# BART: BAckward Regression Trimming

Bart Baesens

I have been struggling a long time what title to come up with for this article. In the end, I decided to give it my own name. The reasons being manifold. First, it is a well-known public secret that the average academic exhibits a somewhat elevated degree of narcissism. Arguing that I am different than that might be proof of it itself, so I consider myself nothing more than the average academic. Secondly and more importantly, the procedure itself excels in simplicity which I consider to be one of my main goals in life (though I may not always be successful in that!). Finally, the method is far from new so I definitely don’t want to claim to be the inventor of it, see e.g. . <https://stackabuse.com/applying-wrapper-methods-in-python-for-feature-selection/>, but another reason for naming it BART is because it’s by far my favorite method for doing variable selection. However, feel free to use this approach and cite it in whatever way you deem suited.

Backward Regression Trimming (BART) is simple and as said I have used it so many times both in industry and academia each time finding that it works well despite the availability of much more scientifically grounded approaches. It is a variable selection procedure which aims at finding the most compact yet well performing analytical model (Baesens, 2014). Before we elaborate on this, a few starting observations based upon years of working in analytics, both scientifically and consulting-wise.

First, due to correlation between variables in the data, there is a clear difference between a variable being redundant and a variable being irrelevant. Consider the variables Age and TimeOnBooks for credit risk modeling. Both are clearly related to credit risk (older people are usually less risky) but also highly correlated (Baesens, Scheule, Rösche, 2016). Hence, one of them is redundant, but that doesn’t mean it’s not relevant to credit risk. It’s simply not needed in an analytical model since the information is already captured by the other variable. Summarizing this point, there are typically many good predictive analytical models with a minimum set of relevant variables. The key question when doing variable selection is to find one and only one of them! Put differently, no need to worry if alternative variable selection procedures come up with different variable subsets, as this is just the statistical nature of the problem.

Secondly, it is my firm belief that simple analytical techniques such as linear and logistic regression do a pretty good job for analyzing structured data sets as one commonly encounters in credit scoring, churn prediction, response modeling, fraud detection, software defect prediction, etc. Deep learning technologies are cool, but far from a panacea to all your analytical modeling challenges. In fact, it could easily lead up to some really deep problems when considering them your first line of analytical modeling. I can backup this claim scientifically with the various benchmarking studies we conducted and which are amongst my most cited papers (apologies if this once again sounds narcissistic!) (Baesens et al., 2003; Lessman et al., 2008; Lessmann et al., 2015; Van Gestel et al., 2004). Hence, my motivation to stick to linear or logistic regression models as your first line of models for doing variable selection.

Thirdly, I am also strongly convinced that many predictive analytical models do pretty well with 3 to 5 variables. When I see scientific papers reporting insignificant p-values of variables in models having 10 or more variables, I always wonder why not kick them out from the model and re-estimate it until only significant variables remain (note that I am deliberately not opening Pandora’s box here on the usage of significance, p-values and confidence intervals as discussed in a recent Nature contribution, see <https://www.nature.com/articles/d41586-019-00857-9>). I am quite confident the resulting interpretation of the model results and impact might change drastically. Mea culpa, I might have sinned against this in some of my earlier papers as well, but the nice thing about doing research is learning how you can continuously improve your own work and methods. The key question when doing variable selection is how to find this minimal set of relevant variables and make sure your model is as compact as possible.

