Course Project

Predict 422 Section 56

Winter 2016

Joshua Peng

Predict 422 Course Project

Joshua Peng

Hyperlinks

[1. Introduction](#_1._Introduction)

[2. Exploring and Preparing Data](#_2._Exploring_and)

[3. Building Classification Models](#_3._Building_Classification)

[4. Selecting the Best Classification Model](#_4._Selecting_the)

[5. Building Prediction Models](#_5._Building_Prediction)

[6. Selecting the Best Prediction Model](#_6._Selecting_the)

[7. Conclusion](#_7._Conclusion)

[Appendix: R Code](#_Appendix:_R_Code)

# 1. Introduction

A charitable organization wishes to develop a machine learning model to improve the costeffectiveness of their direct marketing campaigns to previous donors. According to their recent mailing records, the typical overall response rate is 10%. Out of those who respond (donate) to the mailing, the average donation is $14.50. Each mailing costs $2.00 to produce and send; the mailing includes a gift of personalized address labels and assortment of cards and envelopes. It is not cost-effective to mail everyone because the expected profit from each mailing is 14.50 x 0.10 – 2 = -$0.55. I will develop a classification model using data from the most recent campaign that can effectively capture likely donors so that the expected net profit is maximized.

The entire charity dataset consists of 3984 training observations, 2018 validation observations, and 2007 test observations. Weighted sampling has been used, over-representing the responders so that the training and validation samples have approximately equal numbers of donors and non-donors. The response rate in the test sample has the more typical 10% response rate. I will build a prediction model to predict expected gift amounts from donors. The data for the prediction model will consist of the records for donors only.

# 2. Exploring and Preparing Data

Examining Variable Type

I begin by examining the Data Dictionary provided in the Course Project introduction document to see what kind of variables are included in the data. Based on the quick preview of the data and the description below, I can also determine the variable type. There are 2 response variables: DONR and DAMT, which have several missing observations and an index variable, ID. All of the missing data are in the test sets as indicated by the PART variable which I will have to fill in with predicted values. These 4 non-predictor variables are highlighted in yellow in the table below in order to distinguish them from the other (predictor) variables. When the data is initially loaded in, all of the remaining predictor variables are quantitative variables of “int” type (integer values) except for agif which is “num” (decimal values). The region variables (REG1, REG2, REG3, and REG4), HOME, GENF, DONR are dummy (indicator) variables. Region may have been a 5 class categorical variable which has now been converted to 4 dummy variables with REG5 serving as the reference. In addition to the dummy variables, 2 of the int type variables, can be utilized as categorical variables: HINC and WRAT. I will need to examine the data distribution of these variables to see if they will be better utilized as categorical variables.

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| ID | Index | Index of observations |
| REG1, REG2, REG3, REG4 | Binary Categorical | Region (There are five geographic regions; only four are needed for analysis since if a potential donor falls into none of the four he or she must be in the other region. Inclusion of all five indicator variables would be redundant and cause some modeling techniques to fail. A “1” indicates the potential donor belongs to this region.) |
| HOME | Binary Categorical | (1 = homeowner, 0 = not a homeowner) |
| CHLD | Continuous | Number of children |
| HINC | Continuous/  Categorical | Household income (7 categories) |
| GENF | Binary Categorical | Gender (0 = Male, 1 = Female) |
| WRAT | Continuous/  Categorical | Wealth Rating (Wealth rating uses median family income and population statistics from each area to index relative wealth within each state. The segments are denoted 0-9, with 9 being the highest wealth group and 0 being the lowest.) |
| AVHV | Continuous | Average Home Value in potential donor's neighborhood in $ thousands |
| INCM | Continuous | Median Family Income in potential donor's neighborhood in $ thousands |
| INCA | Continuous | Average Family Income in potential donor's neighborhood in $ thousands |
| PLOW | Continuous | Percent categorized as “low income” in potential donor's neighborhood |
| NPRO | Continuous | Lifetime number of promotions received to date |
| TGIF | Continuous | Dollar amount of lifetime gifts to date |
| LGIF | Continuous | Dollar amount of largest gift to date |
| RGIF | Continuous | Dollar amount of most recent gift |
| TDON | Continuous | Number of months since last donation |
| TLAG | Continuous | Number of months between first and second gift |
| AGIF | Continuous | Average dollar amount of gifts to date |
| DONR | Binary Categorical | Classification Response Variable (1 = Donor, 0 = Non-donor) |
| DAMT | Continuous | Response Variable: Donation Amount in dollars |
| PART | Categorical | Indicates whether the observation is part of the training (train), test (test), or validation (valid) set |

Examining Variables

In this next section, I will examine the histograms of the continuous variables and the frequency tables of the categorical variables.

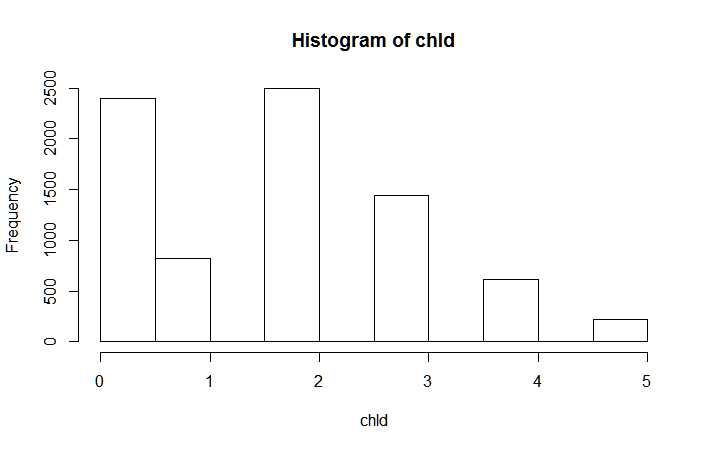
REGx (all 5 Region variables)

By taking the sum of each REGx variable, I find the instances for each of the 5 geographical regions (displayed in the following table). I also find the correlation coefficient of each REGx variable with DONR in the training set. Region 2 (REG2) has the most instances and the highest correlation with DONR.

|  |  |  |
| --- | --- | --- |
| Region | # Instances | Correlation with DONR |
| Region 1 | 1605 | 0.05645603 |
| Region 2 | 2555 | 0.2470784 |
| Region 3 | 1071 | -0.1043229 |
| Region 4 | 1117 | -0.1262810 |
| Region 5 | 1661 | -0.1548909 |

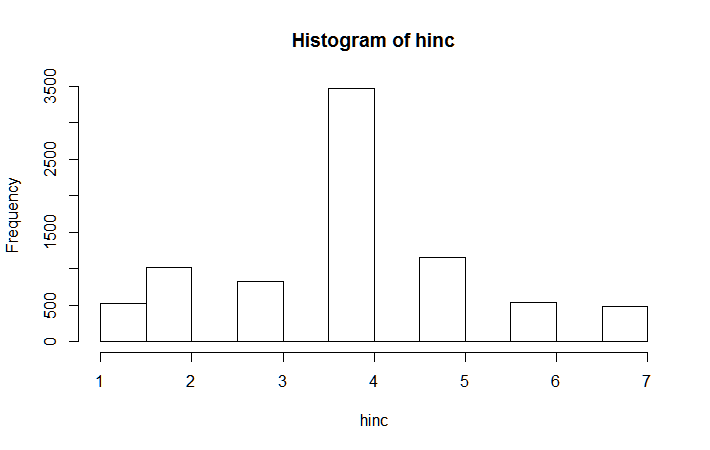
CHLD

The CHLD histogram is zero inflated. The highest count is for 2 children followed by 0 children. Even without the zero inflation, the histogram would appear positively skewed with a long right tail.



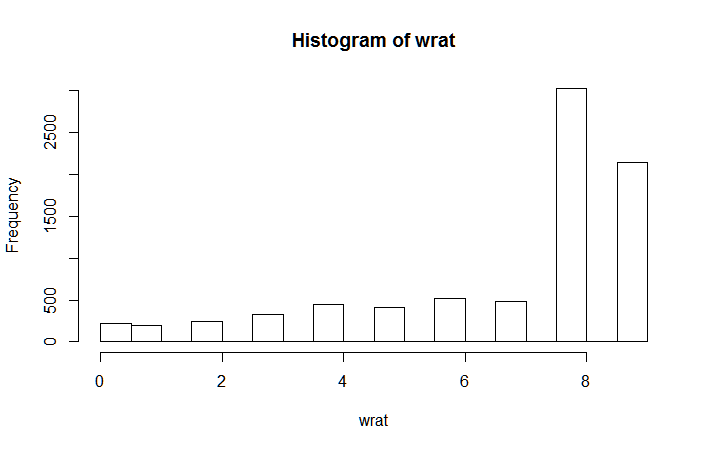
HINC

The HINC histogram appears evenly distributed except with a strong singular central spike at 4 which is also its median and mode. This spike makes the distribution highly kurtotic. The Anderson-Darling test for normality p-value < 2.2e16 meaning that the HINC histogram does not resemble a normal distribution.



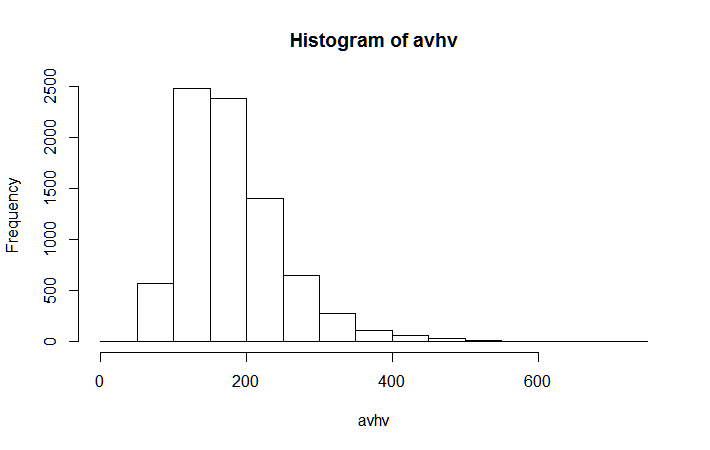
WRAT

The WRAT histogram is extremely negatively skewed. The distribution gradually increasing from 1 to 7 but then jumps up at 8 and drops slightly at 9. The highest frequency wealth group is 8 which is also its median and mode.



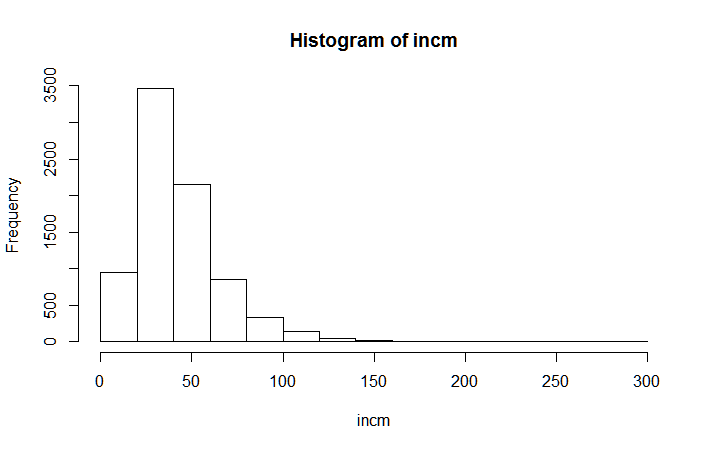
AVHV

The AVHV histogram is positively skewed with a long right tail. The median is 169 and the mean is 182.6, which are both close to the highest bin with the peak.



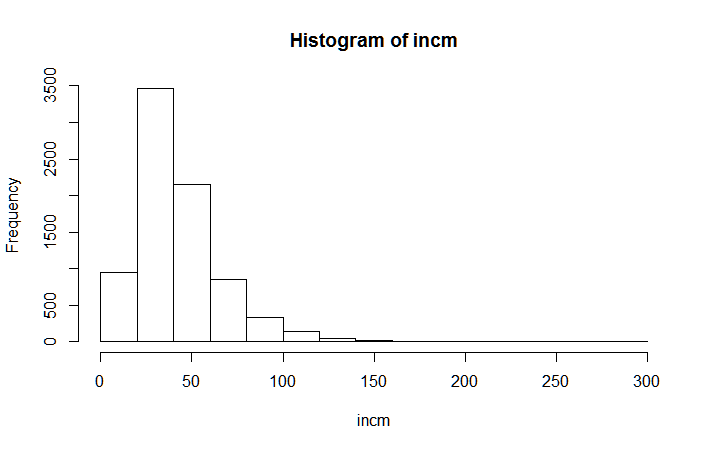
INCM

The INCM histogram is positively skewed with a long right tail. The median is 38 and the mean is 43.47, which are contained in the highest bin with the peak.



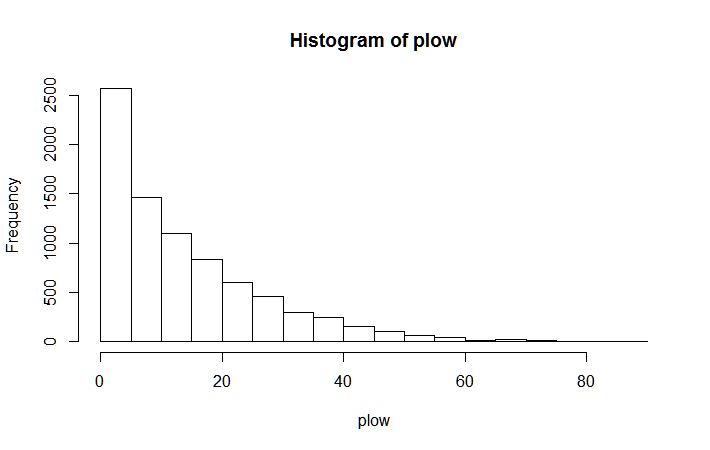
INCA

The INCA histogram is positively skewed with a long right tail. The median is 169 and the mean is 182.6, which are both contained in the highest bin with the peak.



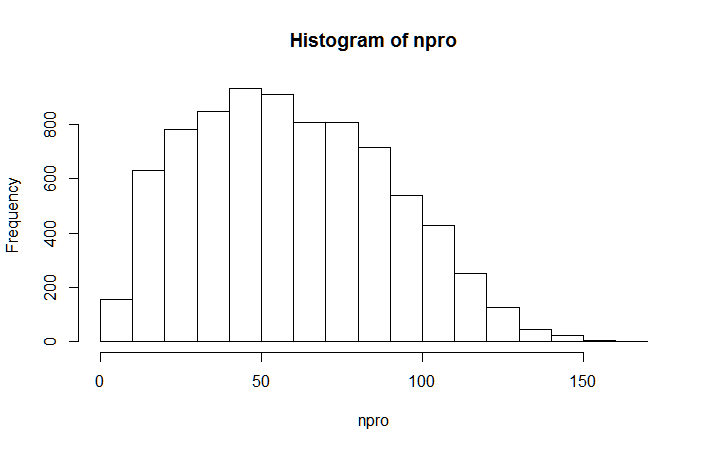
PLOW

The PLOW histogram is shaped like a poisson distribution that decays from left to right. The histogram is zero inflated with a long right tail extending out to a maximum of 87.



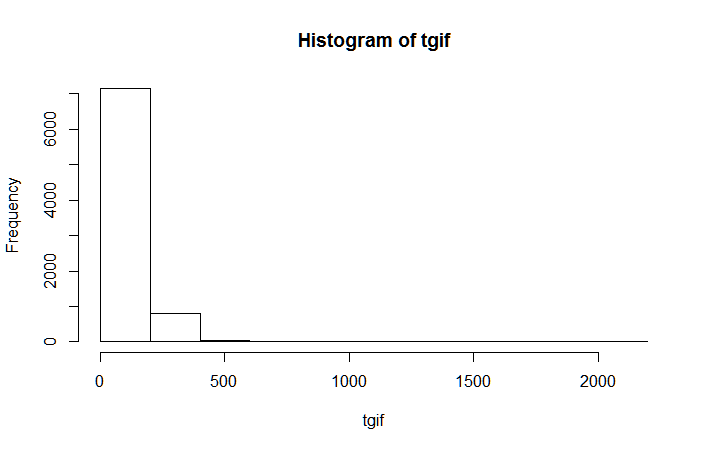
NPRO

The NPRO histogram sharply increases from its first to second bin then after the peak gradually descends with a longer right tail. The Anderson-Darling test for normality p-value < 2.2e16 meaning that the HINC histogram does not resemble a normal distribution.



