Final Project Report

Rahul Sangole

Mar 12, 2017

PREDICT - 422 Section 58

Table of Contents

Abstract 3
Final Project Report 4
Overview of Methodology Used 4
Data Understanding 4
DONR Response Variable 5
DAMT Response Variable 5
Data Preparation 6
Transformations 6
Data Sets Created for Training & Validation Data 7
Modeling Approach 8
Modeling – `donr` Response 9
Logit 9
LDA, QDA 9
kNN 9
Bagging 10
Boosting 10
Random Forest 11
Evaluation – `donr` 11
Training Data Comparison 11
Profit Maximization Calculation 12
Modeling Part II – `damt` Response 14
Principal Components Regression 14
Partial Least Squares 14
Elastic Net 14
Bagging 15
Boosting 16
Random Forest 16
Neural Net 17
Generalized Additive Models 17
Evaluation – `damt` 18
Training Data Comparison 18
Validation Data Comparison 18
Final Remarks 20
Final Model Selected for `donr` 20
Final Model Selected for `damt` 20
R Packages Used 21

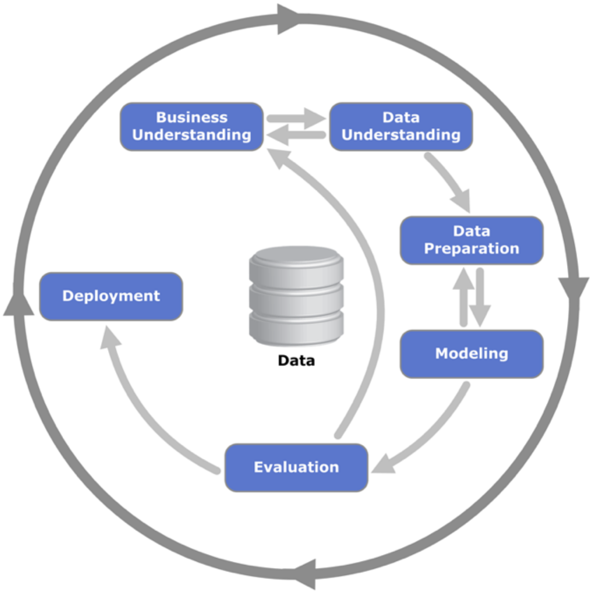
Abstract

This document reports the analysis performed to predict if recipients of direct marketing campaign materials by a charitable organization are likely to donate, and how much they would donate. Since there is a cost impact of sending the marketing material, the organization wants to prioritize the recipient list to maximize donations. A total of 46 models run across eight competing data preparation strategies are described, before a final model is proposed based off of common performance metrics.

Final Project Report

The data consist of donation information for a charitable organization. Observations list the historic response to a marketing campaign. The objective for the analysis is to develop two predictive models – a classification model for the response variable DONR, to predict if the person would donate or not, and a prediction model for the variable DAMT, which predicts how much the person would donate.

Overview of Methodology Used



The CRISP-DM (Cross Industry Standard Process for Data Mining) methodology is followed in this project. The figure below (Chapman, 2000) provides an overview of the life cycle of a data mining and modeling project such as this one in a CRISP-DM framework.

There are six key phases in the life cycle of a data mining project. These phases are not linearly arranged. The two way arrows and outer circle symbolize the cyclic nature of the task of data mining.

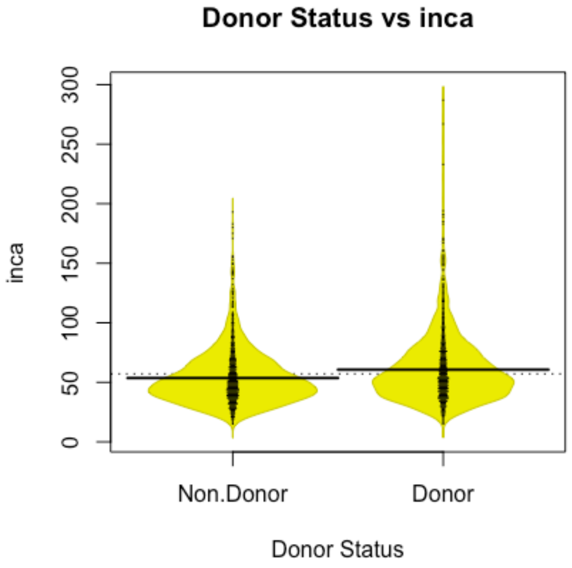
The input given to the analyst by the business has been an output of the first step – Business Understanding. This report focuses on the tasks of Data Understanding, Data Preparation, Modeling, and Evaluation. Each of these steps is summarized below.

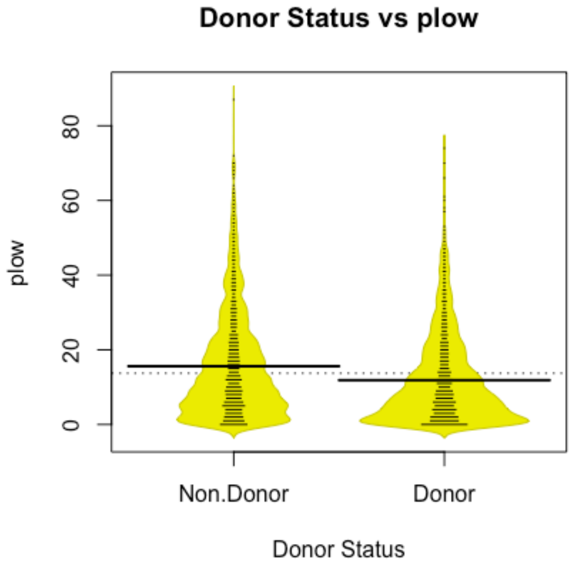
Data Understanding

This section focuses on developing an intimate understanding of the data. Exploratory data analysis in terms of univariate and bivariate statistics, as well as graphical methods are employed. The data set consists of 2 target variables and 20 predictor variables. A total of 8,009 observations in the data set. A few key highlights for the response variables are noted below.

DONR Response Variable

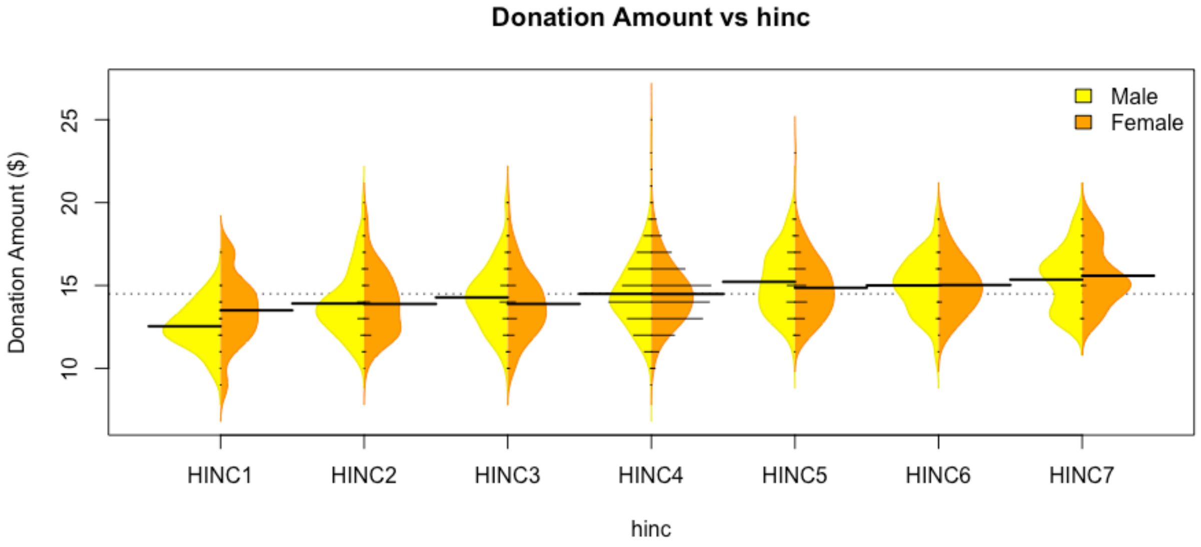
This classification response variable is encoded as a factor variable with two levels: Non-Donor and Donor. Beanplots (Kampstra, 2008) are employed in place of boxplots to investigate the bivariate/multivariate relationships between the response and predictors, due to the richer context they offer. Donor status as a function of average family income shows only a slight increase in mean average income for donors. The distribution of `inca` is fairly normal. In contrast, the percentage of low income households in the neighborhood is significantly lower for donors and seems to be a good predictor. `plow` is very right skewed, but both predictors have long tails. This will be addressed in the data preparation stage.