Fourthly, multiple criteria need to be considered for variable selection. An obvious one is the performance, either statistically (e.g. AIC, AUC) or in terms of profit (e.g., our earlier developed expected maximum profit (EMP) measure). Next, there is the interpretation of the variable. It should be easy to interpret and validate by the business analyst. This pertains to not only the variable itself but also to its impact as measured by the sign of the estimated coefficient. An example, when your model says higher debt improves your creditworthiness, then any business analyst will refuse to use this. It’s against all common sense and intuition and makes it smell like a spurious correlation. The interpretability aspect also comes into play when considering interaction effects. Despite possibly strong statistical significance, interaction effects make the model harder to interpret to the non-statistically schooled business user. The operational efficiency refers to the resources needed to compute the variable. Trend variables (either absolute or relative) typically pop up as very predictive in many analytical models, but might require a lot of resources to compute, depending upon the time window adopted obviously. The economic cost refers to the cost of gathering the variable. Variables collected externally (e.g., credit bureau scores, credit ratings, data from data poolers) may come at a price. Variables should also be regulatory compliant in terms of privacy regulations (e.g. GDPR) or ethical frameworks. Finally, variables should have a sound definition. This especially applies to ratio variables in case the denominator can equal zero, or in case the ratio has a discontinuous effect on the outcome (e.g. when the numerator or denominator can have negative signs) when working with linear models (Van Gestel, Baesens, Martens, 2020).

Given the above observations, we are now ready to roll the drums and introduce BART. As said, it is simple, may be even not worth is own name, but remember the starting paragraph of this article. This is how it works in a classification setting:

* 1. Split the data in training, validation and test set
	2. Estimate a logistic regression model with all n variables on the training set and measure the performance (e.g. AUC, profit) on the validation set
	3. Estimate n logistic regression models with n-1 variables on the training set and proceed with the best one in terms of performance on the validation set
	4. Estimate all n-1 logistic regression models with n-2 variables on the training set and proceed with the best one in terms of performance on the validation set. Continue doing this until no variables left in the model.
	5. Choose the best model based on the validation set performance.
	6. Estimate a model on the combined training and validation set and measure the performance on the independent test set.

You can find the corresponding R and Python code on Github (<https://github.com/MariaOskarsdottir/BART>) and a worked out example in the Appendix. Credit to whom credit is due, so special thanks to Sebastiaan Höppner and María Óskarsdóttir for contributing to this. As you can see, the procedure is quite resource intensive and works in a backward way. For big data sets, it can be sped up by working on a sample thereof. It can also easily be parallelized since the models can be estimated in parallel during each step of the iteration so a multithreaded version should be easy to program. For smaller data sets, step 1 can be extended to a cross-validation setup giving us a more stable performance estimate with accompanying standard deviation and thus confidence interval at the same time. The latter can come in handy when deciding upon the cut-off or optimal variable subset as we discuss below. A couple of other observations follow.

Basically, BART can be used with any classifier and performance metric. As an example, it could even be used with random forests as an alternative to the gini and permutation based importance procedures. In a classification setting, I recommend the AUC in case you are interested in statistical performance (Baesens, 2014). When you are interested in profit, I can recommend our EMP measure which can be downloaded from <https://cran.r-project.org/web/packages/EMP/index.html> and which is available both for churn prediction and credit scoring (Verbraken et al., 2013; Verbraken et al., 2014). In regression, my favorite performance metric to use is the Pearson correlation between the actual and predicted values. Contrary to, e.g. MAD and (R)MSE, the Pearson correlation is scaled between
-1 and +1 which I find an attractive property for interpretation.

In contrast to more scientifically grounded approaches, such as LASSO and ridge regression (which also require an additional regularization parameter to carefully tune, btw!), BART effectively kicks out the variables from the model. Hence, no more need to spend time and expensive resources to collect and/or compute expensive variables (see my earlier point on the economic cost of variables) and then severely punish them in the model formulation anyway.

As an illustration, we applied BART to the HMEQ data set. This data set reports characteristics and delinquency information for 5,960 home equity loans. A home equity loan is a loan where the obligor uses the equity of his/her home as the underlying collateral. The data set has the following characteristics and is also available on Github <https://github.com/MariaOskarsdottir/BART>:

* BAD: 1=applicant defaulted on loan or seriously delinquent; 0=applicant paid loan
* LOAN: Amount of the loan request
* MORTDUE: Amount due on existing mortgage
* VALUE: Value of current property
* REASON: DebtCon=debt consolidation; HomeImp=home improvement
* JOB: Occupational categories
* YOJ: Years at present job
* DEROG: Number of major derogatory reports
* DELINQ: Number of delinquent credit lines
* CLAGE: Age of oldest credit line in months
* NINQ: Number of recent credit inquiries
* CLNO: Number of credit lines
* DEBTINC: Debt-to-income ratio