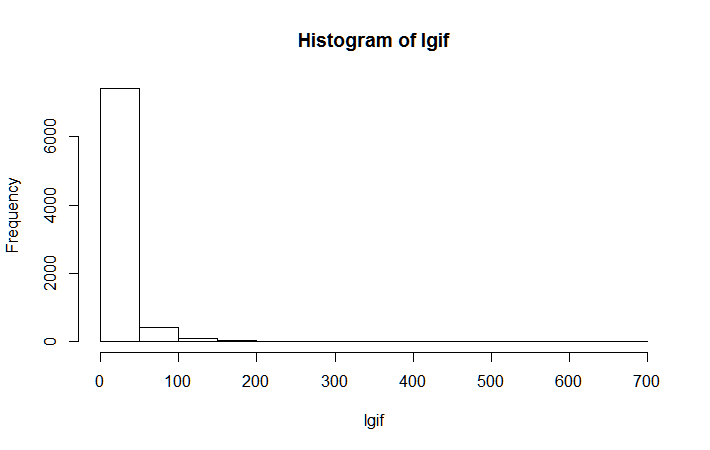
TGIF

The TGIF histogram is shaped like a poisson distribution that decays from left to right. The histogram has a long right tail extending out to a maximum of 2057.



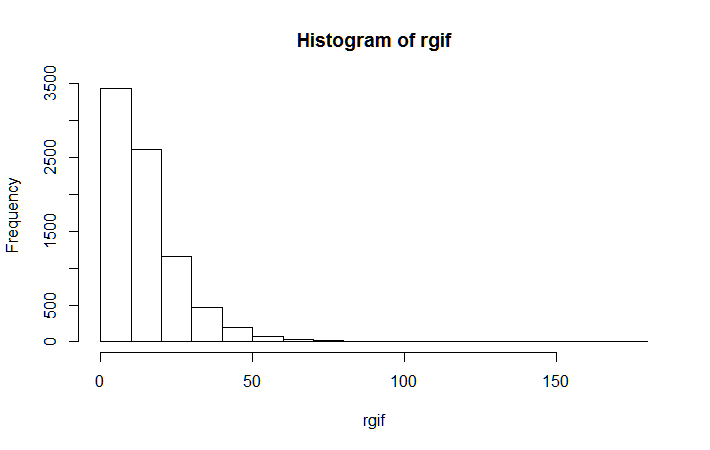
LGIF

The LGIF histogram is shaped like a poisson distribution that decays from left to right. The histogram has a long right tail extending out to a maximum of 681.



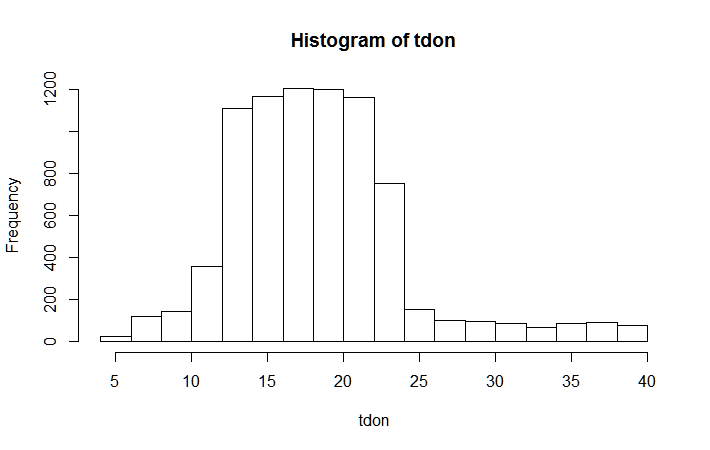
RGIF

The RGIF histogram is shaped like a poisson distribution that decays from left to right. The histogram has a long right tail extending out to a maximum of 173.



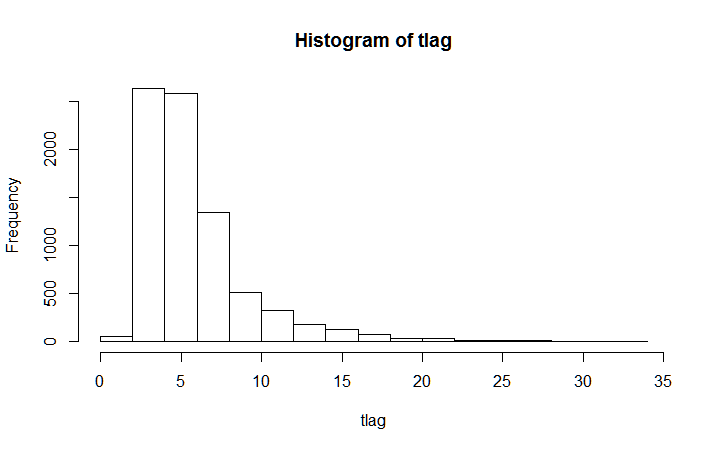
TDON

The TDON histogram is sharply jump up at 12 and stays relatively flat between 13 to 24, and sharply drops at 22. The median is 18 and the mean is 18.86 and both are contained on the top of plateau. The right tail extends out to 40. TDON may the variable closest to resembling a normal distribution, however the Anderson-Darling test for normality p-value < 2.2e16 meaning that the HINC histogram does not resemble a normal distribution.



TLAG

The TLAG histogram sharply jumps up at 4 and then resembles a poisson distribution that decays with a long right tail that extends out to 34.



AGIF

The AGIF histogram sharply jumps up at 5 (the mode is 5.31) and then resembles a poisson distribution that decays with a long right tail that extends out to 72.27.



Adding New Variables

There were 11 continuous variable histograms that were positively skewed (AVHV, INCM, INCA, PLOW, NPRO, TGIF, LGIF, RGIF, TDON, TLAG, and AGIF) and 1 continuous variable histogram that was negatively skewed (WRAT). The square root and natural logarithm transforms can help fix the positively skewed distributions to appear more normal, while the square (to power of 2) or cube (to the power of 3) transforms can adjust negatively skewed distributions to appear more normal. Data points zero or less are undefined when I take the natural logarithm, so I set these values to zero for the natural logarithm transformed variables. Since CHLD was zero inflated, I create CHLD0 as an indicator variable set to 1 when CHLD0 = 0 and set to 0 otherwise. I know that REG5 is an implied reference point for all of the REGx (region) variables, but in order to actually include it into as an automated variable selection input variable, I need to explicitly add the variable. The following table indicates all of the new variables that I added to the charity data set.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Natural Logarithm  Transform | Square Root  Transform | Square  Transform | Cube  Transform | Indicator Variables |
| LN\_CHLD | SR\_CHLD | HINC2 | HINC3 | CHLD0 |
| LN\_AVHV | SR\_AVHV | WRAT2 | WRAT3 | REG5 |
| LN\_INCM | SR\_INCM |  |  |  |
| LN\_INCA | SR\_INCA |  |  |  |
| LN\_PLOW | SR\_PLOW |  |  |  |
| LN\_NPRO | SR\_NPRO |  |  |  |
| LN\_TGIF | SR\_TGIF |  |  |  |
| LN\_LGIF | SR\_LGIF |  |  |  |
| LN\_RGIF | SR\_RGIF |  |  |  |
| LN\_TDON | SR\_TDON |  |  |  |
| LN\_TLAG | SR\_TLAG |  |  |  |
| LN\_AGIF | SR\_AGIF |  |  |  |

Preparing Data

Prior to building models, I have to choose between standardizing the data and then splitting up the data into training, test, and validation sets or splitting up the data and then standardizing the data. There are disadvantages to both sequences. When you perform standardization after splitting the data set, you are additionally testing the separation of the means and variances of the variables of each subset. When you perform standardization before before splitting the data set, you will be polluting your validation and test samples by incorporating information from the means and variances of the training set. For this project, I will generate a set of models for both sequences.

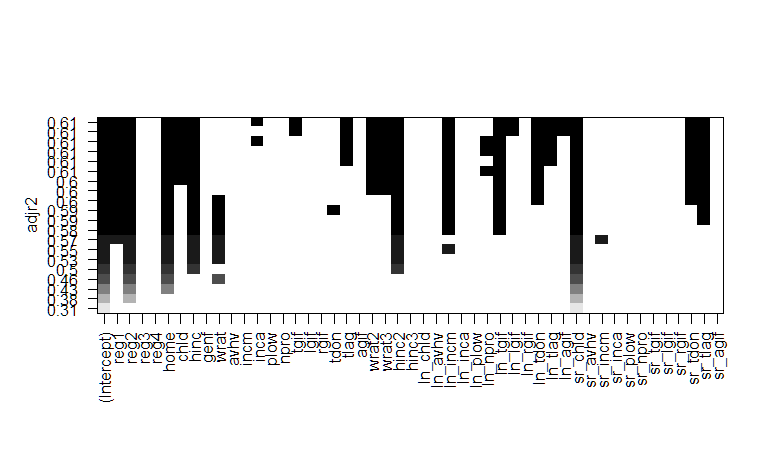
First, I divide up the data into training, test, and validation sets. The PART variable indicates which set each observation is assigned. The entire charity dataset consists of 3984 training observations, 2018 validation observations, and 2007 test observations. Next, I take the mean and standard deviation of all variables for all sets in order to standardize all variable data to have zero mean and unit standard deviation. I then create a separate dataframe for each set to classify DONR and a separate dataframe for each set to predict DAMT. Second, I perform standardization for the whole data set so that all variable data has zero mean and unit standard deviation. Then, I split up the data into training, test and validation sets. Both sequences end up with the following data variables.

|  |  |  |  |
| --- | --- | --- | --- |
| Data set | # Obs | # Var | Purpose |
| data.train | 3984 | 55 | Unstandardized training set |
| data.train.std.c | 3984 | 52 | To classify DONR |
| data.train.std.y | 1995 | 52 | To predict DAMT |
| data.valid | 2018 | 55 | Unstandardized validation set |
| data.valid.std.c | 2018 | 52 | To classify DONR |
| data.valid.std.y | 999 | 52 | To predict DAMT |
| data.test | 2007 | 55 | Unstandardized test set |
| data.test.std | 2007 | 51 | Standardized test set with missing  response variable observations |

# 3. Building Classification Models

The next step is to find the best subset of variables to use when building models. The original data set included 24 variables with 2 response variables, 1 index variable, and 1 data set specification variable (PART). Thus, there were only 20 predictor variables in the original charity data set. Even though the variable set has expanded to 51 predictor variables, I still expect the best model to have the same number of variables as the original data set or less.

First, I run an automated variable selection procedure on the training data set with the function REGSUBSETS in the LEAPS package with the “exhaustive search” method to find the best 20 variables out of the total 51 variables (including the newly added variables) for classifying DONR. The following plot is a graphical representation of the best models with 1 through 20 variables, where black colored squares indicate that the particular variable was included in that model. The y-axis orders the models by Adjusted R-Squared values. This figure indicates that the model with 20 variables has the highest Adjusted R-Squared value.



I do not immediately assume that the model with 20 variables has the best performing subset of variables. There are many redundant variables that may add unwanted multicollinearity problems such as the original variable and their transformed variant. Even if multicollinearity issues do not arise, it will be better to go with a more parsimonious model rather than a more complex model if performance is similar. More complex models will tend to perform well on the training set but perform worse on the test set due to overfitting. The following table shows the variables that appear more than once in different forms in the REGSUBSETS best model.

|  |  |  |
| --- | --- | --- |
| Redundant Variables | | |
| CHLD | SR\_CHLD |  |
| HINC | HINC2 |  |
| WRAT2 | WRAT3 |  |
| TGIF | LN\_TGIF |  |
| TLAG | LN\_TLAG | SR\_TLAG |
| LN\_TDON | SR\_TDON |  |

In order to trim the subset even more, I choose the variant that appears the most frequently and more than twice in the above graphical representation of the automated variable selection procedure. If multiple variants of the same variable appear in many of the 20 models, then each will be added to this new trimmed subset. The result is the following trimmed subset of 12 variables. I’ve included the default subset of all original 20 predictor variables (plus HINC2 and LN\_AVHV added into the given code) and the variable selection subset of 20 variables with this trimmed subset.

|  |  |  |
| --- | --- | --- |
| Default original set (DO) | Variable selection subset (VS) | Trimmed subset (TS) |
| REG1 | REG1 | REG1 |
| REG2 | REG2 | REG2 |
| REG3 | HOME | HOME |
| REG4 | CHLD | SR\_CHLD |
| HOME | HINC | HINC |
| CHLD | INCA | HINC2 |
| HINC | TGIF | WRAT2 |
| HINC2 | TLAG | WRAT3 |
| GENF | WRAT2 | LN\_TGIF |
| WRAT | WRAT3 | SR\_TDON |
| LN\_AVHV | HINC2 | SR\_TLAG |
| INCM | LN\_INCM | LN\_INCM |
| INCA | LN\_TGIF |  |
| PLOW | LN\_LGIF |  |
| NPRO | LN\_TDON |  |
| TGIF | LN\_TLAG |  |
| LGIF | LN\_AGIF |  |
| RGIF | SR\_CHLD |  |
| TDON | SR\_TDON |  |
| TLAG | SR\_TLAG |  |
| AGIF |  |  |

Model Building Procedure

I will calculate profit for a particular classification model applied to the validation data. Each donor donates $14.50 on average and each mailing costs $2.00. Therefore, in order to find an ordered profit function (ordered from most likely donor to least likely), I will calculate the posterior probabilities for the validation dataset. Next, I will sort DONR in order of the posterior probabilities from highest to lowest. Then, I will calculate the cumulative sum of (14.5 x DONR – 2) as I go down the list. After that, I will find the maximum of this profit function. Lastly, I will create a classification table (or confusion matrix) to visualize the results. I will repeat this 5 step process for each subsequent model. As classification tables may be arranged or formatted differently, the following table indicates the format for my classification tables.

|  |  |  |
| --- | --- | --- |
|  | Actual | |
| Predicted | 0 | 1 |
| 0 | True Negatives | False Negatives |
| 1 | False Positives | True Positives |

Please note that when I compare the model results below I do not mention Models e through f because for the most part they are exactly the same as their models a through c counterparts.

Classification Model 1: Logistic Regression

The first method I will apply is classic Logistic Regression with each subset of variables of the standardized training set to classify DONR and test the model’s performance on the standardized validation set. Logistic Regression will serve as a good starting point and standard for model comparison. The following table shows the classification table for all 6 Logistic Regression classification models. Models 1a, 1b, and 1c are built on the training where splitting was performed before standardization. Model 1b performs the best with the highest true positives and true negatives and the lowest false positives and false negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 1a: DO subset  Splitting Before Standardization | | Classification Model 1b: VS subset  Splitting Before Standardization | | Classification Model 1c: TS subset  Splitting Before Standardization | |
| 709 | 18 | 752 | 12 | 677 | 5 |
| 310 | 981 | 267 | 987 | 342 | 994 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 1d: DO subset  Standardization Before Splitting | | Classification Model 1e: VS subset  Standardization Before Splitting | | Classification Model 1f: TS subset  Standardization Before Splitting | |
| 709 | 18 | 752 | 12 | 677 | 5 |
| 310 | 981 | 267 | 987 | 342 | 994 |

Classification Model 2: Linear Discriminant Analysis

Next, I perform Linear Discriminant Analysis (LDA) with each subset of variables of the standardized training set to classify DONR and test its performance on the standardized validation set. LDA is not normally used with qualitative predictors however at this stage the model building process, however the goal is to ultimately to find good predictive models. Model 2b performed the best with the most true positives plus true negatives and least false positives plus false negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 2a: DO subset  Splitting Before Standardization | | Classification Model 2b: VS subset  Splitting Before Standardization | | Classification Model 2c: TS subset  Splitting Before Standardization | |
| 675 | 14 | 719 | 6 | 667 | 5 |
| 344 | 985 | 300 | 993 | 352 | 994 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 2d: DO subset  Standardization Before Splitting | | Classification Model 2e: VS subset  Standardization Before Splitting | | Classification Model 2f: TS subset  Standardization Before Splitting | |
| 675 | 14 | 719 | 6 | 667 | 5 |
| 344 | 985 | 300 | 993 | 352 | 994 |

Classification Model 3: Quadratic Discriminant Analysis

Then, I build models with Quadratic Discriminant Analysis with each subset of variables of the standardized training set to classify DONR and test its performance on the standardized validation set.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 3a: DO subset  Splitting Before Standardization | | Classification Model 3b: VS subset  Splitting Before Standardization | | Classification Model 3c: TS subset  Splitting Before Standardization | |
| 610 | 36 | 499 | 16 | 493 | 22 |
| 409 | 963 | 520 | 983 | 526 | 977 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 3d: DO subset  Standardization Before Splitting | | Classification Model 3e: VS subset  Standardization Before Splitting | | Classification Model 3f: TS subset  Standardization Before Splitting | |
| 610 | 36 | 499 | 16 | 493 | 22 |
| 409 | 963 | 520 | 983 | 526 | 977 |