DAMT Response Variable

A similar analysis on the `damt` response variable is conducted. An example of a strong predictor for the donation amount is household income, as shown below. Mean donation amount is monotonic with income, and isn't significantly different between Males and Females. Clearly, HINC4 has the largest amount of data (as seen by the length of the thin black beanlines). Another observation is that the beanlines are equi-spaced at whole numbers indicating that either donations always come in as whole numbers, or some preprocessing was performed to round the donation to the nearest whole number. Barring a few points in HINC4, this variable doesn't suffer from long tails or massive outliers.



Data Preparation

Transformations

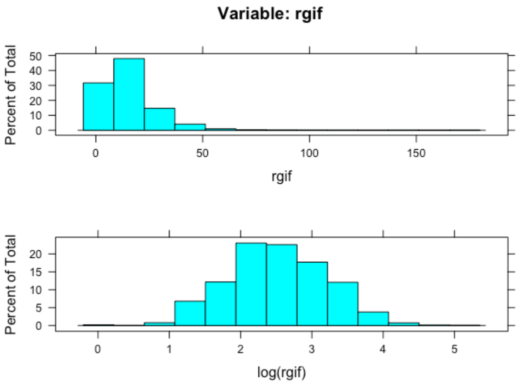
The data for the analysis is clean : it has missing values, and thus needs no imputation. But, the data does benefit from transformations. The skewness for the continuous variables is as follows:

avhv	incm	inca	plow	npro	tgif	lgif	rgif	tdon	tlag	agif
1.57	2.07	1.92	1.42	0.17	2.66	8.33	3.08	1.04	2.61	1.67

Apart from `npro`, all the variables have skewness which can benefit from a transformation. One method of adjusting for the skewness is to run BoxCox transformations on the variables to select the right value of lamda for each transformation. Here are the lambda values selected using the `preProcess(…, method = 'BoxCox')` function in `caret` (for lambda = 0 ± 0.2, a log transform is used):

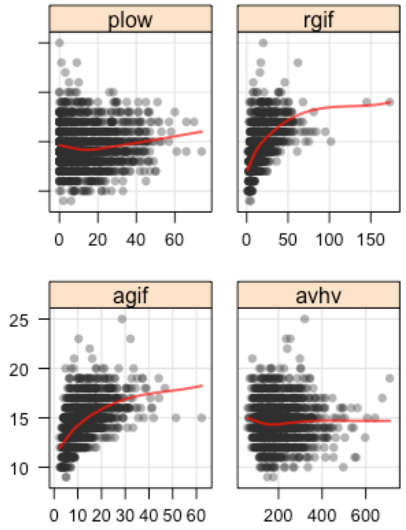
avhv	incm	inca	plow	npro	tgif	lgif	rgif	tdon	tlag	agif
-0.1	0	-0.1	-	0.6	-0.3	-0.2	0	0.1	-0.4	0

To the right is an example of a transformed variable `rgif`, for which a log transform was applied. The transformed variable shows a much more normal distribution.

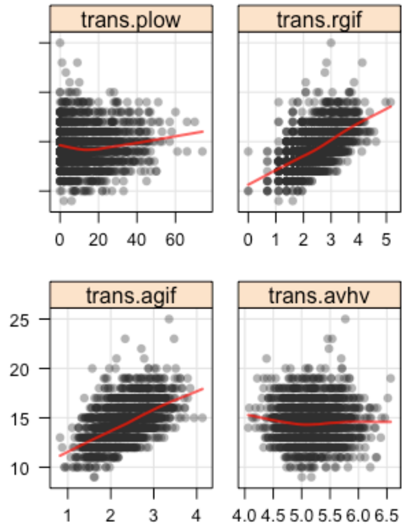


A comparison between the scatter plots for a few predictors show the improvement due to transformations:

Before Transformation



After Transformation



Data Sets Created for Training & Validation Data

Most models are run on many datasets, as defined in the following table. Each data set is prepared with a different approach. Not all models were run on every dataset due to the organic nature of modeling – learn as you go, and apply the learnings to future models.

Data Set Name	Categorical Variables	No change	BoxCox Transformations	Log Transformations	Squared Terms	Standardization
Grouped Raw	Factor	X				X
Ungrouped Raw	Dummy	X				X
Ungrouped Transformed	Dummy		X			X
Ungrouped Transformed Sq	Dummy		X		X	X
Grouped Transformed	Factor		X			X
Grouped Transformed Sq	Factor		X		X	X
Ungrouped Log	Dummy			X		X
Grouped Log	Factor			X		X

The difference between the `Grouped` and `Ungrouped` variables is the way categorical variables are treated. In the Grouped data, a categorical variable with *n* factors is defined as one factor-variable, which means that there is one column defined for the category. On the other hand, in Ungrouped data, the *n* factor categorical variable is coded as *(n-1)* dummy variables.

Analyses like linear or logistic regression are not affected by the grouping vs ungrouping, since they rely on dummy variables only. But, tree-based analyses like random forests, or boosting can vary between the two styles of data. This is investigated in the analysis documented below.

Modeling Approach

In all the models built for the project, the `caret` package has been used. This package consists of wrapper functions which can call ~233 models from R packages. The wrapper functions are an efficient way to:

1. Apply any data preprocessing like centering, scaling, BoxCox transformations
2. Apply validation procedures like repeated-train-test on bootstrapped samples using standard bootstrap, bootstrap632, repeated cross-validation or LOOCV.
3. Apply automated grid searches for tuning parameters for each model. For example: optimal number of components in a PCR analysis

Performance of a 25 repetition bootstrap approach is compared against 10-fold Cross Validation on numerous models (for tuning parameter selection, and RMSE estimation), and no statistically significant different is found. Unless specified otherwise, all models are run with a 25 repetition bootstrap as the resampling method to identify model parameter estimates or tuning parameters.

A total of 12 modeling approaches are investigated in the project, summarized as follows:

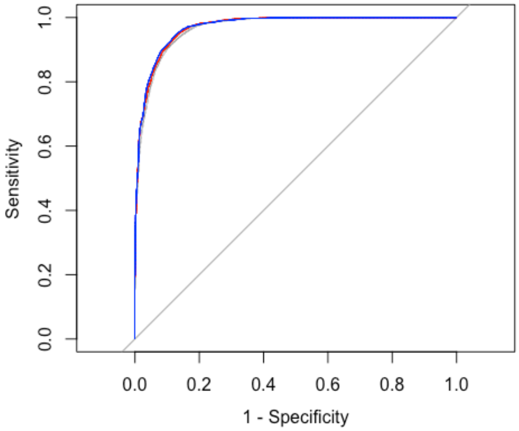
Model	donr	damt
Logit	X	
LDA	X	
QDA	X	
kNN	X	
Bagging	X	X
Boosting	X	X
RF	X	X
PCR		X
PLS		X
Elastic Net	X	X
Neural Net		X
GAM		X

Modeling – `donr` Response

The objective of this section is to develop a classification model to predict if a recipient of the marketing campaign would make a donation. The response variable is `donr`. A total of 8 modeling approaches spanning 28 models are presented below.

