the importance of the features that came just before ‘score\_17.’ This would suggest that our optimized model performed well with respect to reducing the effects of autocorrelation.

5 Further Testing

Given more time the team would have liked to continue exploring parameter tuning with grid search. The trend in Figure 3 makes it clear that the model would benefit from an even lower learning rate. Moreover, parameter tuning with a model with such a high degree of regularization is rather cavalier. That is, it might be fun to push the limits of our xGboost regularization via incremental increases in number of training rounds (‘n\_rounds’) with decreases in learning rate.

Undoubtably we could expect the residual mean squared error to continue to decline as we implemented this future grid search with a respective decrease in the importance of ‘score\_17, ’ but only to a point. More diagnostic plotting and visualizations might help to increase the confidence in our grid-search experiment.

6 References

Chen, T. & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In B. Krishnapuram (ed.), *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* NY: ACM.

Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics, 29,* 1189-1232.

Mason, L., Baxter, J., Bartlett, P.L. & Frean, M. (1999). Boosting algorithms as gradient descent. In S.A. Solla, T.K. Leen and K. Müller (eds.). *Advances in Neural Information Processing Systems 12.* Cambridge, MA: MIT Press.

Natekin, A. & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neuroscience, 7,* Article 21, 1-21.