Classification Model 4: K Nearest Neighbors

Next, I build models using the non-parametric K Nearest Neighbors (KNN) method. I can generate 6 KNN models by varying the smoothness parameter k where k = 1 through 6. In order to guarantee reproducibility of results I use set.seed(1). Apparently, this is the only instance when splitting before or after standardization actually affects the results. All of the models perform similarly, however Model 4e performs of the 12 KNN models because it has the highest true positives plus true negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 4a: k = 1  Splitting Before Standardization | | Classification Model 4b: k = 2  Splitting Before Standardization | | Classification Model 4c: k = 3  Splitting Before Standardization | |
| 784 | 85 | 775 | 100 | 781 | 64 |
| 235 | 914 | 244 | 899 | 238 | 935 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 4d: k = 4  Splitting Before Standardization | | Classification Model 4e: k = 5  Splitting Before Standardization | | Classification Model 4f: k = 6  Splitting Before Standardization | |
| 771 | 59 | 771 | 49 | 766 | 51 |
| 248 | 940 | 248 | 950 | 253 | 948 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 4g: k = 1  Standardization Before Splitting | | Classification Model 4h: k = 2  Standardization Before Splitting | | Classification Model 4i: k = 3  Standardization Before Splitting | |
| 785 | 88 | 772 | 83 | 782 | 61 |
| 234 | 911 | 247 | 916 | 237 | 938 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 4j: k = 4  Standardization Before Splitting | | Classification Model 4k: k = 5  Standardization Before Splitting | | Classification Model 4l: k = 6  Standardization Before Splitting | |
| 777 | 63 | 767 | 51 | 760 | 50 |
| 242 | 936 | 252 | 948 | 259 | 949 |

Classification Model 5: Logistic Regression using GAM with Natural Splines

Following that, I build logistic regression models with natural splines using the GAM package and function. I only apply natural splines to continuous variables (not to any variables that can be considered categorical variables) with 4 degrees of freedom. Model 5b performs horribly compared to the other natural spline logistic regression models. Model 5c performs the best with the most true positives and true negatives and least false positives and false negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 5a: DO subset  Splitting Before Standardization | | Classification Model 5b: VS subset  Splitting Before Standardization | | Classification Model 5c: TS subset  Splitting Before Standardization | |
| 713 | 14 | 486 | 545 | 764 | 10 |
| 306 | 985 | 533 | 454 | 255 | 989 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 5d: DO subset  Standardization Before Splitting | | Classification Model 5e: VS subset  Standardization Before Splitting | | Classification Model 5f: TS subset  Standardization Before Splitting | |
| 713 | 14 | 486 | 545 | 764 | 10 |
| 306 | 985 | 533 | 454 | 255 | 989 |

Classification Model 6: Logistic Regression using GAM with Smoothing Splines

After that, I build logistic regression models with smoothing splines using the GAM package and function. I only apply smoothing splines to continuous variables (not to any variables that can be considered categorical variables) with 4 degrees of freedom. All of the smoothing spline logistic regression models perform fairly well. Model 6c performs the best with the highest true positives plus true negatives and lowest false positives plus false negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 6a: DO subset  Splitting Before Standardization | | Classification Model 6b: VS subset  Splitting Before Standardization | | Classification Model 6c: TS subset  Splitting Before Standardization | |
| 659 | 5 | 740 | 7 | 764 | 10 |
| 360 | 994 | 279 | 992 | 255 | 989 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 6d: DO subset  Standardization Before Splitting | | Classification Model 6e: VS subset  Standardization Before Splitting | | Classification Model 6f: TS subset  Standardization Before Splitting | |
| 659 | 5 | 740 | 7 | 764 | 10 |
| 360 | 994 | 279 | 992 | 255 | 989 |

Classification Model 7: Logistic Regression using GAM with Local Regression (span = 0.2)

Next, I build logistic regression models with local linear regression smoothing with spans covering 20% of the observations. I only apply local regression smoothing to continuous variables (not to any variables that can be considered categorical variables). Model 7b performs the best with highest true positives plus true negatives and lowest false positives and false negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 7a: DO subset  Splitting Before Standardization | | Classification Model 7b: VS subset  Splitting Before Standardization | | Classification Model 7c: TS subset  Splitting Before Standardization | |
| 705 | 13 | 786 | 15 | 770 | 8 |
| 314 | 986 | 233 | 984 | 249 | 991 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 7d: DO subset  Standardization Before Splitting | | Classification Model 7e: VS subset  Standardization Before Splitting | | Classification Model 7f: TS subset  Standardization Before Splitting | |
| 705 | 13 | 786 | 15 | 770 | 8 |
| 314 | 986 | 233 | 984 | 249 | 991 |

Classification Model 8: Logistic Regression using GAM with Local Regression (span = 0.5)

Next, I build logistic regression models with local linear regression smoothing with spans covering 50% of the observations. I only apply local regression smoothing to continuous variables (not to any variables that can be considered categorical variables). Model 8c performs the best with most true positives and true negatives and least false positives and false negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 8a: DO subset  Splitting Before Standardization | | Classification Model 8b: VS subset  Splitting Before Standardization | | Classification Model 8c: TS subset  Splitting Before Standardization | |
| 663 | 6 | 737 | 6 | 741 | 6 |
| 356 | 993 | 282 | 993 | 278 | 993 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 8d: DO subset  Standardization Before Splitting | | Classification Model 8e: VS subset  Standardization Before Splitting | | Classification Model 8f: TS subset  Standardization Before Splitting | |
| 663 | 6 | 737 | 6 | 741 | 6 |
| 356 | 993 | 282 | 993 | 278 | 993 |

Classification Model 9: Decision Tree

Now I build a full classification decision tree with all 3 subsets of variables using the tree function in the TREE package. In order to guarantee reproducibility of results, I use set.seed(1). Without any pruning of the decision tree, Model 9b and 9c perform exactly the same and Model 9a performs the best with the most true positives and true negatives and least false positives and false negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 9a: DO subset  Splitting Before Standardization | | Classification Model 9b: VS subset  Splitting Before Standardization | | Classification Model 9c: TS subset  Splitting Before Standardization | |
| 645 | 37 | 631 | 69 | 631 | 69 |
| 374 | 962 | 388 | 930 | 388 | 930 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 9d: DO subset  Standardization Before Splitting | | Classification Model 9e: VS subset  Standardization Before Splitting | | Classification Model 9f: TS subset  Standardization Before Splitting | |
| 645 | 37 | 631 | 69 | 631 | 69 |
| 374 | 962 | 388 | 930 | 388 | 930 |

Classification Model 10: Bagging

Next, I apply bagging to the 3 subsets using the randomForest package in R. Bagging is simply constructing regression trees using bootstrapped training sets, and averaging the resulting predictions. These trees are grown deep, and are not pruned. Hence each individual tree has high variance, but low bias. Bagging is simply a special case of a random forest with m = p (m = number of predictors considered for each tree, p = number of predictors). For bagging, I set the number of trees to 500. In order to guarantee reproducibility of results, I use set.seed(1). Model 10b performs the best with the highest true positives plus true negatives and lowest false positives plus false negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 10a: DO subset  Splitting Before Standardization | | Classification Model 10b: VS subset  Splitting Before Standardization | | Classification Model 10c: TS subset  Splitting Before Standardization | |
| 693 | 14 | 748 | 23 | 693 | 15 |
| 326 | 985 | 271 | 976 | 326 | 984 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 10d: DO subset  Standardization Before Splitting | | Classification Model 10e: VS subset  Standardization Before Splitting | | Classification Model 10f: TS subset  Standardization Before Splitting | |
| 693 | 14 | 710 | 17 | 693 | 15 |
| 326 | 985 | 309 | 982 | 326 | 984 |

Classification Model 11: Random Forests

Next, I apply random forests to the 3 subsets using the randomForest package in R. For random forests, m = round(sqrt(p)). In building a random forest, at each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors. In this way, strong and dominant predictors do not have as much influence on the tree structure as they would in bagging when all predictors are considered. For random forests, I set the number of trees to 500. In order to guarantee reproducibility of results, I use set.seed(1). Model 11a performs the best with the highest true positives plus true negatives and lowest false positives plus false negatives. Model 11a also produced most true negatives than any other method used thus far.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 11a: DO subset  Splitting Before Standardization | | Classification Model 11b: VS subset  Splitting Before Standardization | | Classification Model 11c: TS subset  Splitting Before Standardization | |
| 806 | 26 | 744 | 15 | 780 | 20 |
| 213 | 973 | 275 | 984 | 239 | 979 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 11d: DO subset  Standardization Before Splitting | | Classification Model 11e: VS subset  Standardization Before Splitting | | Classification Model 11f: TS subset  Standardization Before Splitting | |
| 804 | 26 | 744 | 15 | 780 | 20 |
| 215 | 973 | 275 | 984 | 239 | 979 |

Classification Model 12: Boosting

Next, I apply boosting to the 3 subsets using the GBM package in R. Boosting works in a similar way as bagging, except that the trees are grown sequentially: each tree is fit using the residuals from previously grown trees so that learning takes place. The major advantage of boosting is that it is very resistant against overfitting although it can occur very slowly. For boosting, I set the number of trees to 5000. In order to guarantee reproducibility of results, I use set.seed(1). Model 12a performs the best with the highest true positives plus true negatives and lowest false positives plus false negatives. Model 12a ties Model 12a for producing the most true negatives than any other method used thus far.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 12a: DO subset  Splitting Before Standardization | | Classification Model 12b: VS subset  Splitting Before Standardization | | Classification Model 12c: TS subset  Splitting Before Standardization | |
| 806 | 18 | 764 | 7 | 760 | 6 |
| 213 | 981 | 255 | 992 | 259 | 993 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 12d: DO subset  Standardization Before Splitting | | Classification Model 12e: VS subset  Standardization Before Splitting | | Classification Model 12f: TS subset  Standardization Before Splitting | |
| 800 | 18 | 751 | 5 | 760 | 6 |
| 219 | 981 | 268 | 994 | 259 | 993 |

Classification Model 13: Artificial Neural Network

After that, I fit a neural network with a single hidden layer of 20 nodes with 100 iterations and decay of 0.001 to the 3 subsets using the NNET package in R. Artificial neural networks are relatively crude electronic networks of neurons based on the neural structure of the brain. They process observations one at a time, and learn by comparing their classification of the observation with the known actual classification of the observation. The errors from the initial classification of the first observation is fed back into the network, and used to modify the network’s algorithm for further iterations. In order to guarantee reproducibility of results, I use set.seed(1). Model 13f performs the best with the highest true positives plus true negatives and lowest false positives plus false negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 13a: DO subset  Splitting Before Standardization | | Classification Model 13b: VS subset  Splitting Before Standardization | | Classification Model 13c: TS subset  Splitting Before Standardization | |
| 793 | 90 | 672 | 15 | 823 | 113 |
| 226 | 909 | 347 | 984 | 193 | 886 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 13d: DO subset  Standardization Before Splitting | | Classification Model 13e: VS subset  Standardization Before Splitting | | Classification Model 13f: TS subset  Standardization Before Splitting | |
| 780 | 71 | 688 | 19 | 808 | 63 |
| 239 | 928 | 331 | 980 | 211 | 936 |

Classification Model 14: Support Vector Machine (linear kernel)

After that, I use support vector machine with a linear kernel to the 3 subsets using the E1071 package in R. A support vector machine is a discriminative classifier formally defined by a separating hyperplane. For model 14, the hyperplane is based on a linear kernel. In order to guarantee reproducibility of results, I use set.seed(1). Model 14b performs the best with the highest true positives plus true negatives and lowest false positives plus false negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 14a: DO subset  Splitting Before Standardization | | Classification Model 14b: VS subset  Splitting Before Standardization | | Classification Model 14c: TS subset  Splitting Before Standardization | |
| 660 | 14 | 699 | 6 | 649 | 6 |
| 359 | 985 | 320 | 993 | 370 | 993 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 14d: DO subset  Standardization Before Splitting | | Classification Model 14e: VS subset  Standardization Before Splitting | | Classification Model 14f: TS subset  Standardization Before Splitting | |
| 659 | 14 | 695 | 6 | 649 | 6 |
| 360 | 985 | 324 | 993 | 370 | 993 |

Classification Model 15: Support Vector Machine (radial kernel)

After that, I use support vector machine with a linear kernel to the 3 subsets using the E1071 package in R. For model 15, the hyperplane is based on a radial kernel. In order to guarantee reproducibility of results, I use set.seed(1). Model 15d performs the best with the highest true positives plus true negatives and lowest false positives plus false negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 15a: DO subset  Splitting Before Standardization | | Classification Model 15b: VS subset  Splitting Before Standardization | | Classification Model 15c: TS subset  Splitting Before Standardization | |
| 625 | 7 | 653 | 10 | 664 | 10 |
| 394 | 992 | 366 | 989 | 355 | 989 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classification Model 15d: DO subset  Standardization Before Splitting | | Classification Model 15e: VS subset  Standardization Before Splitting | | Classification Model 15f: TS subset  Standardization Before Splitting | |
| 613 | 5 | 648 | 11 | 655 | 10 |
| 406 | 994 | 371 | 988 | 364 | 989 |

# 4. Selecting the Best Classification Model

In order to select the best model, I compute a whole host of classification metrics based off of the confusion table values. The following table shows all of the classification that I use and whether the value should be maximized or minimized to indicate a better fit.

|  |  |
| --- | --- |
| Metrics | Goal |
| Number of True Positives (TP) | Higher |
| Number of True Negatives (TN) | Higher |
| Number of False Positives | Lower |
| Number of False Negatives | Lower |
| Misclassification Rate (Misclass.Rate) | Lower |
| Area under the ROC curve (AUC) | Higher |
| Accuracy | Higher |
| Sensitivity (or Recall) | Higher |
| Specificity | Higher |
| Precision | Higher |
| Fall out | Lower |
| False Discovery Rate or False Positive Rate (FDR) | Lower |
| Miss Rate or False Negative Rate | Lower |
| F Measure (weighted avg of precision and recall) | Higher |
| Matthews correlation coefficient: correlation coefficient between the observed and predicted binary classifications, values between -1 and +1 (MCC) | Higher |
| Number of Mailings | Higher |
| Profit | Higher |

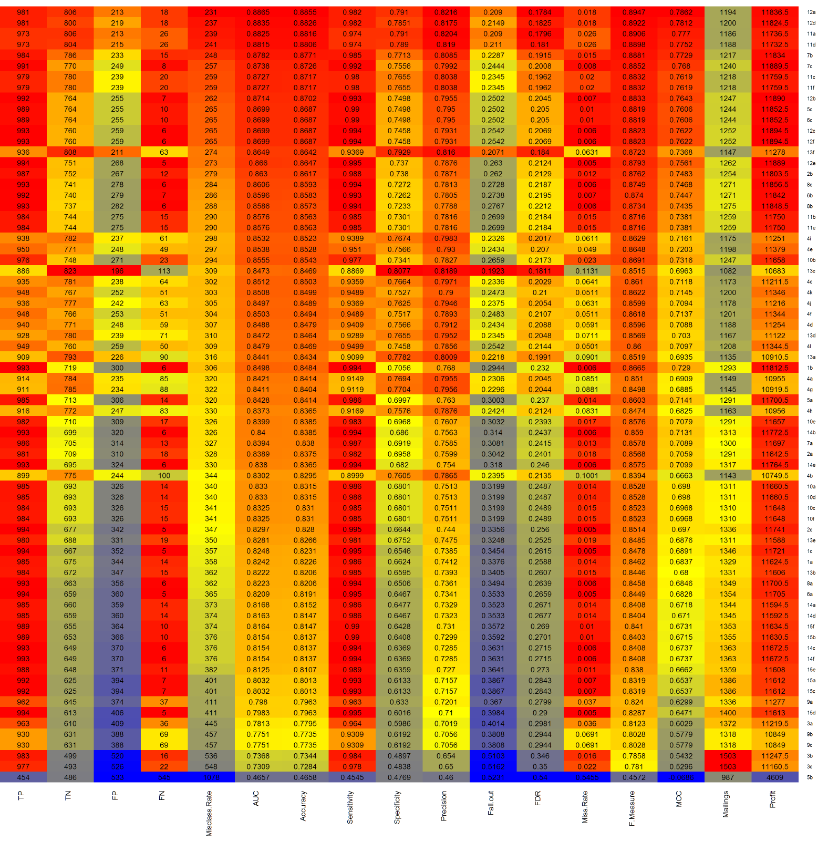
I scale all of the values to be a proportion of the highest score minus the lowest score for each metric so that I can compare the models against each other. I then calculate the mean value of the scaled scores and then generate a color based heat map to visualize the best model. Here the worst models have values in blue colored cells and the best models have values in red colored cells. The color gradient moves from blue to yellow to red, therefore the best classification model will be most deeply colored red.