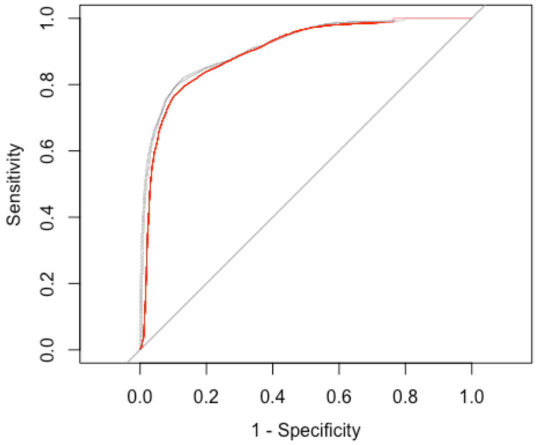
Logit

A total of 4 logit models are fit to the data using 5-repeated 10-fold cross validation on the training data sets. For each model, the variables which are statistically insignificant at the 0.05 level as determined of a z-score are eliminated to improve the model AIC (ranging from 1817.7 to 1689.5) and ROC AUC values (ranging from 0.9671 to 0.9714). The four models have very similar ROC curves as shown here.



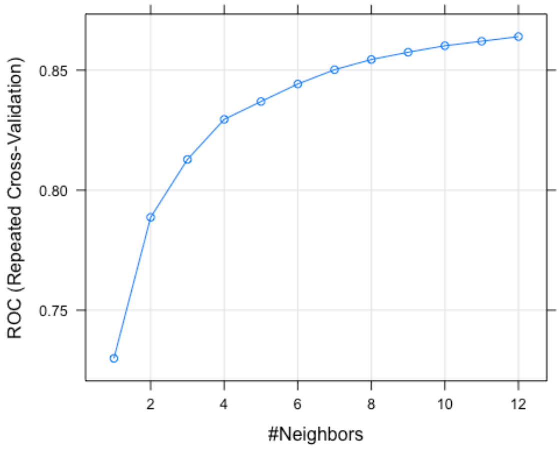
LDA, QDA

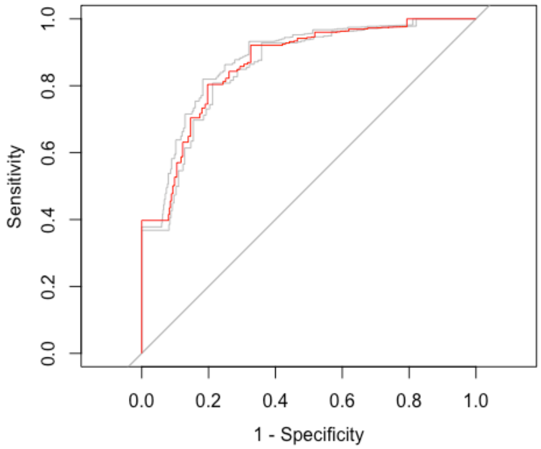
Linear and Quadratic Discriminant Analyses are run for the various data sets using 5-repeated 10-fold cross-validation on the training datasets. LDA fares better with AUC values of around 0.9636, while QDA underperforms with AUC values of around 0.9113. ROC curves for QDA also show the underperformance of the model.



kNN

kNN models do not lend themselves to good models with large number of predictors due to the curse of dimensionality. Nevertheless, the model is used to test their performance. The result is an ROC AUC of 0.8279 to 0.8614, which already underperforms both Logit and LDA models. 10-fold cross validation is used to select the number of neighbors for tuning parameter *k* in the `knn` model.

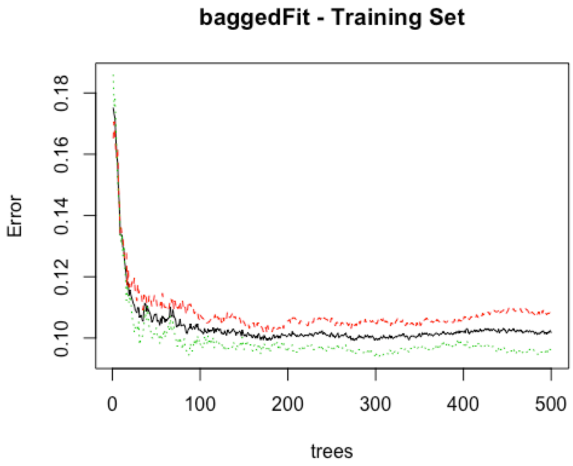


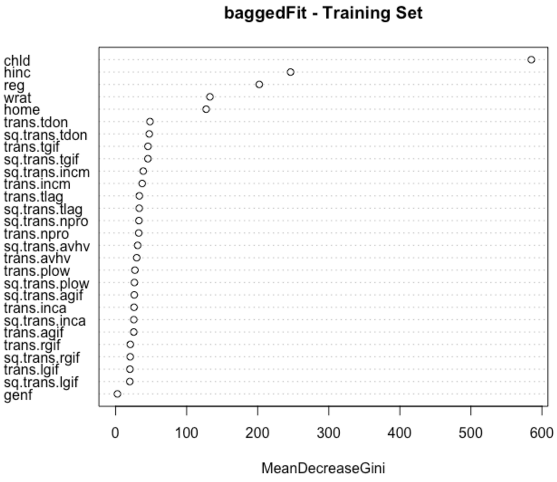


Bagging

Bagging models are run using the `rf` with `mtry` set to the number of variables in the dataset. 500 trees are run on 5-repeat 10-fold cross validation resamples to calculate statistics of the ROC curve. Bagging models are built on grouped and ungrouped datasets.

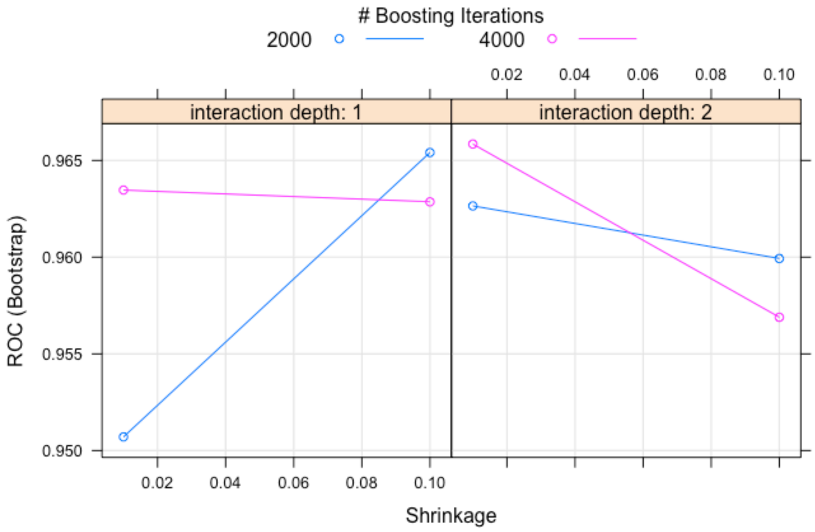
Convergence plot on the left shows that 500 trees have stabilized the error. Variable importance plots indicate `chld`, `hinc`, `reg`, `wrat` and `home` are the highest contributors to the model. The AUC values for the bagging models range from 0.9383 to 0.9571.





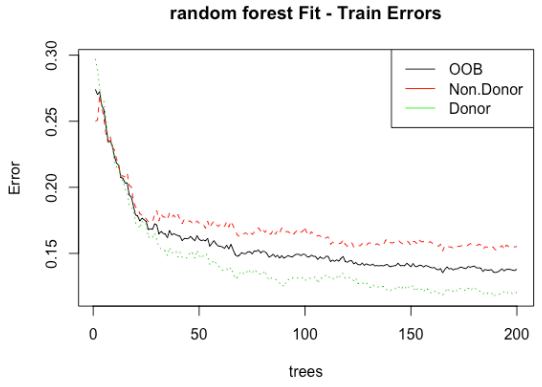
Boosting

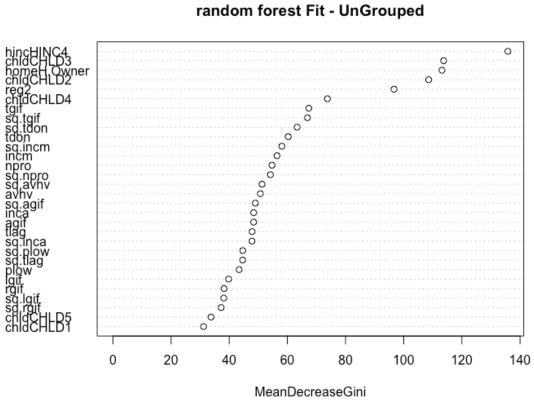
Boosting models have three main parameters to optimize using cross validation methods – number of trees per model, the interaction depth, and the shrinkage. A grid search was performed on trees from 2000 to 4000, interaction depth between 1 and 2, and shrinkage between 0.1 and 0.01. the parameters with the highest ROC AUC are selected. The final values used for the model are n.trees = 4000, interaction.depth = 2, and shrinkage = 0.01. Boosting models are built on grouped and ungrouped datasets.