It’s not a big data set in terms of number of variables and observations as you can see, but it suffices for illustration purposes. Applying BART using logistic regression and AUC results in the following graph depicting the AUC on the validation set:



This is a very nice illustration of the effect of variable selection. You can see that in the beginning the performance stagnates and even increases afterwards. It nicely illustrates the destructive impact that an abundance of variables might have on model performance which relates back to my earlier point of always kicking out redundant variables and make analytical models as compact as possible. An obvious question is how to select the optimal variable subset. Remember, the above plot is on the validation set which is the data set that is typically used for model selection in predictive analytics. Personally, I would go for the common sense approach and choose 4 variables (JOB, DEROG, DELINQ and CLAGE) giving us a validation set AUC of 82.6%. Having applied this on many data sets, we found this approach to work well. It is usually somewhat sexier referred to as “the business expert approach” to cut-off determination. The AUC performance of this reduced model on the test set is 74.64% whereas the full model with all variables obtained 73.09% AUC. A more scientifically grounded approach to determine the optimal variable subset could start from the maximum AUC and then slide down the curve applying a statistical test (such as the DeLong, DeLong, Clarke Pearson (1988) test) each time comparing the corresponding AUCs and obviously making sure to neatly apply Bonferonni-wise corrections for multiple testing. The optimal variable subset is then found where the performance decreases significantly at e.g. a 99% confidence level. Another remark pertains to multicollinearity, a question I often get when presenting this procedure. Admitted, a nasty statistical problem since multicollinearity between variables is never perfect, yet can have a disturbing effect on the volatility of the parameter estimates. My take on it with BART is the following. In case of perfect multicollinearity, then obviously one variable will be kicked out (see my earlier point on variable relevance and redundancy). Although I don’t suspect this to be the case in many settings, it could however be that there remain variables which are collinear to a certain degree. However, since the multicollinearity is not perfect, they do measure something different and it is precisely this difference which justifies their needed presence in the predictive model to obtain the best performance (e.g., AUC or profit-wise). Hence, my argument to tolerate this modest form of multicollinearity for the sake of performance.

Finally, as a side remark, a variant of the above procedure can also be used to tune the hyperparameters of your analytical models. In fact, I recently used it to successfully tune the hyperparameter of a Yeo-Johnson transformation (one of my favorite transformations, btw!) to model the saturation effect of an amount variable on credit risk. We also extensively used it in our previous research to tune the regularization and bandwidth parameter of an RBF SVM and other predictive analytics techniques (Van Gestel et al., 2004).

To conclude, I hope you found this contribution useful and look forward to hearing your feedback if you would decide to use it in real-life. Once more, apologies for the name and feel free to use it and refer to it in any way you like! Science is all about progressing and disseminating knowledge and not about the people who claim to do this.

# Acknowledgments

I would like to wholeheartedly acknowledge the contribution of (in random order!) Katrien, Ann-Sophie, Victor, Hannelore, Tim, Nuria, Joachim, Karolien, Wouter, Luit, Maria, Stefanie, Jeroen, Pieter, Stéphanie, Gerard, Vera, Hugo, Rosa, Bart, Tine, Christophe, Thomas, Seppe, Jochen, Wilfried, Monique, Jan, Véronique, Inneke, Sebastiaan ,Pascal, Vincent, Gijs, Ann, Noémi, Sien, Katrien, Chloé, Sarah, Wouter, Stevie, Caroline, Tine, Johan, Thomas, Nathalie, Marina and Dee. They know why!

# References

Baesens B., *Analytics in a Big Data World*, Wiley, ISBN 1118892704, 2014.

Baesens B., Bapna R., Marsden J.R., Vanthienen J., Zhao J.L., Transformational Issues of Big Data and Analytics in Networked Business, *MIS Quarterly*, Volume 40, Number 4, pp. 1-12, 2016.