As you can see in the color based heat map, the boosting and random forest models performed the best. Model 12a, a boosting (boosted random forest of classification decision trees) model using the default original set of variables with data that was standardized before data splitting, performed the best based on these classification metrics out of all the classification models. With this specific boosting model, I used 5000 trees, a shrinkage parameter of 0.01, and an interaction depth (tree size) of 4.

I am not surprised that boosting generated the best model. It is one of the most powerful methods that we learned this quarter in Predict 422. Boosting, like bagging, is an approach for improving prediction results for various machine learning methods. It particularly well suited to decision trees. Rather than resampling the data, boosting weights on some examples during learning. In boosting, trees are grown sequentially so that each tree is grown using information from previously grown trees. Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly. By fitting small trees to the residuals, we slowly improve the tree modeling function in areas where it does not perform well. The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

As the Model labels are too small to easily read, here are the models ranked by mean value of scaled scores from best to worst.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Rank | Model | Rank | Model | Rank | Model | Rank | Model | Rank | Model | Rank |
| 12a | 1 | 12f | 13 | 13c | 25 | 5a | 37 | 2c | 49 | 14f | 61 |
| 12d | 2 | 13f | 14 | 4c | 26 | 4h | 38 | 13e | 50 | 15e | 62 |
| 11a | 3 | 12e | 15 | 4k | 27 | 10e | 39 | 1c | 51 | 15a | 63 |
| 11d | 4 | 2b | 16 | 4j | 28 | 14b | 40 | 1a | 52 | 15c | 64 |
| 7b | 5 | 8c | 17 | 4f | 29 | 7a | 41 | 13b | 53 | 9a | 65 |
| 7c | 6 | 6b | 18 | 4d | 30 | 2a | 42 | 8a | 54 | 15d | 66 |
| 11c | 7 | 8b | 19 | 13d | 31 | 14e | 43 | 6a | 55 | 3a | 67 |
| 11f | 8 | 11b | 20 | 4l | 32 | 4b | 44 | 14a | 56 | 9b | 68 |
| 12b | 9 | 11e | 21 | 13a | 33 | 10a | 45 | 14d | 57 | 9c | 69 |
| 5c | 10 | 4i | 22 | 1b | 34 | 10d | 46 | 15f | 58 | 3b | 70 |
| 6c | 11 | 4e | 23 | 4a | 35 | 10c | 47 | 15b | 59 | 3c | 71 |
| 12c | 12 | 10b | 24 | 4g | 36 | 10f | 48 | 14c | 60 | 5b | 72 |



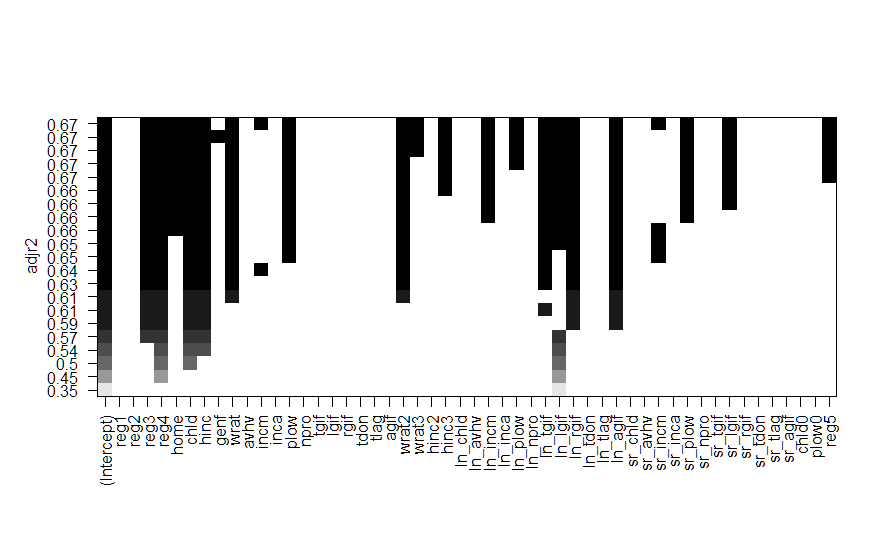
Next, I calculate the posterior probabilities for the test data set. Then, I sort DONR in order of the posterior probabilities from highest to lowest. I calculate the cumulative sum of (14.5 x DONR – 2) while going down the list. Then, I find the maximum of this profit function. Based on Model 12a applied to the test data set, I will mail to the 278 highest posterior probabilities.

Best classification model = Model 12a: Boosting (standardization before splitting)

|  |  |
| --- | --- |
| Classification Model 12a: DO subset  Splitting Before Standardization | |
| 806 | 18 |
| 213 | 981 |

# 5. Building Prediction Models

In the following prediction models, I will use data where standardization was performed before splitting and splitting was performed before standardization. Similar to the process with classification models, I will start out this section by identifying the best variable set to predict donation amount in the response variable DAMT. First, I run an automated variable selection procedure on the training data set with the function REGSUBSETS in the LEAPS package with the “exhaustive search” method to find the best 20 variables out of the total 51 variables (including the newly added variables) for predicting DAMT. The following plot is a graphical representation of the best models with 1 through 20 variables, where black colored squares indicate that the particular variable was included in that model. The y-axis orders the models by Adjusted R-Squared values. This figure indicates that the model with 20 variables has the highest Adjusted R-Squared value.



It is very interesting that the other 3 region variables (REG3, REG4, and REG5) that were not selected than when classifying DONR (REG1 and REG2) were selected for predicting DAMT. It may just be a coincidence as the situation is completely different (classifying DONR vs predicting DAMT), so there may be no relationship between the variables selected with REGSUBSETS. There is a noticeable lack of TDON and TLAG variables which appeared in almost all of the classifying DONR subsets. Otherwise, the remaining selected variables are relatively similar to those for the classifying DONR list. Once again, I identify redundant variables in the REGSUBSETS subset in order to choose a more parsimonious and reduced subset.

|  |  |  |
| --- | --- | --- |
| Redundant Variables | | |
| WRAT | WRAT2 | WRAT3 |
| PLOW | LN\_PLOW | SR\_PLOW |
| LN\_INCM | SR\_INCM |  |
| LN\_LGIF | SR\_LGIF |  |

Only variables that appear in REGSUBSET results more than twice are included in the variable selection subset. Both WRAT and WRAT2 are included in a majority of the subsets, however WRAT3 is only included in 3, so it is eliminated from the trimmed subset. Similarly, PLOW is included in a majority of the subsets, however LN\_PLOW and SR\_PLOW are only in 4 and 5, respectively, so they are taken out of the trimmed subset. The same reasoning is applied for leaving out SR\_INCM and SR\_LGIF. Please note that the default original set does not include HINC2, which was included in the set of variables used to classify DONR.

|  |  |  |
| --- | --- | --- |
| Default original set (DO) | Variable selection subset (VS) | Trimmed subset (TS) |
| REG1 | REG3 | REG3 |
| REG2 | REG4 | REG4 |
| REG3 | HOME | REG5 |
| REG4 | CHLD | HOME |
| HOME | HINC | CHLD |
| CHLD | WRAT | HINC |
| HINC | PLOW | PLOW |
| GENF | WRAT2 | WRAT |
| WRAT | WRAT3 | WRAT2 |
| LN\_AVHV | HINC3 | HINC3 |
| INCM | LN\_INCM | LN\_INCM |
| INCA | LN\_PLOW | LN\_LGIF |
| PLOW | LN\_TGIF | LN\_TGIF |
| NPRO | LN\_LGIF | LN\_RGIF |
| TGIF | LN\_RGIF | LN\_AGIF |
| LGIF | LN\_AGIF |  |
| RGIF | SR\_INCM |  |
| TDON | SR\_PLOW |  |
| TLAG | SR\_LGIF |  |
| AGIF | REG5 |  |

In order to compare models I will compute and analyze goodness of fit and model performance metrics including mean squared error (MSE), standard error (Std Error), R Squared value (R2), Adjusted R Squared value (Adj R2), Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC).

Prediction Model 1: OLS Linear Regression

The first method I will apply is classic ordinary least squares regression with each subset of variables of the standardized training set to predict DAMT and test the model’s performance on the standardized validation set. Least Squares Regression will serve as a good starting point and standard for model comparison. The following table shows the classification table for all 6 Linear Regression classification models. Models 1a, 1b, and 1c are built on the training where splitting was performed before standardization. Model 1b performs the best with the lowest MSE, lowest Standard Error, highest R Squared, highest Adjusted R Squared, lowest AIC, and lowest BIC

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 1a: DO subset  Splitting Before Standardization | Prediction Model 1b: VS subset  Splitting Before Standardization | Prediction Model 1c: TS subset  Splitting Before Standardization |
| MSE | 1.867523 | 1.3547 | 1.392546 |
| Std Error | 0.1696615 | 0.1501628 | 0.1520951 |
| R2 | 0.5722429 | 0.668497 | 0.6604173 |
| Adj R2 | 0.5679089 | 0.6651383 | 0.6578434 |
| AIC | 6646.579 | 6138.016 | 6176.056 |
| BIC | 6784.96 | 6276.396 | 6282.987 |

Prediction Model 2: Ridge Regression with GLMNET

The next method I will apply is ridge regression using the GLMNET function which applies a tuning parameter to all estimated coefficients (not including the intercept) shrinking them towards 0 (but never actually equal to 0) as specified by the lambda value (λ). When λ = 0, the penalty term has no effect, and ridge regression will produce the least squares estimates. However, as λ→ ∞, the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach 0. I use cross validation to identify the optimal lambda from a grid of 10000 lambda values (between 0 and 10e+10) for each model. I have to specify a random seed first so my results will be reproducible, since the choice of the cross-validation folds is random.

The GLMNET package does not provide R Squared, Adjusted R Squared, AIC, or BIC values. The reason for this is that standard errors are not very meaningful for strongly biased estimates that arise from penalized estimation methods. Penalized estimation is a procedure that reduces the variance of estimators by introducing substantial bias. The bias of each estimator is therefore a major component of its mean squared error, whereas its variance may contribute only a small part.

Unfortunately, in most applications of penalized regression it is impossible to obtain a sufficiently precise estimate of the bias. Any bootstrap-based calculations can only give an assessment of the variance of the estimates. Reliable estimates of the bias are only available if reliable unbiased estimates are available, which is typically not the case in situations in which penalized estimates are used. Reporting a standard error of a penalized estimate therefore tells only part of the story. It can give a mistaken impression of great precision, completely ignoring the inaccuracy caused by the bias. Model 2b performs the best with the lowest MSE and lowest Standard Error.

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 2a: DO subset  Splitting Before Standardization | Prediction Model 2b: VS subset  Splitting Before Standardization | Prediction Model 2c: TS subset  Splitting Before Standardization |
| MSE | 1.868898 | 1.363563 | 1.420405 |
| Std Error | 0.1703007 | 0.151085 | 0.1545045 |
| λ | 0.0399264 | 0.01 | 0.01 |

Prediction Model 3: Lasso with GLMNET

The next method I will apply is the lasso using the GLMNET function which applies a tuning parameter to all estimated coefficients (not including the intercept) shrinking them towards 0 (where some actually equal to 0) as specified by the lambda value (λ). When λ = 0, the penalty term has no effect, and the lasso will produce the least squares estimates. However, as λ→ ∞, the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach and may equal 0. Therefore, the difference between ridge regression and the lasso are that some coefficient estimates are forced to be exactly equal to zero when the tuning parameter λ is sufficiently large. I use cross validation to identify the optimal lambda from a grid of 100 lambda values for each model. I have to specify a random seed first so my results will be reproducible, since the choice of the cross-validation folds is random. In the chart below, I list the lambda value and the number of coefficient estimates set exactly to 0. Model 3b performs the best with the lowest MSE and lowest Standard Error.

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 3a: DO subset  Splitting Before Standardization | Prediction Model 3b: VS subset  Splitting Before Standardization | Prediction Model 3c: TS subset  Splitting Before Standardization |
| MSE | 1.861808 | 1.412496 | 1.462396 |
| Std Error | 0.1694304 | 0.1537956 | 0.1566224 |
| λ | 0.01 | 0.66832615 | 0.01 |
| Num β= 0 | 1 | 2 | 0 |

Prediction Model 4: Principal Components Regression

Now, I add models built using Principal Components Regression. The result of running a Principal Components Regression model is a table of variance explained for each number of principal components included. I compute the ten-fold cross-validation error for each possible value of number of PCs used. I have to specify a random seed first so my results will be reproducible, since the choice of the cross-validation folds is random. The cross validation score which is reported as root mean squared error is computed for each number of PCs. I report the smallest cross validation error when all PCs are included in the model along with the mean square error and standard error. Model 4b performs the best with the lowest MSE and lowest Standard Error.

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 4a: DO subset  Splitting Before Standardization | Prediction Model 4b: VS subset  Splitting Before Standardization | Prediction Model 4c: TS subset  Splitting Before Standardization |
| MSE | 2.272426 | 1.981271 | 2.02648 |
| Std Error | 0.1924591 | 0.183062 | 0.1807794 |
| CV Error | 1.286 | 1.132 | 1.142 |

Prediction Model 5: Partial Least Squares

Now, I add models built using Partial Least Squares which works similar to Principal Components Regression. The result of running a Partial Least Squares model is a table of variance explained for each number of principal components included. I compute the ten-fold cross-validation error for each possible value of number of PCs used. I have to specify a random seed first so my results will be reproducible, since the choice of the cross-validation folds is random. The cross validation score which is reported as root mean squared error is computed for each number of PCs. I report the smallest cross validation error when all PCs are included in the model along with the mean square error and standard error. Model 5b performs the best with the lowest MSE and lowest Standard Error.

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 5a: DO subset  Splitting Before Standardization | Prediction Model 5b: VS subset  Splitting Before Standardization | Prediction Model 5c: TS subset  Splitting Before Standardization |
| MSE | 1.891649 | 1.455105 | 1.51345 |
| Std Error | 0.1712315 | 0.1548187 | 0.157661 |
| CV Error | 1.286 | 1.132 | 1.142 |

Prediction Model 6: Least Squares Regression using GAM with Natural Splines

Following that, I build logistic regression models with natural splines using the GAM package and function. I only apply natural splines to continuous variables (not to any variables that can be considered categorical variables) with 4 degrees of freedom. Model 6b performs horribly compared to the other natural spline least squares regression models. Here GAM does not provide R Squared or Adjusted R Squared values. Here Model 6b produces a very high error rate about 10 times as much as the other models. Model 6c performs the best with the lowest MSE and lowest Standard Error.

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 6a: DO subset  Splitting Before Standardization | Prediction Model 6b: VS subset  Splitting Before Standardization | Prediction Model 6c: TS subset  Splitting Before Standardization |
| MSE | 1.679664 | 11.97154 | 1.399422 |
| Std Error | 0.14687 | 0.547503 | 0.1391948 |
| AIC | 6303.19 | 6142.199 | 6140.123 |
| BIC | 6649.142 | 6469.281 | 6360.274 |

Prediction Model 7: Least Squares Regression with Smoothing Splines

Following that, I build logistic regression models with smoothing splines using the GAM package and function. I only apply smoothing splines to continuous variables (not to any variables that can be considered categorical variables) with 4 degrees of freedom. Model 6b performs horribly compared to the other smoothing spline least squares regression models. Here GAM does not provide R Squared or Adjusted R Squared values. Here Model 6b produces a very high error rate about 10 times as much as the other models. Here Model 7a performs the best with the lowest MSE, however Model 7c has the lowest Standard Error, AIC, and BIC

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 7a: DO subset  Splitting Before Standardization | Prediction Model 7b: VS subset  Splitting Before Standardization | Prediction Model 7c: TS subset  Splitting Before Standardization |
| MSE | 1.1569567 | 1.346153 | 1.335262 |
| Std Error | 0.1595578 | 0.1501249 | 0.1487894 |
| AIC | 6326.331 | 6148.189 | 6129.225 |
| BIC | 6464.712 | 6286.569 | 6236.156 |

Prediction Model 8: Least Squares Regression using GAM with Local Regression (span = 0.2)

Next, I build least squares regression models with local linear regression smoothing with spans covering 20% of the observations. I only apply local regression smoothing to continuous variables (not to any variables that can be considered categorical variables). Model 8c performs the best with the lowest MSE, Standard Error, AIC, and BIC.