Random Forest

Random forest implementations use the `rf` method by selecting the `mtry` argument (number of variables randomly sampled as candidates at each split) to be < n. The value of `mtry` is set at sqrt(n), but the optimal number can be selected using cross-validation. For the dataset 'UnGrouped Transformed', the performance of the fit by number of trees is shown to the left. 200 trees seems adequate as the errors stabilize. On the right, the variable importance plot shows that home income, # of children, home ownership, and region are important predictors.

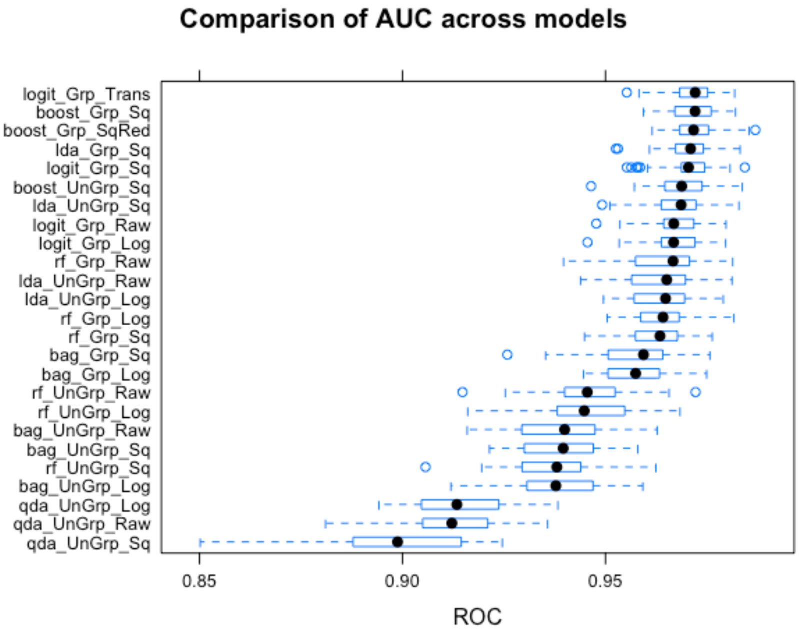




Evaluation – `donr`

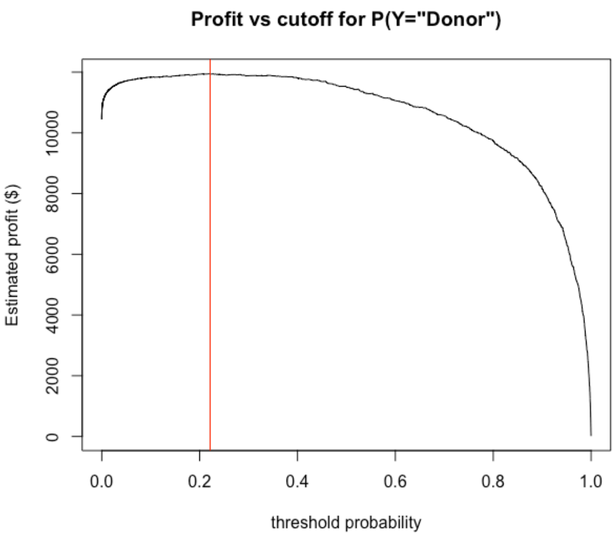
Training Data Comparison

All training models for this `donr`. The following chart is the performance of the `donr` models (ROC AUC) on the training data. The boxplots are calculated off the resampling strategy of 5-repeat 10-fold cross validation. The top three performing models are Logit model (on Grouped Transformed data), Boosted model (on Grouped data with Squared variables) and Boosted model (on Grouped data with Squared variables with reduced variable counts). The performance of the first four models is statistically no different from one other, as checked by ANOVA tests using the `diff(resamples)` command structure in `caret`.



Profit Maximization Calculation

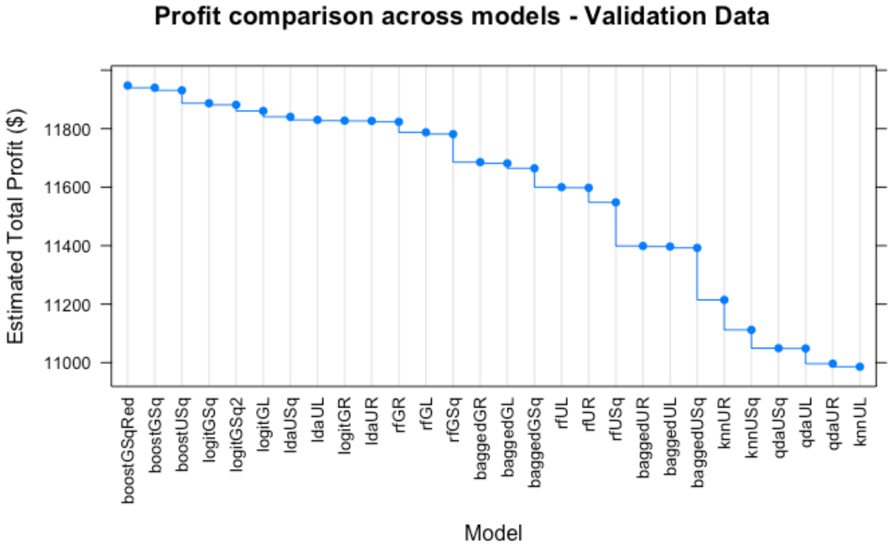
The default probability of cutoff to classify model responses as ‘Donor’ vs ‘Non-Donor’ is 0.5. Changing this value affects the specificity and sensitivity of the model. The objective of this project is to maximize profit, and thus altering the value allows us to optimize the model for profit maximization. For example, the following plot is representative of most models. As threshold probability is reduced, a larger number of ‘Non-Donor’ classifications are now classified as ‘Donor’. While this will impact the specificity, it has the value of also increasing profit. The average expected profit



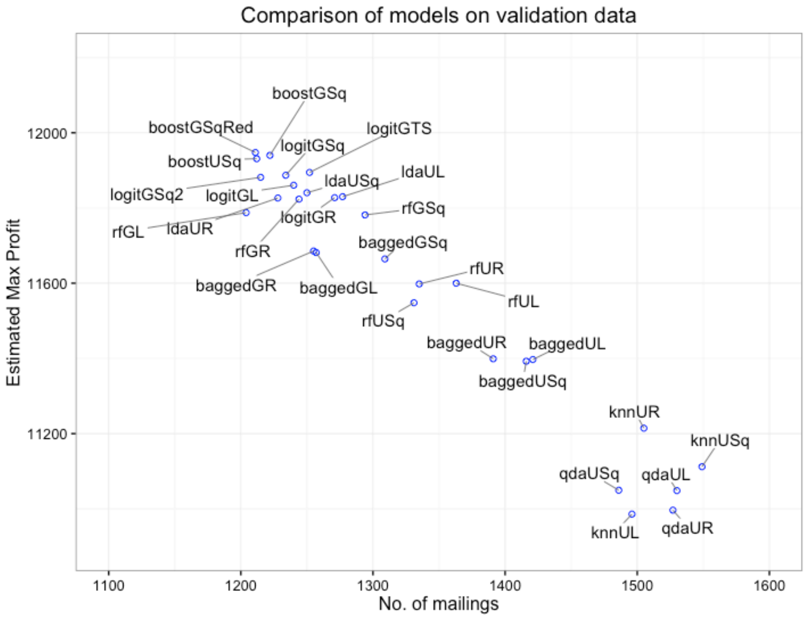
of $12.5 is used for any observations classified 'Donor'.