Baesens B., Roesch D., Scheule H., *Credit Risk Analytics – Measurement Techniques, Applications and Examples in SAS*, Wiley, 2016.

Baesens B., Van Gestel T., Viaene S., Stepanova M., Suykens J., Vanthienen J., Benchmarking State of the Art Classification Algorithms for Credit Scoring, *Journal of the Operational Research Society*, Volume 54, Number 6, pp. 627-635, 2003.

DeLong ER, DeLong DM, Clarke-Pearson DL, Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach, *Biometrika*, Sep;44(3), pp. 837-45, 1988.

Lessmann S., Baesens B., Mues C., Pietsch S., Benchmarking classification models for software defect prediction: A proposed framework and novel findings, *IEEE Transactions on Software Engineering*, Volume 34, Number 4, pp. 485-496, 2008.

Lessmann S., Baesens B., Seow H.V., Thomas L.C., Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research, *European Journal of Operational Research*, Volume 247, Issue 1, pp. 124–136, 2015.

Van Gestel T., Baesens B., Martens D., *Predictive Analytics -Techniques and Applications in Credit Risk Modelling*, Oxford University Press, 2020.

Van Gestel T., Suykens J., Baesens B., Viaene S., Vanthienen J., Dedene G., De Moor B., Vandewalle J., Benchmarking Least Squares Support Vector Machine Classifiers, *Machine Learning*, Volume 54, Issue 1, pp. 5-32, January 2004.

Verbraken T., Bravo C., Weber R., Baesens B., Development and application of consumer credit scoring models using profit-based classification measures, *European Journal of Operational Research*, Volume 238, pp.505–513, 2014.

Verbraken T., Verbeke W. Baesens B., A Novel Profit Maximizing Metric for Measuring Classification Performance of Customer Churn Prediction Models, *IEEE Transactions on Knowledge and Data Engineering*, Volume 25, Issue 5, pp. 961-973, 2013.

# Appendix

# -------------------------------------------------------------------------------------------------

# Written by Sebastiaan Höppner, 2019

# -------------------------------------------------------------------------------------------------

rm(list = ls()) # empty memory

# Load data ---------------------------------------------------------------------------------------

hmeq <- read.csv("hmeq.csv")

hmeq$BAD <- as.factor(hmeq$BAD) # convert BAD into factor (i.e. categorical) variable

# Since we're going to fit logistic regression models, we remove rows with NA values:

hmeq <- hmeq[complete.cases(hmeq), ]

# Load packages -----------------------------------------------------------------------------------

library(AUC)

library(caret)

# Split data in training, validation and test set -------------------------------------------------

library(caret)

set.seed(10)

index\_test <- createDataPartition(y = hmeq$BAD, times = 1, p = 0.2, list = FALSE)

index\_train <- createDataPartition(y = hmeq$BAD[-index\_test], times = 1, p = 0.8, list = FALSE)

test <- hmeq[ index\_test, ]

train <- hmeq[-index\_test, ]

valid <- train[-index\_train, ]

train <- train[ index\_train, ]

# Backward variable selection based on AUC --------------------------------------------------------

logit <- logit <- glm(BAD ~ ., data = train, family = "binomial")

predictions <- predict(logit, newdata = valid, type = "response")

AUC <- auc(roc(predictions, labels = valid$BAD))

removed\_predictors <- c()

predictor\_list <- list()

best\_AUCs <- c()

best\_AUCs <- c(best\_AUCs, AUC)

y\_train <- train$BAD

y\_valid <- valid$BAD

X\_train\_old <- train[, -1]

X\_valid\_old <- valid[, -1]

predictor\_list[[1]] <- colnames(X\_train\_old)

while (NCOL(X\_train\_old) > 1) {

 nPredictors <- NCOL(X\_train\_old)

 AUCs\_temp\_vec <- c()

 for (j in 1:nPredictors) {

 X\_train\_new <- X\_train\_old[, -j]