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 8: DO subset  Splitting Before Standardization | Prediction Model 8b: VS subset  Splitting Before Standardization | Prediction Model 8c: TS subset  Splitting Before Standardization |
| MSE | 2.208205 | 1.341621 | 1.334256 |
| Std Error | 0.6697493 | 0.1463981 | 0.1458805 |
| AIC | 6351.096 | 6229.713 | 6154.597 |
| BIC | 6489.477 | 6368.094 | 6261.528 |

Prediction Model 9: Least Squares Regression using GAM with Local Regression (span = 0.5)

Next, I build least squares regression models with local linear regression smoothing with spans covering 50% of the observations. I only apply local regression smoothing to continuous variables (not to any variables that can be considered categorical variables). Model 9c performs the best with the lowest MSE, Standard Error, AIC, and BIC.

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 9a: DO subset  Splitting Before Standardization | Prediction Model 9b: VS subset  Splitting Before Standardization | Prediction Model 9c: TS subset  Splitting Before Standardization |
| MSE | 1.625461 | 1.345475 | 1.341848 |
| Std Error | 0.1700076 | 0.1492391 | 0.1484047 |
| AIC | 6334.956 | 6155.241 | 6132.99 |
| BIC | 6473.337 | 6293.622 | 6239.921 |

Prediction Model 10: Regression Tree

Now I build a full regression decision tree with all 3 subsets of variables using the tree function in the TREE package. Regression trees are very similar to classification trees. Instead of stratifying data into regions based on class proportions, it stratifies data by cutoff thresholds. In order to guarantee reproducibility of results, I use set.seed(1). Without any pruning of the decision tree, Models 10a, 10b, and 10c all perform exactly the same.

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 10a: DO subset  Splitting Before Standardization | Prediction Model 10b: VS subset  Splitting Before Standardization | Prediction Model 10c: TS subset  Splitting Before Standardization |
| MSE | 2.241075 | 2.241075 | 2.241075 |
| Std Error | 0.1920681 | 0.1920681 | 0.1920681 |
| Deviance | 1.917 | 1.917 | 1.917 |

Prediction Model 11: Bagging

Next, I apply bagging to the 3 subsets using the randomForest package in R. For bagging, I set the number of trees to 500. In order to guarantee reproducibility of results, I use set.seed(1). I post the highest R Squared value out of all 500 trees generated. Model 11b performs the best with lowest MSE, Standard Error, and second highest highest R Squared (just slightly less than of Model 11c).

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 11a: DO subset  Splitting Before Standardization | Prediction Model 11b: VS subset  Splitting Before Standardization | Prediction Model 11c: TS subset  Splitting Before Standardization |
| MSE | 1.70452 | 1.680767 | 1.690726 |
| Std Error | 0.1746157 | 0.1700015 | 0.1717354 |
| Highest R2 | 0.6026312 | 0.6069172 | 0.607098 |

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 11d: DO subset  Standardization Before Splitting | Prediction Model 11e: VS subset  Standardization Before Splitting | Prediction Model 11f: TS subset  Standardization Before Splitting |
| MSE | 1.706585 | 1.681794 | 1.691131 |
| Std Error | 0.1748117 | 0.1701418 | 0.1717578 |
| Highest R2 | 0.6015721 | 0.6066416 | 0.6063775 |

Prediction Model 12: Random Forests

Next, I apply random forests to the 3 subsets using the randomForest package in R. For random forests, I set m = round(sqrt(p)) and the number of trees to 500. In order to guarantee reproducibility of results, I use set.seed(1). I post the highest R Squared value out of all 500 trees generated. Model 12c performs the best with the lowest MSE, lowest Standard Error, and highest R Squared value than any previous prediction model I’ve built.

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 12a: DO subset  Splitting Before Standardization | Prediction Model 12b: VS subset  Splitting Before Standardization | Prediction Model 12c: TS subset  Splitting Before Standardization |
| MSE | 1.66794 | 1.632706 | 1.609385 |
| Std Error | 0.1733482 | 0.1732356 | 0.1702943 |
| Highest R2 | 0.604422 | 0.6152546 | 0.6217699 |

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 12d: DO subset  Standardization Before Splitting | Prediction Model 12e: VS subset  Standardization Before Splitting | Prediction Model 12f: TS subset  Standardization Before Splitting |
| MSE | 1.670802 | 1.6341 | 1.610562 |
| Std Error | 0.1735235 | 0.1733228 | 0.1704411 |
| Highest R2 | 0.6039727 | 0.6151073 | 0.6213717 |

Prediction Model 13: Boosting

Last, I apply boosting to the 3 subsets using the GBM package in R. For boosting, I set the number of trees to 5000. GBM does not provide any R Squared or Adjusted R Squared values. In order to guarantee reproducibility of results, I use set.seed(1). Model 13a performs the best

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 913: DO subset  Splitting Before Standardization | Prediction Model 13b: VS subset  Splitting Before Standardization | Prediction Model 13c: TS subset  Splitting Before Standardization |
| MSE | 1.437068 | 1.432977 | 1.432369 |
| Std Error | 0.1622368 | 0.1623572 | 0.1628687 |

|  |  |  |  |
| --- | --- | --- | --- |
| Metrics\Models | Prediction Model 13d: DO subset  Standardization Before Splitting | Prediction Model 13e: VS subset  Standardization Before Splitting | Prediction Model 13f: TS subset  Standardization Before Splitting |
| MSE | 1.442785 | 1.431097 | 1.430536 |
| Std Error | 0.1633923 | 0.1623848 | 0.162481 |

# 6. Selecting the Best Prediction Model

Similar to the process for choosing the best classification model, I also use a color based heat map to compare prediction models and select the model with the highest scaled mean value of scores which should appear as the top of the heat map and the most deeply red colored. For some methods and models, it was not meaningful to report AIC, BIC, R Squared, or Adjusted R Squared values so I compare models based on mean squared error (MSE) and standardized error (Std Error).

I am slightly surprised that GAMs with natural splines, smoothing splines, and local regression outperformed the tree based methods. Decision trees and random forests may not perform very well when all of the data is numeric. I believe that parametric methods based on least squares regression work well when there are unseen but present relationships between the response variables and predictor variables that can be modeled with a regression equation. However, in this situation for predicting DAMT, the GAM approach to nonparametric regression produces better fitting models than the parametric approaches. Some of the most powerful nonparametric methods we learned this quarter in Predict 422 are least squares regression with GAM using splines.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  | MSE | Std Error |  |

As you can see in the color based heat map, the least squares regression models using a generalized additive model with natural splines, smoothing splines, and local regression performed the best. Model 6c, a least squares regression model using GAM with natural splines using the trimmed subset of variables with data that was standardized before data splitting, performed the best based on mean squared error and standardized error metrics (common for all out of all the prediction models. With this specific GAM with natural splines, I used 4 knots for all continuous variables based on default settings (automatically placed according to percentiles).

Best prediction model = Model 6c: Boosting (standardization before splitting)

|  |  |
| --- | --- |
| Metrics\Models | Prediction Model 6c: TS subset  Splitting Before Standardization |
| MSE | 1.399422 |
| Std Error | 0.1391948 |
| AIC | 6140.123 |
| BIC | 6360.274 |

Finally, I save the predicted DONR and DAMT values for the test set in a CSV file entitled “JP.csv”. The column named “chat” contains all the predicted DONR values (0 = not donor, 1 = donor) and the column named “yhat” contains all the predicted DAMT values or donation amounts.

# 7. Conclusion

A charitable organization wishes to develop a machine learning model to improve the cost-effectiveness of their direct marketing campaigns to previous donors. In this course project, I developed a classification model using data from the most recent campaign to effectively capture likely donors so that the expected net profit is maximized. I also developed a prediction model to predict donation amounts for donors (consisting of the records for donors only).

I explored the data and analyzed the shape of the histogram of continuous variables. I added variable transformations based on the particular skewness of the variables including square root, natural logarithm, square, and cube transforms. I standardized the data both before and after splitting the data into training, test, and validation sets and then build many models based on various methods we covered in this quarter in Predict 422. For classification models, I used logistic regression, linear discriminant analysis, quadratic discriminant analysis, k nearest neighbors, logistic regression using GAM with natural splines, logistic regression using GAM with smoothing splines, logistic regression using GAM with local regression smoothing (span = 0.2, 0.5), decision tree, bagging, random forests, boosting, artificial neural networks, and support vector machines (linear and radial kernels). For prediction models, I used least squares regression, ridge regression, the lasso, principal components regression, least squares regression using GAM with natural splines, least squares regression using GAM with smoothing splines, least squares regression using GAM with local regression smoothing (span = 0.2, 0.5), decision tree, bagging, random forests, and boosting. The best classification model and method was boosting (5000 trees, interaction depth = 4, shrinkage parameter = 0.01) and the best prediction model and method was least squares regression using GAM with natural splines (4 knots). I should always pursue the simplest model to predict or explain dependent variables, but sometimes data does not cooperate and the search for good models leads to very complex, data-driven, nonparametric models as it did for the charity data set. However, sometimes the most powerful and complex models do not significantly outperform simpler models, and in these cases, it may be preferable to choose the parsimonious model for practicality, interpretation, and implementation reasons. Although they perform very well and can be very accurate, decision trees and random forests can take a very long time to train. The best methods for generating classification models are not always the best methods for generate prediction models.

I am grateful for the opportunity to have learned so many new and powerful machine learning algorithms and methods for analytical modeling. This project was a wonderful way to utilize and apply everything we learned about different methods of building models on a real world data set.

# Appendix: R Code

# PREDICT 422 Practical Machine Learning

# Course Project

# Winter 2016

# Joshua Peng

# OBJECTIVE: A charitable organization wishes to develop a machine learning

# model to improve the cost-effectiveness of their direct marketing campaigns

# to previous donors.

# 1) Develop a classification model using data from the most recent campaign that

# can effectively capture likely donors so that the expected net profit is maximized.

# 2) Develop a prediction model to predict donation amounts for donors - the data

# for this will consist of the records for donors only.

# Load the data

charity <- read.csv("charity.csv") # load the "charity.csv" file

attach(charity)

# Identify Variables with Missing Values

colnames(charity)[colSums(is.na(charity)) > 0]

# Adding in reference reg5 to analyze correlations

sum(reg1)

sum(reg2)

sum(reg3)

sum(reg4)

8009-(1605+2555+1071+1117)

reg5 = rep(0,nrow(charity))

for(i in 1:nrow(charity)) {

if (reg1[i]==0 && reg2[i]==0 && reg3[i]==0 && reg4[i]==0) {reg5[i]=1}}

charity$reg5 = reg5

charity.train <- charity[charity$part=="train",]

rm(charity.train)

# Analyzing REGx

library('corrplot') #package corrplot

regx\_cor = cor(charity.train[,c(2:5,25,22)])

corrplot(regx\_cor, method = "number") #plot matrix

# normality tests

library(nortest)

ad.test(hinc)

ad.test(npro)

ad.test(tdon)

# descriptive summary of variables

library(modeest)

summary(charity)

mfv(hinc)[1]

mfv(wrat)[1]

mfv(avhv)[1]

mfv(tlag)[1]

mfv(agif)[1]

# histograms

hist(reg1) # dichotomous

hist(reg2) # dichotomous

hist(reg3) # dichotomous

hist(reg4) # dichotomous

hist(home) # dichotomous

hist(chld) # zero inflated

hist(hinc) # median spike

hist(genf) # dichotomous

hist(wrat) # left skewed

hist(avhv) # right skewed

hist(incm) # right skewed

hist(inca) # right skewed

hist(plow) # right skewed, poisson

hist(npro) # right skewed

hist(tgif) # right skewed, poisson

hist(lgif) # right skewed, poisson

hist(rgif) # right skewed, poisson

hist(tdon) # slightly right skewed

hist(tlag) # right skewed

hist(agif) # right skewed

# predictor transformations

charity$wrat2 = wrat^2

charity$wrat3 = wrat^3

charity$hinc2 = hinc^2

charity$hinc3 = hinc^3

charity$ln\_chld = log(chld)

charity$ln\_avhv = log(avhv)

charity$ln\_incm = log(incm)

charity$ln\_inca = log(inca)

charity$ln\_plow = log(plow)

charity$ln\_npro = log(npro)

charity$ln\_tgif = log(tgif)

charity$ln\_lgif = log(lgif)

charity$ln\_rgif = log(rgif)

charity$ln\_tdon = log(tdon)

charity$ln\_tlag = log(tlag)

charity$ln\_agif = log(agif)

charity$sr\_chld = sqrt(chld)

charity$sr\_avhv = sqrt(avhv)

charity$sr\_incm = sqrt(incm)

charity$sr\_inca = sqrt(inca)

charity$sr\_plow = sqrt(plow)

charity$sr\_npro = sqrt(npro)

charity$sr\_tgif = sqrt(tgif)

charity$sr\_lgif = sqrt(lgif)

charity$sr\_rgif = sqrt(rgif)

charity$sr\_tdon = sqrt(tdon)

charity$sr\_tlag = sqrt(tlag)

charity$sr\_agif = sqrt(agif)

chld0 = rep(0,nrow(charity))

plow0 = rep(0,nrow(charity))

reg5 = rep(0,nrow(charity))

for(i in 1:nrow(charity)) {

if(chld[i]==0) {chld0[i]=1}

if(plow[i]==0) {plow0[i]=1}

if(reg1[i]==0 && reg2[i]==0 && reg3[i]==0 && reg4[i]==0) {reg5[i]=1}}

charity$reg5 = reg5

charity$chld0 = chld0

charity$plow0 = plow0

charity$ln\_plow[charity$ln\_plow <= 0] = 0

charity$ln\_chld[charity$ln\_chld <= 0] = 0

attach(charity)

# histograms after data transformations

hist(hinc2)

hist(hinc3)

hist(wrat2)

hist(wrat3)

hist(ln\_chld)

hist(ln\_avhv)

hist(ln\_incm)

hist(ln\_inca)

hist(ln\_plow)

hist(ln\_npro)

hist(ln\_tgif)

hist(ln\_lgif)

hist(ln\_rgif)

hist(ln\_tdon)

hist(ln\_tlag)

hist(ln\_agif)

hist(sr\_chld)

hist(sr\_avhv)

hist(sr\_incm)

hist(sr\_inca)

hist(sr\_plow)

hist(sr\_npro)

hist(sr\_tgif)

hist(sr\_lgif)

hist(sr\_rgif)

hist(sr\_tdon)

hist(sr\_tlag)

hist(sr\_agif)

# Set up data for analysis (initial steps)

data.train <- charity[charity$part=="train",]

x.train <- data.train[,c(2:21, 25:55)]

c.train <- data.train[,22] # donr

n.train.c <- length(c.train) # 3984

y.train <- data.train[c.train==1,23] # damt for observations with donr=1

n.train.y <- length(y.train) # 1995

data.valid <- charity[charity$part=="valid",]

x.valid <- data.valid[,c(2:21, 25:55)]

c.valid <- data.valid[,22] # donr

n.valid.c <- length(c.valid) # 2018

y.valid <- data.valid[c.valid==1,23] # damt for observations with donr=1

n.valid.y <- length(y.valid) # 999

data.test <- charity[charity$part=="test",]

n.test <- dim(data.test)[1] # 2007

x.test <- data.test[,c(2:21, 25:55)]

# Set up data for analysis (data splitting before standardization)

rm(data.train.std.y, data.train.std.c, data.valid.std.y, data.valid.std.c, data.test.std)

x.train.mean <- apply(x.train, 2, mean)

x.train.sd <- apply(x.train, 2, sd)

x.train.std <- t((t(x.train)-x.train.mean)/x.train.sd) # standardize to have zero mean and unit sd

apply(x.train.std, 2, mean) # check zero mean

apply(x.train.std, 2, sd) # check unit sd

data.train.std.c <- data.frame(x.train.std, donr=c.train) # to classify donr

data.train.std.y <- data.frame(x.train.std[c.train==1,], damt=y.train) # to predict damt when donr=1

x.valid.std <- t((t(x.valid)-x.train.mean)/x.train.sd) # standardize using training mean and sd

data.valid.std.c <- data.frame(x.valid.std, donr=c.valid) # to classify donr

data.valid.std.y <- data.frame(x.valid.std[c.valid==1,], damt=y.valid) # to predict damt when donr=1

x.test.std <- t((t(x.test)-x.train.mean)/x.train.sd) # standardize using training mean and sd

data.test.std <- data.frame(x.test.std)