An optimization routine select the cutoff-P for each model. In the graph above, profit is maximum for cutoff-P = 0.215.

Using this technique, the expected cumulative profit is plotted for each model. The top three models are all Boosted models with different input datasets, followed by logit models.



For the estimated profit to the number of mailings plotted for each model, we can see that the QDA and KNN models performed the worst (lower right corner). Bagged & random forest models occupy the center region. The best models on the top left are logit, LDA and boosted models. What's interesting in this plot is the best performing models maximize profitability while reducing the number of mailings required to do so.

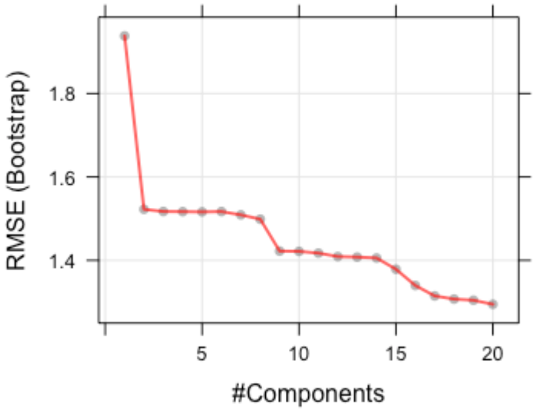


Modeling Part II – `damt` Response

The objective of this section is to develop a predictive model of the amount of donations expected if a recipient of the marketing campaign does wish to donate. The response variable is `damt`, which is the donation amount in USD. A total of 8 modeling approaches spanning 18 models is presented below.

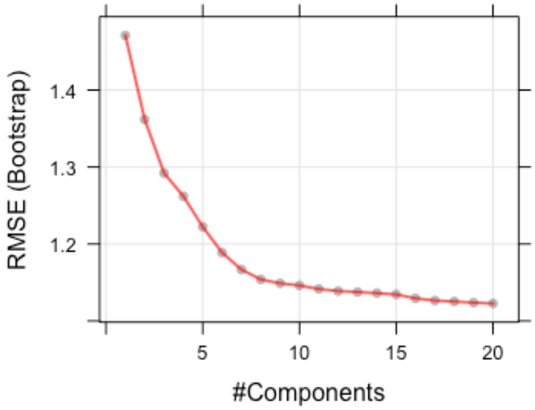
Principal Components Regression

PCR was run over grouped and ungrouped transformed data. A tuning grid of 1 to 20 principal components was evaluated using a repeated bootstrap method. RMSE monotonically decresed with # components as shown here. Number of components selected in the final model is 20.



Partial Least Squares

PLS was run over grouped and ungrouped transformed data. A tuning grid of 1 to 20 principal components was evaluated using a repeated bootstrap method. RMSE monotonically decresed with # components as shown here. Number of components selected in the final model is 20.

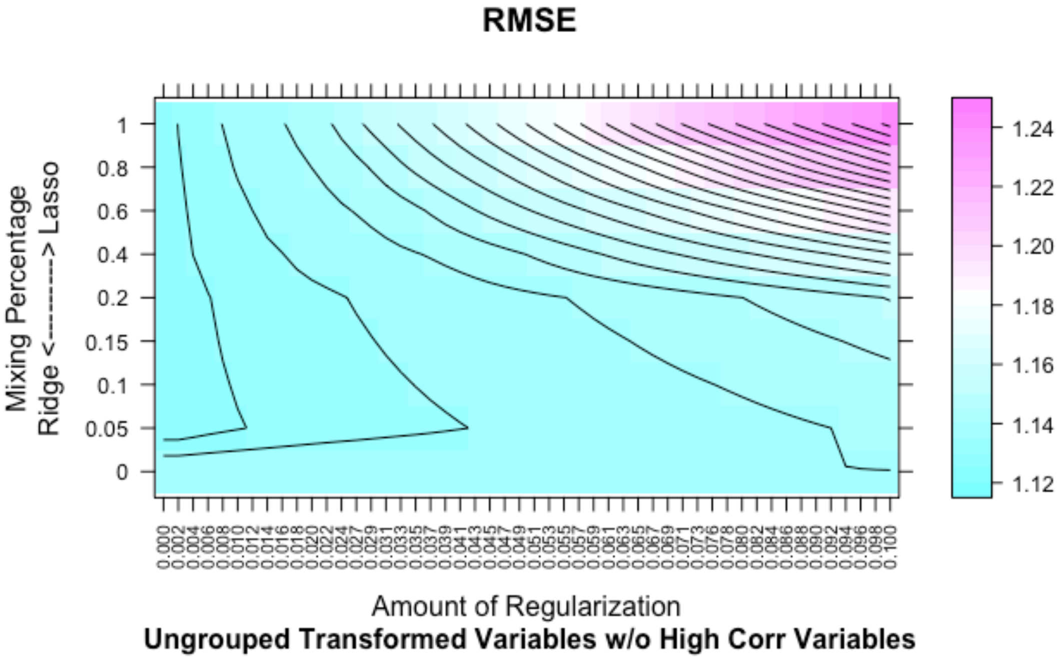


Elastic Net

The elastic net is a regularized regression method that linearly combines the L1 and L2 penalties of the lasso and ridge methods. `caret` enables a grid search over a range of values of alpha – the mixing percentage of L1 and L2 penalties, and lambda – the amount of regularization. Alpha = 1 is a lasso model, and alpha = 0 is a ridge regression model. Any other value of alpha is an elastic-net model.

25 bootstrapped resampling strategy was used to build 200 models to determine the optimal parameters using the `glmnet` package and running parameters `alpha` and `lambda`.

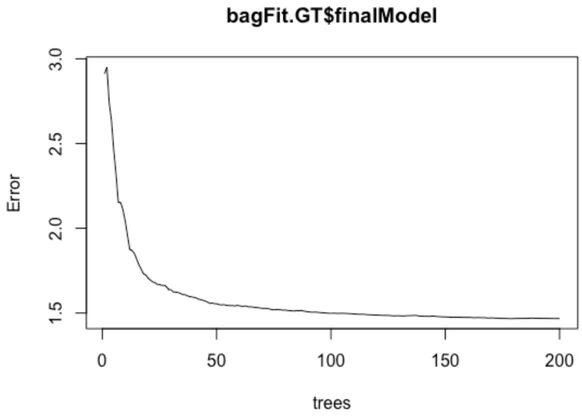
RMSE was used to select the optimal model using the smallest value. The final values used for the model were alpha = 0.2 and lambda = 1e-05. The low value of lambda indicates that the method could not identify any variables worth reducing to zero, thus not having selected the lasso method. The contour plot below shows how RMSE changes over the search grid of the two tuning parameters.

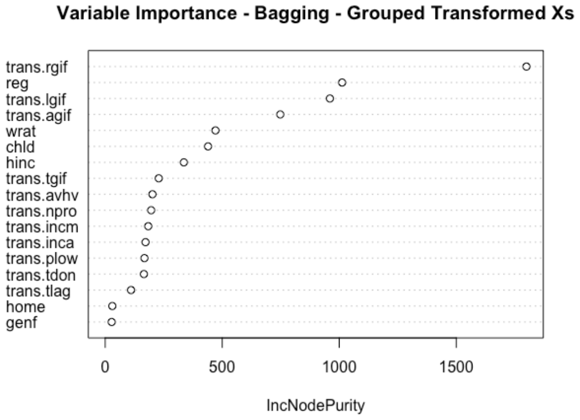


Bagging

Bagging models are run using the `rf` with `mtry` set to the number of variables in the dataset. 200 trees are run on repeated bootstrapped samples for a total of 25 runs to calculate the RMSE. Bagging models are built on grouped and ungrouped datasets.