 X\_valid\_new <- X\_valid\_old[, -j]

 train\_new <- cbind.data.frame(BAD = y\_train, X\_train\_new)

 valid\_new <- cbind.data.frame(BAD = y\_valid, X\_valid\_new)

 colnames(train\_new) <- c("BAD", colnames(X\_train\_old)[-j])

 colnames(valid\_new) <- c("BAD", colnames(X\_valid\_old)[-j])

 logit <- glm(BAD ~ ., data = train\_new, family = "binomial")

 predictions <- predict(logit, newdata = valid\_new, type = "response")

 AUC <- auc(roc(predictions, labels = valid$BAD))

 AUCs\_temp\_vec <- c(AUCs\_temp\_vec, AUC)

 }

 best\_index <- which.max(AUCs\_temp\_vec)

 removed\_predictors <- c(removed\_predictors, colnames(X\_train\_old)[best\_index])

 best\_AUCs <- c(best\_AUCs, AUCs\_temp\_vec[best\_index])

 remaining\_predictors <- colnames(X\_train\_old)[-best\_index]

 predictor\_list[[length(predictor\_list) + 1]] <- remaining\_predictors

 X\_train\_old <- X\_train\_old[, -best\_index]

 X\_valid\_old <- X\_valid\_old[, -best\_index]

 train\_old <- cbind.data.frame(BAD = y\_train, X\_train\_old)

 valid\_old <- cbind.data.frame(BAD = y\_train, X\_train\_old)

 colnames(train\_old) <- c("BAD", remaining\_predictors)

 colnames(valid\_old) <- c("BAD", remaining\_predictors)

}

removed\_predictors <- c(removed\_predictors, remaining\_predictors)

logit <- glm(BAD ~ 1, data = train, family = "binomial")

predictions <- predict(logit, newdata = valid, type = "response")

AUC <- auc(roc(predictions, labels = valid$BAD))

best\_AUCs <- c(best\_AUCs, AUC)

# Plot results ------------------------------------------------------------------------------------

library(ggplot2)

df <- data.frame(AUC = best\_AUCs, nPredictors = (ncol(train)-1):0)

breaks <- seq(ncol(train)-1, 0, -2)

xlabels <- paste(breaks, "Vars")

xlabels[2:length(xlabels)] <- paste0(xlabels[2:length(xlabels)],

 "\n(", removed\_predictors[seq(2, ncol(train)-1, 2)], " out)")

auc\_graph <- ggplot(data = df, mapping = aes(x = nPredictors, y = AUC)) +

 geom\_point(shape = 17, size = 5, color = "dodgerblue") +

 geom\_line(size = 2, color = "dodgerblue") +

 ylab("AUC value") + xlab("Variables") +

 theme\_bw() + theme(text = element\_text(size = 22), legend.position = "none") +

 geom\_text(data = df, mapping = aes(x = nPredictors, y = AUC),

 label = round(df$AUC, 3), size = 7, nudge\_y = 0.015) +

 scale\_x\_reverse(breaks = breaks,

 labels = xlabels)

plot(auc\_graph)

# Estimate model on the combined training and validation set --------------------------------------

logit\_full <- glm(BAD ~ ., data = rbind.data.frame(train, valid), family = "binomial")

# Based on the AUC-graph we decide to use the reduced model with 4 predictor variables:

predictors\_reduced\_model <- predictor\_list[[9]]

print(predictors\_reduced\_model)

# [1] "JOB" "DEROG" "DELINQ" "CLAGE"

logit\_reduced <- glm(BAD ~ .,

 data = rbind.data.frame(train, valid)[, c("BAD", predictors\_reduced\_model)],

 family = "binomial")

# Measure the performance on the independent test set ---------------------------------------------

predictions\_full <- predict(logit\_full, newdata = test, type = "response")

AUC\_full <- auc(roc(predictions\_full, labels = test$BAD))

print(AUC\_full)

# [1] 0.7308562

predictions\_reduced <- predict(logit\_reduced, newdata = test, type = "response")

AUC\_reduced <- auc(roc(predictions\_reduced, labels = test$BAD))

print(AUC\_reduced)

# [1] 0.7464325