# Set up data for analysis (Data splitting after standardization)

rm(data.train.std.y, data.train.std.c, data.valid.std.y, data.valid.std.c, data.test.std)

x.charity = charity[,c(2:21, 25:55)]

c.charity = charity[,22] # donr

n.charity.c = length(c.charity) # 8009

y.charity = charity[c.charity==1,23] # damt for observations with donr=1

n.charity.y = length(y.charity) # 5001

x.charity.mean = apply(x.charity, 2, mean)

x.charity.sd = apply(x.charity, 2, sd)

x.charity.std = t((t(x.charity)-x.charity.mean)/x.charity.sd) # standardize to have zero mean and unit sd

apply(x.charity.std, 2, mean) # check zero mean

apply(x.charity.std, 2, sd) # check unit sd

data.charity.std.c = data.frame(x.charity.std, donr=c.charity, part=charity$part) # to classify donr

data.charity.std.y = data.frame(x.charity.std[c.charity==1,], damt=y.charity) # to predict damt when donr=1

data.charity.std.y0 = data.frame(x.charity.std, part=charity$part) # add in part

data.charity.std.y1 = data.frame(data.charity.std.y0[c.charity==1,], damt=y.charity) # to predict damt when donr=1

i.train.c = which(data.charity.std.c$part=="train")

i.train.y = which(data.charity.std.y1$part=="train")

i.valid.c = which(data.charity.std.c$part=="valid")

i.valid.y = which(data.charity.std.y1$part=="valid")

i.test = which(charity$part=="test")

data.train.std.c = data.charity.std.c[i.train.c,]

data.train.std.y = data.charity.std.y[i.train.y,]

data.valid.std.c = data.charity.std.c[i.valid.c,]

data.valid.std.y = data.charity.std.y[i.valid.y,]

data.train.std.c$part = NULL

data.train.std.y$part = NULL

data.valid.std.c$part = NULL

data.valid.std.y$part = NULL

data.test.std = x.charity.std[i.test,]

##### CLASSIFICATION MODELING ######

library(MASS)

library(leaps)

library(gam)

library(ROCR)

library(AUC)

vs1 = regsubsets(donr ~ ., data = data.train.std.c, nvmax = 20)

plot(vs1, scale="adjr2")

# Logistic Regression

# M1c: Trimmed subset of 12 variables

model.log2 = glm(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

ln\_incm + ln\_tgif + sr\_tdon + sr\_tlag, data.train.std.c, family=binomial("logit"))

# M1b: Best 20 variable model from regsubsets

model.log2 = glm(donr ~ reg1 + reg2 + home + chld + hinc + inca + tgif + tlag + wrat2 +

wrat3 + hinc2 + ln\_incm + ln\_tgif + ln\_lgif + ln\_tdon + ln\_tlag +

ln\_agif + sr\_chld + sr\_tdon + sr\_tlag, data.train.std.c, family=binomial("logit"))

# M1a: Default Original subset

model.log2 <- glm(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data.train.std.c, family=binomial("logit"))

##### My Logistic Regression Model

post.valid.log2 <- predict(model.log2, data.valid.std.c, type="response") # n.valid post probs

profit.log2 <- cumsum(14.5\*c.valid[order(post.valid.log2, decreasing=T)]-2)

plot(profit.log2) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.log2) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.log2)) # report number of mailings and maximum profit

cutoff.log2 <- sort(post.valid.log2, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.log2 <- ifelse(post.valid.log2>cutoff.log2, 1, 0) # mail to everyone above the cutoff

table(chat.valid.log2, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.log2, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

#####

model.log1 <- glm(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data.train.std.c, family=binomial("logit"))

post.valid.log1 <- predict(model.log1, data.valid.std.c, type="response") # n.valid post probs

# calculate ordered profit function using average donation = $14.50 and mailing cost = $2

profit.log1 <- cumsum(14.5\*c.valid[order(post.valid.log1, decreasing=T)]-2)

plot(profit.log1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.log1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.log1)) # report number of mailings and maximum profit

# 1291.0 11642.5

cutoff.log1 <- sort(post.valid.log1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.log1 <- ifelse(post.valid.log1>cutoff.log1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.log1, c.valid) # classification table

# c.valid

#chat.valid.log1 0 1

# 0 709 18

# 1 310 981

# check n.mail.valid = 310+981 = 1291

# check profit = 14.5\*981-2\*1291 = 11642.5

# Results

# n.mail Profit Model

# 1329 11624.5 LDA1

# 1291 11642.5 Log1

# Linear Discriminant Analysis

# M2c: Trimmed subset of 12 variables

model.lda2 = lda(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

ln\_incm + ln\_tgif + sr\_tdon + sr\_tlag, data.train.std.c)

# M2b: Best 20 variable model from regsubsets

model.lda2 = lda(donr ~ reg1 + reg2 + home + chld + hinc + inca + tgif + tlag + wrat2 +

wrat3 + hinc2 + ln\_incm + ln\_tgif + ln\_lgif + ln\_tdon + ln\_tlag +

ln\_agif + sr\_chld + sr\_tdon + sr\_tlag, data.train.std.c)

# M2a: Default Original subset

model.lda2 <- lda(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data.train.std.c) # include additional terms on the fly using I()

##### My Linear Discriminant Analysis

post.valid.lda2 <- predict(model.lda2, data.valid.std.c)$posterior[,2] # n.valid.c post probs

profit.lda2 <- cumsum(14.5\*c.valid[order(post.valid.lda2, decreasing=T)]-2)

plot(profit.lda2) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.lda2) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.lda2)) # report number of mailings and maximum profit

cutoff.lda2 <- sort(post.valid.lda2, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.lda2 <- ifelse(post.valid.lda2>cutoff.lda2, 1, 0) # mail to everyone above the cutoff

table(chat.valid.lda2, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.lda2, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

#####

model.lda1 <- lda(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data.train.std.c) # include additional terms on the fly using I()

# Note: strictly speaking, LDA should not be used with qualitative predictors,

# but in practice it often is if the goal is simply to find a good predictive model

post.valid.lda1 <- predict(model.lda1, data.valid.std.c)$posterior[,2] # n.valid.c post probs

# calculate ordered profit function using average donation = $14.50 and mailing cost = $2

profit.lda1 <- cumsum(14.5\*c.valid[order(post.valid.lda1, decreasing=T)]-2)

plot(profit.lda1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.lda1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.lda1)) # report number of mailings and maximum profit

# 1329.0 11624.5

cutoff.lda1 <- sort(post.valid.lda1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.lda1 <- ifelse(post.valid.lda1>cutoff.lda1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.lda1, c.valid) # classification table

# c.valid

#chat.valid.lda1 0 1

# 0 675 14

# 1 344 985

# check n.mail.valid = 344+985 = 1329

# check profit = 14.5\*985-2\*1329 = 11624.5

# Quadratic Discriminant Analysis

# M3c: Trimmed subset of 12 variables

model.qda2 = qda(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

ln\_incm + ln\_tgif + sr\_tdon + sr\_tlag, data.train.std.c)

# M3b: Best 20 variable model from regsubsets

model.qda2 = qda(donr ~ reg1 + reg2 + home + chld + hinc + inca + tgif + tlag + wrat2 +

wrat3 + hinc2 + ln\_incm + ln\_tgif + ln\_lgif + ln\_tdon + ln\_tlag +

ln\_agif + sr\_chld + sr\_tdon + sr\_tlag, data.train.std.c)

# M3a: Default Original subset

model.qda2 <- qda(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data.train.std.c) # include additional terms on the fly using I()

post.valid.qda2 <- predict(model.qda2, data.valid.std.c)$posterior[,2] # n.valid.c post probs

profit.qda2 <- cumsum(14.5\*c.valid[order(post.valid.qda2, decreasing=T)]-2)

plot(profit.qda2) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.qda2) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.qda2)) # report number of mailings and maximum profit

cutoff.qda2 <- sort(post.valid.qda2, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.qda2 <- ifelse(post.valid.qda2>cutoff.qda2, 1, 0) # mail to everyone above the cutoff

table(chat.valid.qda2, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.qda2, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

#####

# K Nearest Neighbors

library(class)

set.seed(1)

model.knn1=knn(data.train.std.c,data.valid.std.c,c.train,k=1)

set.seed(1)

model.knn1=knn(data.train.std.c,data.valid.std.c,c.train,k=2)

set.seed(1)

model.knn1=knn(data.train.std.c,data.valid.std.c,c.train,k=3)

set.seed(1)

model.knn1=knn(data.train.std.c,data.valid.std.c,c.train,k=4)

set.seed(1)

model.knn1=knn(data.train.std.c,data.valid.std.c,c.train,k=5)

set.seed(1)

model.knn1=knn(data.train.std.c,data.valid.std.c,c.train,k=6)

post.valid.knn1 <- as.integer(as.character(model.knn1))

profit.knn1 <- cumsum(14.5\*c.valid[order(post.valid.knn1, decreasing=T)]-2)

plot(profit.knn1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.knn1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.knn1)) # report number of mailings and maximum profit

cutoff.knn1 <- sort(post.valid.knn1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.knn1 <- ifelse(post.valid.knn1>cutoff.knn1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.knn1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.knn1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

# Logistic Regression using GAM with Natural Splines

library(gam)

# M5c: Trimmed subset of 12 variables

model.ns1 = gam(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

ns(ln\_incm,4) + ns(ln\_tgif,4) + ns(sr\_tdon,4) + ns(sr\_tlag,4),

data.train.std.c, family=binomial("logit"))

# M5b: Best 20 variable model from regsubsets

model.ns1 = gam(donr ~ reg1 + reg2 + home + chld + hinc + ns(inca,4) + ns(tgif,4) + ns(tlag,4) + wrat2 +

wrat3 + hinc2 + ns(ln\_incm,4) + ns(ln\_tgif,4) + ns(ln\_lgif,4) + ns(ln\_tdon,4) +

ns(ln\_tlag,4) + ns(ln\_agif,4) + sr\_chld + ns(sr\_tdon,4) + ns(sr\_tlag,4),

data.train.std.c, family=binomial("logit"))

# M5a: Default Original subset

model.ns1 <- gam(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

ns(ln\_avhv,4) + ns(incm,4) + ns(inca,4) + ns(plow,4) + ns(npro,4) + ns(tgif,4) +

ns(lgif,4) + ns(rgif,4) + ns(tdon,4) + ns(tlag,4) + ns(agif,4),

data.train.std.c, family=binomial("logit"))

post.valid.ns1 <- predict(model.ns1, data.valid.std.c, type="response") # n.valid post probs

profit.ns1 <- cumsum(14.5\*c.valid[order(post.valid.ns1, decreasing=T)]-2)

plot(profit.ns1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.ns1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.ns1)) # report number of mailings and maximum profit

cutoff.ns1 <- sort(post.valid.ns1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.ns1 <- ifelse(post.valid.ns1>cutoff.ns1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.ns1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.ns1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

# Logistic Regression using GAM with Smoothing Splines

library(gam)

# M6c: Trimmed subset of 12 variables

model.ss1 = gam(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

s(ln\_incm,4) + s(ln\_tgif,4) + s(sr\_tdon,4) + s(sr\_tlag,4),

data.train.std.c, family=binomial("logit"))

# M6b: Best 20 variable model from regsubsets

model.ss1 = gam(donr ~ reg1 + reg2 + home + chld + hinc + s(inca,4) + s(tgif,4) + s(tlag,4) + wrat2 +

wrat3 + hinc2 + s(ln\_incm,4) + s(ln\_tgif,4) + s(ln\_lgif,4) + s(ln\_tdon,4) +

s(ln\_tlag,4) + s(ln\_agif,4) + sr\_chld + s(sr\_tdon,4) + s(sr\_tlag,4),

data.train.std.c, family=binomial("logit"))

# M6a: Default Original subset

model.ss1 <- gam(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

s(ln\_avhv,4) + s(incm,4) + s(inca,4) + s(plow,4) + s(npro,4) + s(tgif,4) +

s(lgif,4) + s(rgif,4) + s(tdon,4) + s(tlag,4) + s(agif,4),

data.train.std.c, family=binomial("logit"))

post.valid.ss1 <- predict(model.ss1, data.valid.std.c, type="response") # n.valid post probs

profit.ss1 <- cumsum(14.5\*c.valid[order(post.valid.ss1, decreasing=T)]-2)

plot(profit.ss1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.ss1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.ss1)) # report number of mailings and maximum profit

cutoff.ss1 <- sort(post.valid.ss1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.ss1 <- ifelse(post.valid.ss1>cutoff.ss1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.ss1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.ss1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

# Logistic Regression using GAM with Local Regression Smoothing (span=0.2)

library(gam)

# M7c: Trimmed subset of 12 variables

model.lo1 = gam(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

lo(ln\_incm,span=0.2) + lo(ln\_tgif,span=0.2) + lo(sr\_tdon,span=0.2) + lo(sr\_tlag,span=0.2),

data.train.std.c, family=binomial("logit"))

# M7b: Best 20 variable model from regsubsets

model.lo1 = gam(donr ~ reg1 + reg2 + home + chld + hinc + lo(inca,span=0.2) + lo(tgif,span=0.2) +

lo(tlag,span=0.2) + wrat2 + wrat3 + hinc2 + lo(ln\_incm,span=0.2) + lo(ln\_tgif,span=0.2) +

lo(ln\_lgif,span=0.2) + lo(ln\_tdon,span=0.2) + lo(ln\_tlag,span=0.2) + lo(ln\_agif,span=0.2) +

sr\_chld + lo(sr\_tdon,span=0.2) + lo(sr\_tlag,span=0.2),

data.train.std.c, family=binomial("logit"))

# M7a: Default Original subset

model.lo1 <- gam(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

lo(ln\_avhv,span=0.2) + lo(incm,span=0.2) + lo(inca,span=0.2) + lo(plow,span=0.2) +

lo(npro,span=0.2) + lo(tgif,span=0.2) + lo(lgif,span=0.2) + lo(rgif,span=0.2) +

lo(tdon,span=0.2) + lo(tlag,span=0.2) + lo(agif,span=0.2),

data.train.std.c, family=binomial("logit"))

post.valid.lo1 <- predict(model.lo1, data.valid.std.c, type="response") # n.valid post probs

profit.lo1 <- cumsum(14.5\*c.valid[order(post.valid.lo1, decreasing=T)]-2)

plot(profit.lo1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.lo1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.lo1)) # report number of mailings and maximum profit

cutoff.lo1 <- sort(post.valid.lo1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.lo1 <- ifelse(post.valid.lo1>cutoff.lo1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.lo1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.lo1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

# Logistic Regression using GAM with Local Regression Smoothing (span=0.5)

library(gam)

# M8c: Trimmed subset of 12 variables

model.lo2 = gam(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

lo(ln\_incm,span=0.5) + lo(ln\_tgif,span=0.5) + lo(sr\_tdon,span=0.5) + lo(sr\_tlag,span=0.5),

data.train.std.c, family=binomial("logit"))

# M8b: Best 20 variable model from regsubsets

model.lo2 = gam(donr ~ reg1 + reg2 + home + chld + hinc + lo(inca,span=0.5) + lo(tgif,span=0.5) +

lo(tlag,span=0.5) + wrat2 + wrat3 + hinc2 + lo(ln\_incm,span=0.5) + lo(ln\_tgif,span=0.5) +

lo(ln\_lgif,span=0.5) + lo(ln\_tdon,span=0.5) + lo(ln\_tlag,span=0.5) + lo(ln\_agif,span=0.5) +

sr\_chld + lo(sr\_tdon,span=0.5) + lo(sr\_tlag,span=0.5),

data.train.std.c, family=binomial("logit"))

# M8a: Default Original subset

model.lo2 <- gam(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

lo(ln\_avhv,span=0.5) + lo(incm,span=0.5) + lo(inca,span=0.5) + lo(plow,span=0.5) +

lo(npro,span=0.5) + lo(tgif,span=0.5) + lo(lgif,span=0.5) + lo(rgif,span=0.5) +

lo(tdon,span=0.5) + lo(tlag,span=0.5) + lo(agif,span=0.5),

data.train.std.c, family=binomial("logit"))

post.valid.lo2 <- predict(model.lo2, data.valid.std.c, type="response") # n.valid post probs

profit.lo2 <- cumsum(14.5\*c.valid[order(post.valid.lo2, decreasing=T)]-2)

plot(profit.lo2) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.lo2) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.lo2)) # report number of mailings and maximum profit

cutoff.lo2 <- sort(post.valid.lo2, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.lo2 <- ifelse(post.valid.lo2>cutoff.lo2, 1, 0) # mail to everyone above the cutoff

table(chat.valid.lo2, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.lo2, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