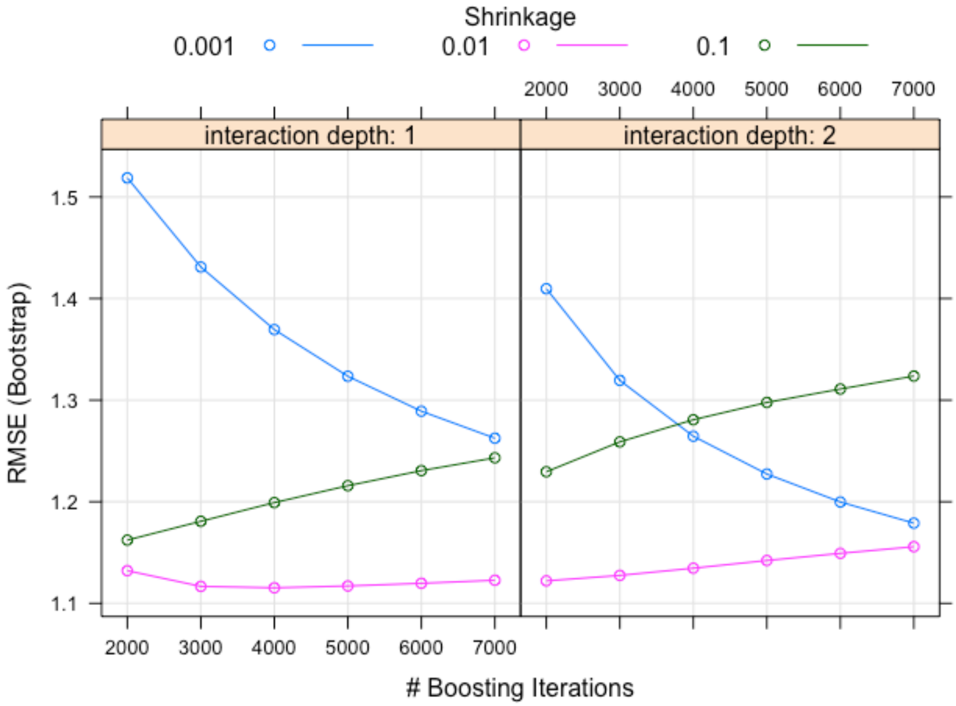
Convergence plot on the left shows that 200 trees have stabilized the error. Variable importance plots indicate `rgif`, `reg`, `lgif` and `agif` are the highest contributors to the model.



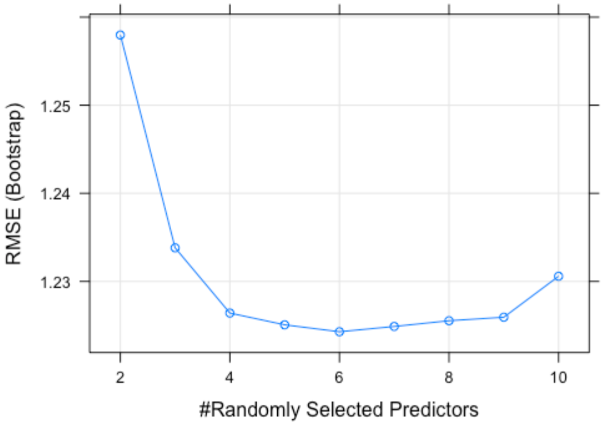


Boosting

Boosting models have three main parameters to optimize using cross validation methods – number of trees per model, the interaction depth, and the shrinkage. A grid search was performed on trees from 2000 to 7000, interaction depth between 1 and 2, and shrinkage between 0.1, 0.01 and 0.001. the parameters with the lowest RMSE are selected. The final values used for the model were n.trees = 4000, interaction.depth = 1, and shrinkage = 0.01. Boosting models are built on grouped and ungrouped datasets.



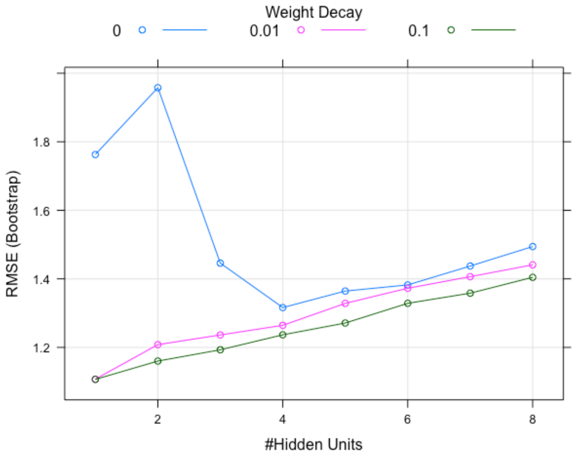
Random Forest



Random forest implementations use the `rf` method by selecting the `mtry` argument (number of variables randomly sampled as candidates at each split) to be < n. Typically, it is set at sqrt(n), but the optimal number can be selected using cross-validation. The plot on the right shows that `mtry = 6` is the selected value that minimizes RMSE. Random forest models are built on grouped and ungrouped datasets.

Neural Net

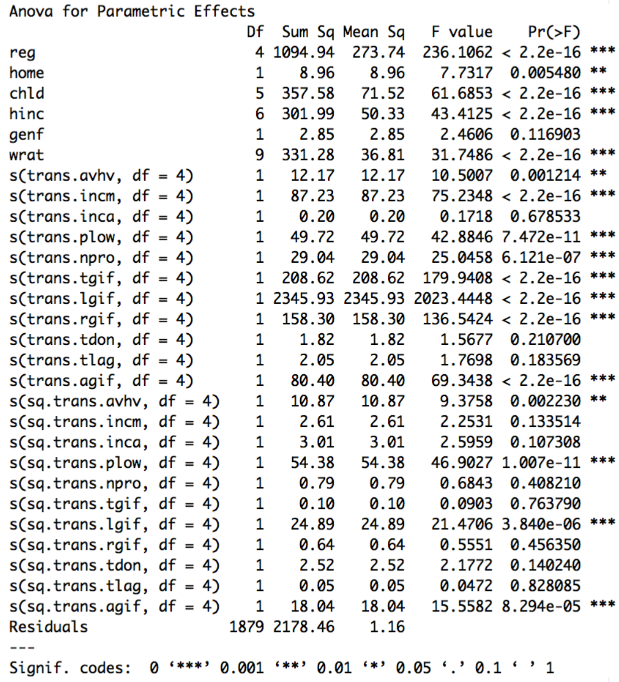
Neural net models have two main parameters to optimize using cross validation methods – weight decay, and number of hidden units. A grid search was performed on decay of 0, 0.1 and 0.01, and size from 1 to 8. The final values used for the model were size = 1 and decay = 0.1. The graph on the right shows the results of the cross-validation study.



Generalized Additive Models

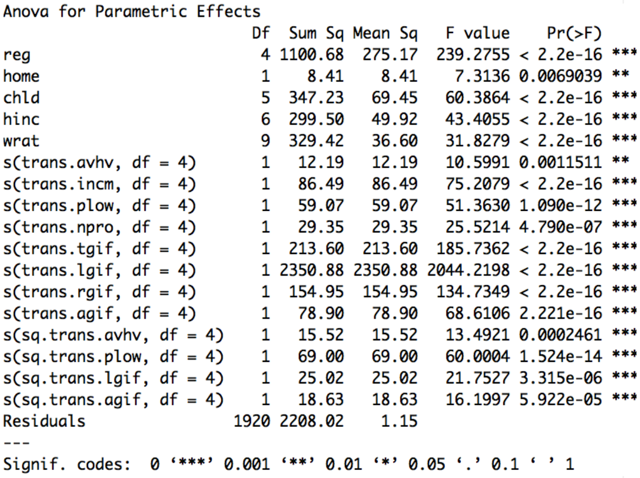
The `gam` package is used to fit GAM models. Although all the models used so far have been using `caret`, the constraint with using `caret` and `gam` is that any selected technique like smoothing splines or LOESS are applied to all variables including categorical variables. As seen in the Evaluation section below, this doesn't lend itself to a model which is competitive with the other models built using `caret`.

GAM Customized



```
Anova for Parametric Effects
                           Df  Sum Sq Mean Sq   F value    Pr(>F)    
reg                         4 1094.94  273.74  236.1062 < 2.2e-16 ***
home                        1    8.96    8.96    7.7317  0.005480 ** 
chld                        5  357.58   71.52   61.6853 < 2.2e-16 ***
hinc                        6  301.99   50.33   43.4125 < 2.2e-16 ***
genf                        1    2.85    2.85    2.4606  0.116903    
wrat                        9  331.28   36.81   31.7486 < 2.2e-16 ***
s(trans.avhv, df = 4)       1   12.17   12.17   10.5007  0.001214 ** 
s(trans.incm, df = 4)       1   87.23   87.23   75.2348 < 2.2e-16 ***
s(trans.inca, df = 4)       1    0.20    0.20    0.1718  0.678533    
s(trans.plow, df = 4)       1   49.72   49.72   42.8846 7.472e-11 ***
s(trans.npro, df = 4)       1   29.04   29.04   25.0458 6.121e-07 ***
s(trans.tgif, df = 4)       1  208.62  208.62  179.9408 < 2.2e-16 ***
s(trans.lgif, df = 4)       1 2345.93 2345.93 2023.4448 < 2.2e-16 ***
s(trans.rgif, df = 4)       1  158.30  158.30  136.5424 < 2.2e-16 ***
s(trans.tdon, df = 4)       1    1.82    1.82    1.5677  0.210700    
s(trans.tlag, df = 4)       1    2.05    2.05    1.7698  0.183569    
s(trans.agif, df = 4)       1   80.40   80.40   69.3438 < 2.2e-16 ***
s(sq.trans.avhv, df = 4)    1   10.87   10.87    9.3758  0.002230 ** 
s(sq.trans.incm, df = 4)    1    2.61    2.61    2.2531  0.133514    
s(sq.trans.inca, df = 4)    1    3.01    3.01    2.5959  0.107308    
s(sq.trans.plow, df = 4)    1   54.38   54.38   46.9027 1.007e-11 ***
s(sq.trans.npro, df = 4)    1    0.79    0.79    0.6843  0.408210    
s(sq.trans.tgif, df = 4)    1    0.10    0.10    0.0903  0.763790    
s(sq.trans.lgif, df = 4)    1   24.89   24.89   21.4706 3.840e-06 ***
s(sq.trans.rgif, df = 4)    1    0.64    0.64    0.5551  0.456350    
s(sq.trans.tdon, df = 4)    1    2.52    2.52    2.1772  0.140240    
s(sq.trans.tlag, df = 4)    1    0.05    0.05    0.0472  0.828085    
s(sq.trans.agif, df = 4)    1   18.04   18.04   15.5582 8.294e-05 ***
Residuals                1879 2178.46    1.16                        
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

GAM Customized Reduced



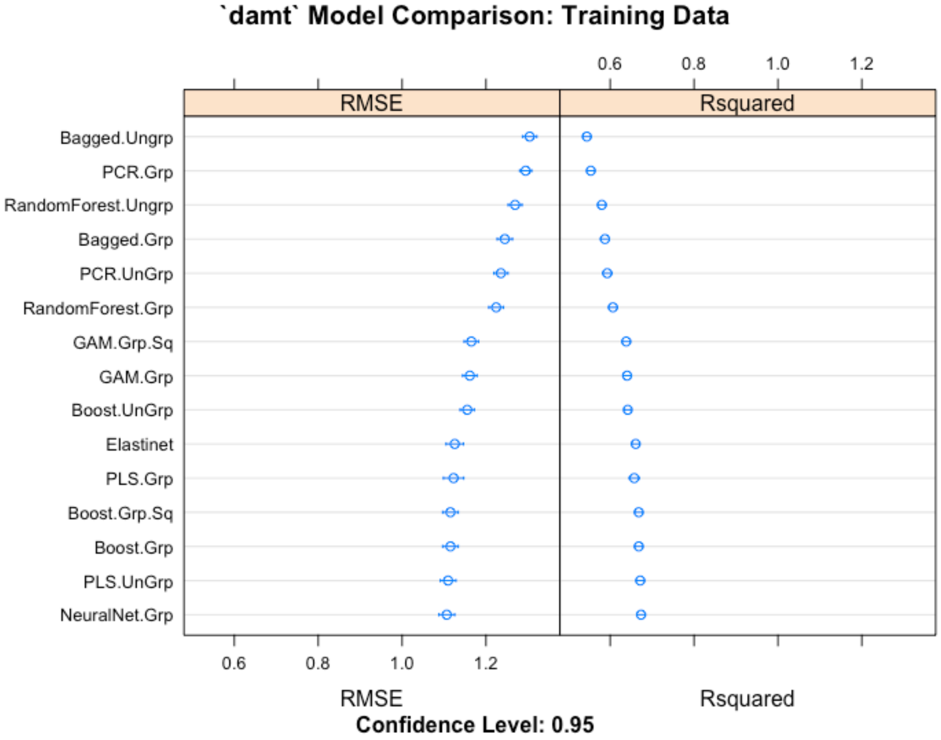
```
Anova for Parametric Effects
                           Df  Sum Sq Mean Sq   F value    Pr(>F)    
reg                         4 1100.68  275.17  239.2755 < 2.2e-16 ***
home                        1    8.41    8.41    7.3136 0.0069039 ** 
chld                        5  347.23   69.45   60.3864 < 2.2e-16 ***
hinc                        6  299.50   49.92   43.4055 < 2.2e-16 ***
wrat                        9  329.42   36.60   31.8279 < 2.2e-16 ***
s(trans.avhv, df = 4)       1   12.19   12.19   10.5991 0.0011511 ** 
s(trans.incm, df = 4)       1   86.49   86.49   75.2079 < 2.2e-16 ***
s(trans.plow, df = 4)       1   59.07   59.07   51.3630 1.090e-12 ***
s(trans.npro, df = 4)       1   29.35   29.35   25.5214 4.790e-07 ***
s(trans.tgif, df = 4)       1  213.60  213.60  185.7362 < 2.2e-16 ***
s(trans.lgif, df = 4)       1 2350.88 2350.88 2044.2198 < 2.2e-16 ***
s(trans.rgif, df = 4)       1  154.95  154.95  134.7349 < 2.2e-16 ***
s(trans.agif, df = 4)       1   78.90   78.90   68.6106 2.221e-16 ***
s(sq.trans.avhv, df = 4)    1   15.52   15.52   13.4921 0.0002461 ***
s(sq.trans.plow, df = 4)    1   69.00   69.00   60.0004 1.524e-14 ***
s(sq.trans.lgif, df = 4)    1   25.02   25.02   21.7527 3.315e-06 ***
s(sq.trans.agif, df = 4)    1   18.63   18.63   16.1997 5.922e-05 ***
Residuals                1920 2208.02    1.15                         
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The solution is to build a GAM model using the conventional `gam()` calls and smoothing splines for each continuous variable. GAM.Cust and GAM.CustRed are the two custom models. GAM.CustRed (AIC 6012) is a reduced model from GAM.Cust (AIC 6067) by removing insignificant variables identified using ANOVA. A comparison is shown in the table above.