# Decision Tree

library(tree)

# M9c: Trimmed subset of 12 variables

set.seed(1)

model.tr1 = tree(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

ln\_incm + ln\_tgif + sr\_tdon + sr\_tlag, data=data.train.std.c)

# M9b: Best 20 variable model from regsubsets

set.seed(1)

model.tr1 = tree(donr ~ reg1 + reg2 + home + chld + hinc + inca + tgif + tlag + wrat2 +

wrat3 + hinc2 + ln\_incm + ln\_tgif + ln\_lgif + ln\_tdon + ln\_tlag +

ln\_agif + sr\_chld + sr\_tdon + sr\_tlag, data=data.train.std.c)

# M9a: Default Original subset

set.seed(1)

model.tr1 <- tree(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.c) # include additional terms on the fly using I()

post.valid.tr1 <- predict(model.tr1, data.valid.std.c) # n.valid post probs

profit.tr1 <- cumsum(14.5\*c.valid[order(post.valid.tr1, decreasing=T)]-2)

plot(profit.tr1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.tr1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.tr1)) # report number of mailings and maximum profit

cutoff.tr1 <- sort(post.valid.tr1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.tr1 <- ifelse(post.valid.tr1>cutoff.tr1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.tr1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.tr1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

# Bagging

library(randomForest)

# M10c: Trimmed subset of 12 variables

set.seed(1)

model.bag1 = randomForest(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

ln\_incm + ln\_tgif + sr\_tdon + sr\_tlag, data=data.train.std.c,

mtry=12, ntree=500, importance=T)

# M10b: Best 20 variable model from regsubsets

set.seed(1)

model.bag1 = randomForest(donr ~ reg1 + reg2 + home + chld + hinc + inca + tgif + tlag + wrat2 +

wrat3 + hinc2 + ln\_incm + ln\_tgif + ln\_lgif + ln\_tdon + ln\_tlag +

ln\_agif + sr\_chld + sr\_tdon + sr\_tlag, data=data.train.std.c,

mtry=20, ntree=500, importance=T)

# M10a: Default Original subset

set.seed(1)

model.bag1 = randomForest(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.c, mtry=21, ntree=500, importance=T)

post.valid.bag1 <- predict(model.bag1, data.valid.std.c) # n.valid post probs

profit.bag1 <- cumsum(14.5\*c.valid[order(post.valid.bag1, decreasing=T)]-2)

plot(profit.bag1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.bag1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.bag1)) # report number of mailings and maximum profit

cutoff.bag1 <- sort(post.valid.bag1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.bag1 <- ifelse(post.valid.bag1>cutoff.bag1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.bag1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.bag1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

# Random Forests

library(randomForest)

# M11c: Trimmed subset of 12 variables

set.seed(1)

model.rf1 = randomForest(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

ln\_incm + ln\_tgif + sr\_tdon + sr\_tlag, data=data.train.std.c,

mtry=3, ntree=500, importance=T)

# M11b: Best 20 variable model from regsubsets

set.seed(1)

model.rf1 = randomForest(donr ~ reg1 + reg2 + home + chld + hinc + inca + tgif + tlag + wrat2 +

wrat3 + hinc2 + ln\_incm + ln\_tgif + ln\_lgif + ln\_tdon + ln\_tlag +

ln\_agif + sr\_chld + sr\_tdon + sr\_tlag, data=data.train.std.c,

mtry=4, ntree=500, importance=T)

# M11a: Default Original subset

set.seed(1)

model.rf1 <- randomForest(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.c, mtry=5, ntree=500, importance=T)

post.valid.rf1 <- predict(model.rf1, data.valid.std.c) # n.valid post probs

profit.rf1 <- cumsum(14.5\*c.valid[order(post.valid.rf1, decreasing=T)]-2)

plot(profit.rf1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.rf1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.rf1)) # report number of mailings and maximum profit

cutoff.rf1 <- sort(post.valid.rf1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.rf1 <- ifelse(post.valid.rf1>cutoff.rf1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.rf1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.rf1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

# Boosting

library(gbm)

# M12c: Trimmed subset of 12 variables

set.seed(1)

model.boo1 = gbm(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

ln\_incm + ln\_tgif + sr\_tdon + sr\_tlag, data=data.train.std.c,

distribution="gaussian", n.trees=5000,

interaction.depth=4,shrinkage=0.01,verbose=F)

# M12b: Best 20 variable model from regsubsets

set.seed(1)

model.boo1 = gbm(donr ~ reg1 + reg2 + home + chld + hinc + inca + tgif + tlag + wrat2 +

wrat3 + hinc2 + ln\_incm + ln\_tgif + ln\_lgif + ln\_tdon + ln\_tlag +

ln\_agif + sr\_chld + sr\_tdon + sr\_tlag, data=data.train.std.c,

distribution="gaussian", n.trees=5000,

interaction.depth=4,shrinkage=0.01,verbose=F)

# M12a: Default Original subset

set.seed(1)

model.boo1 <- gbm(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.c, distribution="gaussian", n.trees=5000,

interaction.depth=4,shrinkage=0.01,verbose=F)

post.valid.boo1 <- predict(model.boo1, data.valid.std.c, n.trees=5000) # n.valid post probs

profit.boo1 <- cumsum(14.5\*c.valid[order(post.valid.boo1, decreasing=T)]-2)

plot(profit.boo1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.boo1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.boo1)) # report number of mailings and maximum profit

cutoff.boo1 <- sort(post.valid.boo1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.boo1 <- ifelse(post.valid.boo1>cutoff.boo1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.boo1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.boo1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

# Artificial Neural Network

library(nnet)

xc\_vars = c("reg1","reg2","home","sr\_chld","hinc","hinc2","wrat2","wrat3",

"ln\_incm","ln\_tgif","sr\_tdon","sr\_tlag")

xb\_vars = c("reg1","reg2","home","chld","hinc","inca","tgif","tlag","wrat2",

"wrat3","hinc2","ln\_incm","ln\_tgif","ln\_lgif","ln\_tdon","ln\_tlag",

"ln\_agif","sr\_chld","sr\_tdon","sr\_tlag")

xa\_vars = c("reg1","reg2","reg3","reg4","home","chld","hinc","hinc2","genf","wrat",

"ln\_avhv","incm","inca","plow","npro","tgif","lgif","rgif","tdon",

"tlag","agif")

xc1 = data.train.std.c[,xc\_vars]

xb1 = data.train.std.c[,xb\_vars]

xa1 = data.train.std.c[,xa\_vars]

xvc1 = data.valid.std.c[,xc\_vars]

xvb1 = data.valid.std.c[,xb\_vars]

xva1 = data.valid.std.c[,xa\_vars]

# M12c: Trimmed subset of 12 variables

set.seed(1)

model.nn1 = nnet(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

ln\_incm + ln\_tgif + sr\_tdon + sr\_tlag,

data=data.train.std.c, size=20, maxit=100, decay=.001)

newdata=xvc1

# M12b: Best 20 variable model from regsubsets

set.seed(1)

model.nn1 = nnet(donr ~ reg1 + reg2 + home + chld + hinc + inca + tgif + tlag + wrat2 +

wrat3 + hinc2 + ln\_incm + ln\_tgif + ln\_lgif + ln\_tdon + ln\_tlag +

ln\_agif + sr\_chld + sr\_tdon + sr\_tlag,

data=data.train.std.c, size=20, maxit=100, decay=.001)

newdata=xvb1

# M12a: Default Original subset

set.seed(1)

model.nn1 <- nnet(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + hinc2 + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.c, size=20, maxit=100, decay=.001)

newdata=xva1

post.valid.nn1 <- predict(model.nn1, newdata) # n.valid post probs

profit.nn1 <- cumsum(14.5\*c.valid[order(post.valid.nn1, decreasing=T)]-2)

plot(profit.nn1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.nn1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.nn1)) # report number of mailings and maximum profit

cutoff.nn1 <- sort(post.valid.nn1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.nn1 <- ifelse(post.valid.nn1>cutoff.nn1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.nn1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.nn1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

# Support Vector Machines (Linear kernel)

library(e1071)

# M12c: Trimmed subset of 12 variables

set.seed(1)

model.svm1 = svm(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

ln\_incm + ln\_tgif + sr\_tdon + sr\_tlag,

data=data.train.std.c, kernel="linear", cost=0.1, scale=FALSE)

# M12b: Best 20 variable model from regsubsets

set.seed(1)

model.svm1 = svm(donr ~ reg1 + reg2 + home + chld + hinc + inca + tgif + tlag + wrat2 +

wrat3 + hinc2 + ln\_incm + ln\_tgif + ln\_lgif + ln\_tdon + ln\_tlag +

ln\_agif + sr\_chld + sr\_tdon + sr\_tlag,

data=data.train.std.c, kernel="linear", cost=0.1, scale=FALSE)

# M12a: Default Original subset

set.seed(1)

model.svm1 = svm(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + hinc2 + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.c, kernel="linear", cost=0.1, scale=FALSE)

post.valid.svm1 <- predict(model.svm1, data.valid.std.c) # n.valid post probs

profit.svm1 <- cumsum(14.5\*c.valid[order(post.valid.svm1, decreasing=T)]-2)

plot(profit.svm1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.svm1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.svm1)) # report number of mailings and maximum profit

cutoff.svm1 <- sort(post.valid.svm1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.svm1 <- ifelse(post.valid.svm1>cutoff.svm1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.svm1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.svm1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

# Support Vector Machines (Radial Kernel)

library(e1071)

# M12c: Trimmed subset of 12 variables

set.seed(1)

model.svm1 = svm(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

ln\_incm + ln\_tgif + sr\_tdon + sr\_tlag,

data=data.train.std.c, kernel="radial", cost=0.1, scale=FALSE)

# M12b: Best 20 variable model from regsubsets

set.seed(1)

model.svm1 = svm(donr ~ reg1 + reg2 + home + chld + hinc + inca + tgif + tlag + wrat2 +

wrat3 + hinc2 + ln\_incm + ln\_tgif + ln\_lgif + ln\_tdon + ln\_tlag +

ln\_agif + sr\_chld + sr\_tdon + sr\_tlag,

data=data.train.std.c, kernel="radial", cost=0.1, scale=FALSE)

# M12a: Default Original subset

set.seed(1)

model.svm1 = svm(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + hinc2 + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.c, kernel="radial", cost=0.1, scale=FALSE)

post.valid.svm1 <- predict(model.svm1, data.valid.std.c) # n.valid post probs

profit.svm1 <- cumsum(14.5\*c.valid[order(post.valid.svm1, decreasing=T)]-2)

plot(profit.svm1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.svm1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.svm1)) # report number of mailings and maximum profit

cutoff.svm1 <- sort(post.valid.svm1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.svm1 <- ifelse(post.valid.svm1>cutoff.svm1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.svm1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.svm1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

model.best.c = model.boo1 # model 12a

profit.best.c = profit.boo1

# select best model to have maximum profit in the validation sample

post.test <- predict(model.best.c, data.test.std, n.tree = 5000) # post probs for test data

# Oversampling adjustment for calculating number of mailings for test set

n.mail.valid <- which.max(profit.best.c)

tr.rate <- .1 # typical response rate is .1

vr.rate <- .5 # whereas validation response rate is .5

adj.test.1 <- (n.mail.valid/n.valid.c)/(vr.rate/tr.rate) # adjustment for mail yes

adj.test.0 <- ((n.valid.c-n.mail.valid)/n.valid.c)/((1-vr.rate)/(1-tr.rate)) # adjustment for mail no

adj.test <- adj.test.1/(adj.test.1+adj.test.0) # scale into a proportion

n.mail.test <- round(n.test\*adj.test, 0) # calculate number of mailings for test set

cutoff.test <- sort(post.test, decreasing=T)[n.mail.test+1] # set cutoff based on n.mail.test

chat.test <- ifelse(post.test>cutoff.test, 1, 0) # mail to everyone above the cutoff

table(chat.test)

# 0 1

# 1676 331

# based on this model we'll mail to the 331 highest posterior probabilities

##### PREDICTION MODELING ######

library(leaps)

library(stats)

vs2 = regsubsets(damt ~ ., data = data.train.std.y, really.big = T, nvmax = 20)

plot(vs2, scale="adjr2")

# Least Squares Regression

# M1c: Trimmed subset of 15 variables

model.ls1 = lm(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

hinc3 + ln\_incm + ln\_lgif + ln\_tgif + ln\_rgif + ln\_agif,

data.train.std.y)

# M1b: Best 20 variable model from regsubsets

model.ls1 = lm(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

wrat3 + hinc3 + ln\_incm + ln\_plow + ln\_tgif + ln\_lgif + ln\_rgif +

ln\_agif + sr\_incm + sr\_plow + sr\_lgif, data.train.std.y)

# M1a: Default Original subset

model.ls1 <- lm(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data.train.std.y)

pred.valid.ls1 <- predict(model.ls1, newdata = data.valid.std.y) # validation predictions

mean((y.valid - pred.valid.ls1)^2) # mean prediction error

# 1.867523

sd((y.valid - pred.valid.ls1)^2)/sqrt(n.valid.y) # std error

# 0.1696615

summary(model.ls1)$r.squared

summary(model.ls1)$adj.r.squared

AIC(model.ls1)

AIC(model.ls1,k=8.290042) # for BIC, k=ln(#obs in training set)

# Ridge Regression

library(glmnet)

grid=10^seq(10,-2,length=10000)

y = as.matrix(data.train.std.y$damt)

xc\_vars = c("reg3","reg4","reg5","home","chld","hinc","plow","wrat","wrat2",

"hinc3","ln\_incm","ln\_lgif","ln\_tgif","ln\_rgif","ln\_agif")

xb\_vars = c("reg3","reg4","reg5","home","chld","hinc","plow","wrat","wrat2",

"wrat3","hinc3","ln\_incm","ln\_plow","ln\_tgif","ln\_lgif","ln\_rgif",

"ln\_agif","sr\_incm","sr\_plow","sr\_lgif")

xa\_vars = c("reg1","reg2","reg3","reg4","home","chld","hinc","genf","wrat",

"ln\_avhv","incm","inca","plow","npro","tgif","lgif","rgif","tdon",

"tlag","agif")

xc = as.matrix(data.train.std.y)[,xc\_vars]

xb = as.matrix(data.train.std.y)[,xb\_vars]

xa = as.matrix(data.train.std.y)[,xa\_vars]

xvc = as.matrix(data.valid.std.y)[,xc\_vars]

xvb = as.matrix(data.valid.std.y)[,xb\_vars]

xva = as.matrix(data.valid.std.y)[,xa\_vars]

# M2c: Trimmed subset of 15 variables

set.seed(1)

cv.out = cv.glmnet(xc,y,alpha=0,lambda=grid)

bestlam = cv.out$lambda.min

newdata = xvc

model.rr1 = glmnet(xc,y,alpha=0,lambda=bestlam, standardize=FALSE)

# M2b: Best 20 variable model from regsubsets

set.seed(1)

cv.out = cv.glmnet(xb,y,alpha=0,lambda=grid, standardize=FALSE)

bestlam = cv.out$lambda.min

newdata = xvb

model.rr1 = glmnet(xb,y,alpha=0,lambda=bestlam, standardize=FALSE)

# M2a: Default Original subset

set.seed(1)

cv.out = cv.glmnet(xa,y,alpha=0,lambda=grid)

bestlam = cv.out$lambda.min

newdata = xva

model.rr1 = glmnet(xa,y,alpha=0,lambda=bestlam, standardize=FALSE)

pred.valid.rr1 = predict(model.rr1, newx=newdata, s=bestlam) # validation predictions

mean((y.valid - pred.valid.rr1)^2) # mean prediction error

sd((y.valid - pred.valid.rr1)^2)/sqrt(n.valid.y) # std error

bestlam

# Lasso

library(glmnet)

grid=10^seq(10,-2,length=10000)

y = as.matrix(data.train.std.y$damt)

xc\_vars = c("reg3","reg4","reg5","home","chld","hinc","plow","wrat","wrat2",

"hinc3","ln\_incm","ln\_lgif","ln\_tgif","ln\_rgif","ln\_agif")

xb\_vars = c("reg3","reg4","reg5","home","chld","hinc","plow","wrat","wrat2",

"wrat3","hinc3","ln\_incm","ln\_plow","ln\_tgif","ln\_lgif","ln\_rgif",

"ln\_agif","sr\_incm","sr\_plow","sr\_lgif")

xa\_vars = c("reg1","reg2","reg3","reg4","home","chld","hinc","genf","wrat",

"ln\_avhv","incm","inca","plow","npro","tgif","lgif","rgif","tdon",

"tlag","agif")

xc = as.matrix(data.train.std.y)[,xc\_vars]

xb = as.matrix(data.train.std.y)[,xb\_vars]

xa = as.matrix(data.train.std.y)[,xa\_vars]

xvc = as.matrix(data.valid.std.y)[,xc\_vars]

xvb = as.matrix(data.valid.std.y)[,xb\_vars]

xva = as.matrix(data.valid.std.y)[,xa\_vars]