Evaluation – `damt`

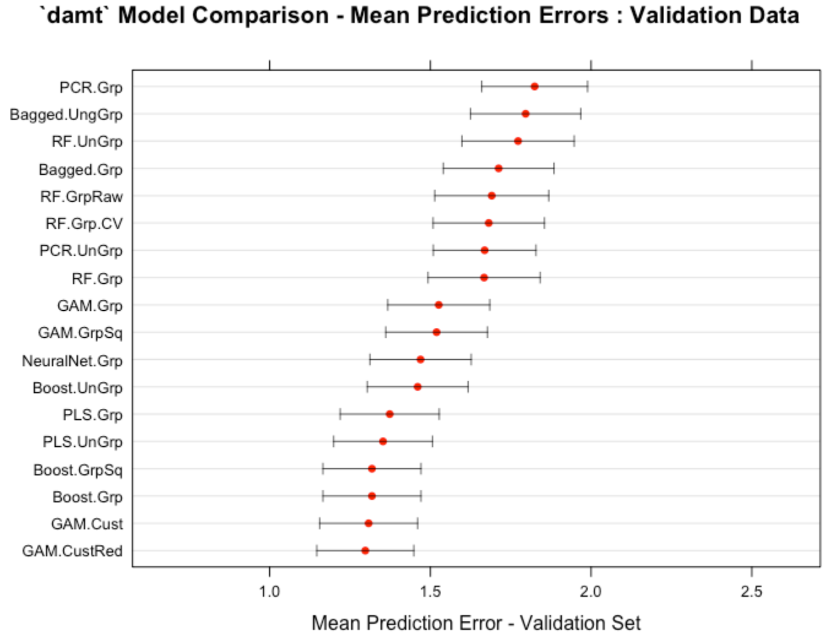
Training Data Comparison

The following chart is the performance of the `damt` models (RMSE) on the training data. Each point is the estimate for the model, and the bars represent the 95% confidence interval determined by the resampling strategy. The top three performing models are PLS (on UnGrouped Transformed data), NeuralNet (on Grouped Transformed data) and Boosted trees (on Grouped Transformed Data). But, the performance of the first six models is statistically no different from one other, as checked by ANOVA tests using the `diff(resamples)` command structure in `caret`. Models above Elastinet get statistically worse the ones below.

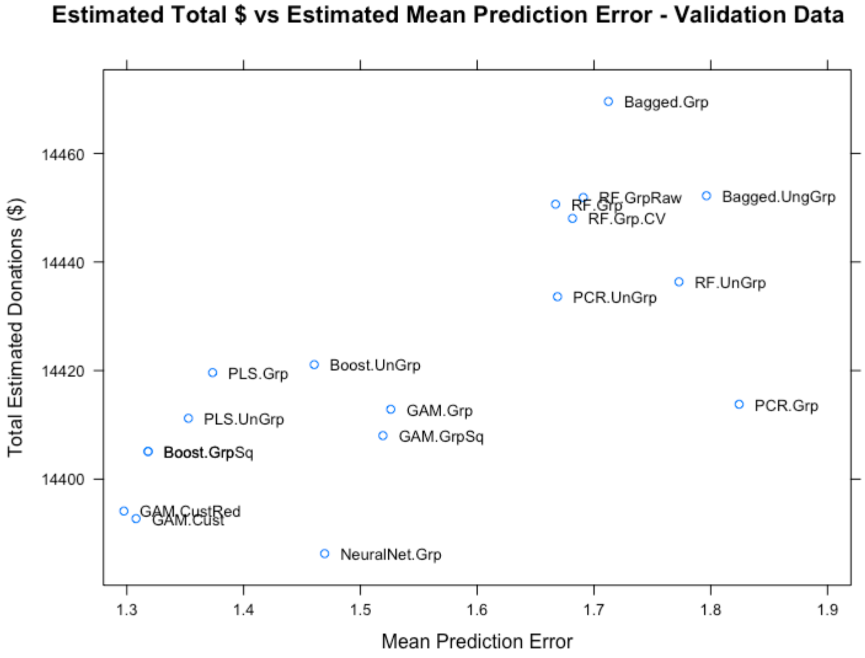


Validation Data Comparison

The models when applied to the validating dataset result in the the Customized GAM models to shoot to the top of the list, followed by the Boosting and PLS models. Although the point estimates show the top four models in a particular, the standard error bars show that all four models are statistically quite the same.


`damt` Model Comparison - Mean Prediction Errors : Validation Data

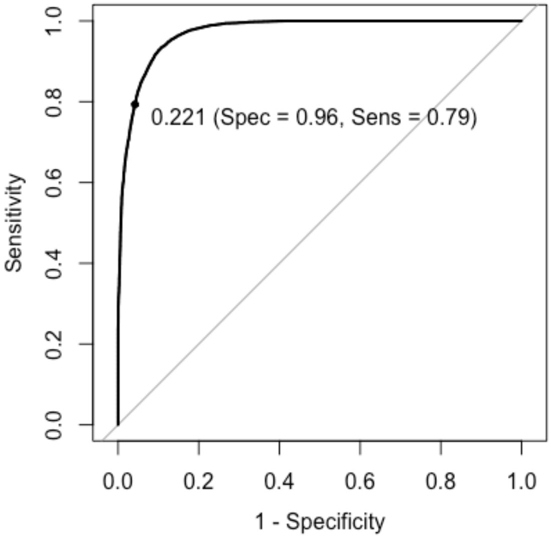
When the total donation estimates are plotted against the mean prediction error, we can see the importance of calculating the prediction errors for the models. Although there are many models that estimate a higher total estimated donation amount (top right of the graph), these are also models with higher prediction errors. The winning models are those in the bottom left corner with the lowest error.


Estimated Total $ vs Estimated Mean Prediction Error - Validation Data

Final Remarks

Final Model Selected for `donr`

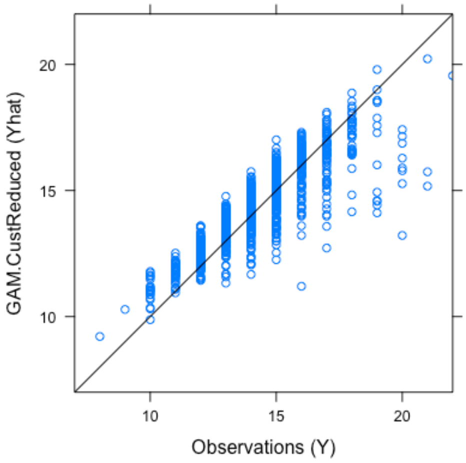
A boosting model gives the maximum expectation of **average profit of $11,947.50** for **1,211 mailings** of the marketing material. The variables to be used in the boosting model chld, hinc, reg, wrat, home, tdon, tgif, incm, tlag, npro, inca, plow, avhv, agif. The boosting model shall be built with the parameters: n.trees = 4000, interaction.depth = 2, and shrinkage = 0.01. This model has an AUC of 0.9722, and uses a cutoff-P value of 0.2.215.

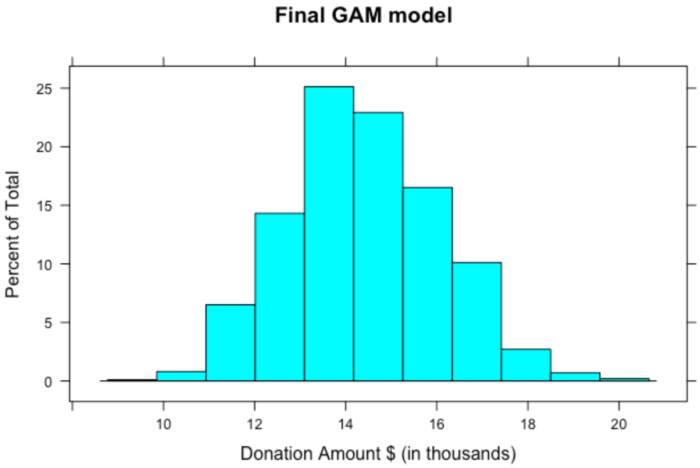


Final Model Selected for `damt`

The reduced GAM model is selected as the final model given that it's performance is as good as the competitor Boosting, NeuralNet & PLS models, but is much more interpretable than the other models.

The **total expected donation amount** predicted by the GAM model is **$14,394**, with a standard error of $1,670.





R Packages Used

nnet_7.3-12,

plyr_1.8.4,

pls_2.5-0,

tree_1.0-37,

gam_1.12,

e1071_1.6-7,

Hmisc_3.17-4,

Formula_1.2-1,

pROC_1.8,

caret_6.0-75,

glmnet_2.0-5,

foreach_1.4.3,

Matrix_1.2-7.1,

MASS_7.3-45,

beanplot_1.2,

gbm_2.1.1,

survival_2.39-5,

randomForest_4.6-12

lattice_0.20-34,

dplyr_0.5.0,

purrr_0.2.2.9000,

readr_0.2.2,

tidyr_0.6.0.9000,

tibble_1.2,

ggplot2_2.1.0,

tidyverse_1.0.0.9000