# M3c: Trimmed subset of 15 variables

set.seed(1)

cv.out = cv.glmnet(xc,y,alpha=1,lambda=grid)

bestlam = cv.out$lambda.min

newdata = xvc

model.las1 = glmnet(xc,y,alpha=1,lambda=bestlam, standardize=FALSE)

# M3b: Best 20 variable model from regsubsets

set.seed(1)

cv.out = cv.glmnet(xb,y,alpha=1,lambda=grid, standardize=FALSE)

bestlam = cv.out$lambda.min

newdata = xvb

model.las1 = glmnet(xb,y,alpha=1,lambda=bestlam, standardize=FALSE)

# M3a: Default Original subset

set.seed(1)

cv.out = cv.glmnet(xa,y,alpha=1,lambda=grid)

bestlam = cv.out$lambda.min

newdata = xva

model.las1 = glmnet(xa,y,alpha=1,lambda=bestlam, standardize=FALSE)

pred.valid.las1 = predict(model.las1, newx=newdata, alpha=1, s=bestlam) # validation predictions

mean((y.valid - pred.valid.las1)^2) # mean prediction error

sd((y.valid - pred.valid.las1)^2)/sqrt(n.valid.y) # std error

bestlam

lasso.coef = predict(model.las1, newx=newdata, s=bestlam, alpha=1, type="coefficients")

lasso.coef

# Principal Components Regression

# M4c: Trimmed subset of 15 variables

library(pls)

set.seed(1)

model.pcr1 = pcr(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

hinc3 + ln\_incm + ln\_lgif + ln\_tgif + ln\_rgif + ln\_agif,

data=data.train.std.y, validation ="CV", scale = F)

# M4b: Best 20 variable model from regsubsets

set.seed(1)

model.pcr1 = pcr(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

wrat3 + hinc3 + ln\_incm + ln\_plow + ln\_tgif + ln\_lgif + ln\_rgif +

ln\_agif + sr\_incm + sr\_plow + sr\_lgif, data=data.train.std.y,

validation ="CV", scale = F)

# M4a: Default Original subset

set.seed(1)

model.pcr1 <- pcr(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.y, validation="CV", scale = F)

pred.valid.pcr1 <- predict(model.pcr1, newdata = data.valid.std.y) # validation predictions

mean((y.valid - pred.valid.pcr1)^2) # mean prediction error

sd((y.valid - pred.valid.pcr1)^2)/sqrt(n.valid.y) # std error

summary(model.pcr1)

# Partial Least Squares

# M5c: Trimmed subset of 15 variables

set.seed(1)

model.pls1 = plsr(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

hinc3 + ln\_incm + ln\_lgif + ln\_tgif + ln\_rgif + ln\_agif,

data=data.train.std.y, validation ="CV", scale = F)

# M5b: Best 20 variable model from regsubsets

set.seed(1)

model.pls1 = plsr(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

wrat3 + hinc3 + ln\_incm + ln\_plow + ln\_tgif + ln\_lgif + ln\_rgif +

ln\_agif + sr\_incm + sr\_plow + sr\_lgif, data=data.train.std.y,

validation ="CV", scale = F)

# M5a: Default Original subset

set.seed(1)

model.pls1 <- plsr(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.y, validation="CV", scale = F)

pred.valid.pls1 <- predict(model.pls1, newdata = data.valid.std.y) # validation predictions

mean((y.valid - pred.valid.pls1)^2) # mean prediction error

sd((y.valid - pred.valid.pls1)^2)/sqrt(n.valid.y) # std error

summary(model.pls1)

# Least Squares Regression using GAM with Natural Splines

library(gam)

# M6c: Trimmed subset of 12 variables

model.ns1 = gam(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + ns(plow,4) + wrat + wrat2 +

hinc3 + ns(ln\_incm,4) + ns(ln\_lgif,4) + ns(ln\_tgif,4) + ns(ln\_rgif,4) + ns(ln\_agif,4),

data.train.std.y, family=gaussian("identity"))

# M6b: Best 20 variable model from regsubsets

model.ns1 = gam(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + ns(plow,4) + wrat + wrat2 +

wrat3 + hinc3 + ns(ln\_incm,4) + ns(ln\_plow,4) + ns(ln\_tgif,4) +

ns(ln\_lgif,4) + ns(ln\_rgif,4) + ns(ln\_agif,4) + ns(sr\_incm,4) +

ns(sr\_plow,4) + ns(sr\_lgif,4),

data.train.std.y, family=gaussian("identity"))

# M6a: Default Original subset

model.ns1 <- gam(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +

ns(ln\_avhv,4) + ns(incm,4) + ns(inca,4) + ns(plow,4) + ns(npro,4) + ns(tgif,4) +

ns(lgif,4) + ns(rgif,4) + ns(tdon,4) + ns(tlag,4) + ns(agif,4),

data.train.std.y, family=gaussian("identity"))

pred.valid.ns1 <- predict(model.ns1, newdata = data.valid.std.y) # validation predictions

mean((y.valid - pred.valid.ns1)^2) # mean prediction error

sd((y.valid - pred.valid.ns1)^2)/sqrt(n.valid.y) # std error

AIC(model.ns1)

AIC(model.ns1,k=8.290042) # for BIC, k=ln(#obs in training set)

# Least Squares Regression using GAM with Smoothing Splines

library(gam)

# M7c: Trimmed subset of 12 variables

model.ss1 = gam(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + s(plow,4) + wrat + wrat2 +

hinc3 + s(ln\_incm,4) + s(ln\_lgif,4) + s(ln\_tgif,4) + s(ln\_rgif,4) + s(ln\_agif,4),

data.train.std.y, family=gaussian("identity"))

# M7b: Best 20 variable model from regsubsets

model.ss1 = gam(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + s(plow,4) + wrat + wrat2 +

wrat3 + hinc3 + s(ln\_incm,4) + s(ln\_plow,4) + s(ln\_tgif,4) +

s(ln\_lgif,4) + s(ln\_rgif,4) + s(ln\_agif,4) + s(sr\_incm,4) +

s(sr\_plow,4) + s(sr\_lgif,4),

data.train.std.y, family=gaussian("identity"))

# M7a: Default Original subset

model.ss1 <- gam(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +

s(ln\_avhv,4) + s(incm,4) + s(inca,4) + s(plow,4) + s(npro,4) + s(tgif,4) +

s(lgif,4) + s(rgif,4) + s(tdon,4) + s(tlag,4) + s(agif,4),

data.train.std.y, family=gaussian("identity"))

pred.valid.ss1 <- predict(model.ss1, newdata = data.valid.std.y) # validation predictions

mean((y.valid - pred.valid.ss1)^2) # mean prediction error

sd((y.valid - pred.valid.ss1)^2)/sqrt(n.valid.y) # std error

AIC(model.ss1)

AIC(model.ss1,k=8.290042) # for BIC, k=ln(#obs in training set)

# Least Squares Regression using GAM with Local Regression Smoothing (span=0.2)

library(gam)

# M8c: Trimmed subset of 12 variables

model.lo1 = gam(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + lo(plow,span=0.2) + wrat + wrat2 +

hinc3 + lo(ln\_incm,span=0.2) + lo(ln\_lgif,span=0.2) + lo(ln\_tgif,span=0.2) +

lo(ln\_rgif,span=0.2) + lo(ln\_agif,span=0.2),

data.train.std.y, family=gaussian("identity"))

# M8b: Best 20 variable model from regsubsets

model.lo1 = gam(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + lo(plow,span=0.2) + wrat + wrat2 +

wrat3 + hinc3 + lo(ln\_incm,span=0.2) + lo(ln\_plow,span=0.2) + lo(ln\_tgif,span=0.2) +

lo(ln\_lgif,span=0.2) + lo(ln\_rgif,span=0.2) + lo(ln\_agif,span=0.2) + lo(sr\_incm,span=0.2) +

lo(sr\_plow,span=0.2) + lo(sr\_lgif,span=0.2),

data.train.std.y, family=gaussian("identity"))

# M8a: Default Original subset

model.lo1 <- gam(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +

lo(ln\_avhv,span=0.2) + lo(incm,span=0.2) + lo(inca,span=0.2) +

lo(plow,span=0.2) + lo(npro,span=0.2) + lo(tgif,span=0.2) +

lo(lgif,span=0.2) + lo(rgif,span=0.2) + lo(tdon,span=0.2) +

lo(tlag,span=0.2) + lo(agif,span=0.2),

data.train.std.y, family=gaussian("identity"))

pred.valid.lo1 <- predict(model.lo1, newdata = data.valid.std.y) # validation predictions

mean((y.valid - pred.valid.lo1)^2) # mean prediction error

sd((y.valid - pred.valid.lo1)^2)/sqrt(n.valid.y) # std error

AIC(model.lo1)

AIC(model.lo1,k=8.290042) # for BIC, k=ln(#obs in training set)

# Least Squares Regression using GAM with Local Regression Smoothing (span=0.5)

library(gam)

# M9c: Trimmed subset of 12 variables

model.lo2 = gam(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + lo(plow,span=0.5) + wrat + wrat2 +

hinc3 + lo(ln\_incm,span=0.5) + lo(ln\_lgif,span=0.5) + lo(ln\_tgif,span=0.5) +

lo(ln\_rgif,span=0.5) + lo(ln\_agif,span=0.5),

data.train.std.y, family=gaussian("identity"))

# M9b: Best 20 variable model from regsubsets

model.lo2 = gam(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + lo(plow,span=0.5) + wrat + wrat2 +

wrat3 + hinc3 + lo(ln\_incm,span=0.5) + lo(ln\_plow,span=0.5) + lo(ln\_tgif,span=0.5) +

lo(ln\_lgif,span=0.5) + lo(ln\_rgif,span=0.5) + lo(ln\_agif,span=0.5) + lo(sr\_incm,span=0.5) +

lo(sr\_plow,span=0.5) + lo(sr\_lgif,span=0.5),

data.train.std.y, family=gaussian("identity"))

# M9a: Default Original subset

model.lo2 <- gam(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +

lo(ln\_avhv,span=0.5) + lo(incm,span=0.5) + lo(inca,span=0.5) +

lo(plow,span=0.5) + lo(npro,span=0.5) + lo(tgif,span=0.5) +

lo(lgif,span=0.5) + lo(rgif,span=0.5) + lo(tdon,span=0.5) +

lo(tlag,span=0.5) + lo(agif,span=0.5),

data.train.std.y, family=gaussian("identity"))

pred.valid.lo2 <- predict(model.lo2, newdata = data.valid.std.y) # validation predictions

mean((y.valid - pred.valid.lo2)^2) # mean prediction error

sd((y.valid - pred.valid.lo2)^2)/sqrt(n.valid.y) # std error

AIC(model.lo2)

AIC(model.lo2,k=8.290042) # for BIC, k=ln(#obs in training set)

# Decision Tree

library(tree)

# M10c: Trimmed subset of 12 variables

set.seed(1)

model.tr1 = tree(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

hinc3 + ln\_incm + ln\_lgif + ln\_tgif + ln\_rgif + ln\_agif,

data=data.train.std.y)

# M10b: Best 20 variable model from regsubsets

set.seed(1)

model.tr1 = tree(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

wrat3 + hinc3 + ln\_incm + ln\_plow + ln\_tgif + ln\_lgif + ln\_rgif +

ln\_agif + sr\_incm + sr\_plow + sr\_lgif, data=data.train.std.y)

# M10a: Default Original subset

set.seed(1)

model.tr1 <- tree(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.y) # include additional terms on the fly using I()

pred.valid.tr1 <- predict(model.tr1, newdata = data.valid.std.y) # validation predictions

mean((y.valid - pred.valid.tr1)^2) # mean prediction error

sd((y.valid - pred.valid.tr1)^2)/sqrt(n.valid.y) # std error

summary(model.tr1)

# Bagging

library(randomForest)

# M11c: Trimmed subset of 12 variables

set.seed(1)

model.bag1 = randomForest(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

hinc3 + ln\_incm + ln\_lgif + ln\_tgif + ln\_rgif + ln\_agif,

data=data.train.std.y, mtry=15, ntree=500, importance=T)

# M11b: Best 20 variable model from regsubsets

set.seed(1)

model.bag1 = randomForest(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

wrat3 + hinc3 + ln\_incm + ln\_plow + ln\_tgif + ln\_lgif + ln\_rgif +

ln\_agif + sr\_incm + sr\_plow + sr\_lgif, data=data.train.std.y,

mtry=20, ntree=500, importance=T)

# M11a: Default Original subset

set.seed(1)

model.bag1 <- randomForest(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.y, mtry=20, ntree=500, importance=T)

pred.valid.bag1 <- predict(model.bag1, newdata = data.valid.std.y) # validation predictions

mean((y.valid - pred.valid.bag1)^2) # mean prediction error

sd((y.valid - pred.valid.bag1)^2)/sqrt(n.valid.y) # std error

max(model.bag1$rsq)

# Random Forests

library(randomForest)

# M12c: Trimmed subset of 12 variables

set.seed(1)

model.rf1 = randomForest(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

hinc3 + ln\_incm + ln\_lgif + ln\_tgif + ln\_rgif + ln\_agif,

data=data.train.std.y, mtry=4, ntree=500, importance=T)

# M12b: Best 20 variable model from regsubsets

set.seed(1)

model.rf1 = randomForest(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

wrat3 + hinc3 + ln\_incm + ln\_plow + ln\_tgif + ln\_lgif + ln\_rgif +

ln\_agif + sr\_incm + sr\_plow + sr\_lgif, data=data.train.std.y,

mtry=4, ntree=500, importance=T)

# M12a: Default Original subset

set.seed(1)

model.rf1 <- randomForest(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.y, mtry=4, ntree=500, importance=T)

pred.valid.rf1 <- predict(model.rf1, newdata = data.valid.std.y) # validation predictions

mean((y.valid - pred.valid.rf1)^2) # mean prediction error

sd((y.valid - pred.valid.rf1)^2)/sqrt(n.valid.y) # std error

max(model.rf1$rsq)

# Boosting

library(gbm)

# M13c: Trimmed subset of 12 variables

set.seed(1)

model.boo1 = gbm(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

hinc3 + ln\_incm + ln\_lgif + ln\_tgif + ln\_rgif + ln\_agif, data=data.train.std.y,

distribution="gaussian", n.trees=5000,

interaction.depth=4,shrinkage=0.01,verbose=F)

# M13b: Best 20 variable model from regsubsets

set.seed(1)

model.boo1 = gbm(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + plow + wrat + wrat2 +

wrat3 + hinc3 + ln\_incm + ln\_plow + ln\_tgif + ln\_lgif + ln\_rgif +

ln\_agif + sr\_incm + sr\_plow + sr\_lgif, data=data.train.std.y,

distribution="gaussian", n.trees=5000,

interaction.depth=4,shrinkage=0.01,verbose=F)

# M13a: Default Original subset

set.seed(1)

model.boo1 <- gbm(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +

ln\_avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,

data=data.train.std.y, distribution="gaussian", n.trees=5000,

interaction.depth=4,shrinkage=0.01,verbose=F)

pred.valid.boo1 <- predict(model.boo1, newdata = data.valid.std.y, n.trees=5000) # validation predictions

mean((y.valid - pred.valid.boo1)^2) # mean prediction error

sd((y.valid - pred.valid.boo1)^2)/sqrt(n.valid.y) # std error

# Colormap for classification models

library(gplots)

library(RColorBrewer)

d1 = read.csv("classmap2.csv", row.names = 1, header=T)

d2 = read.csv("classactual2.csv", row.names = 1, header=T)

rownames = rownames(d1)

#d1 = d1[,-c(16,17)]

#d2 = d2[,-c(16,17)]

d1 = as.matrix(d1)

d2 = as.matrix(d2)

# creates a own color palette from blue to red

my\_palette <- colorRampPalette(c("blue","yellow","red"))(n = 299)

# (optional) defines the color breaks manually for a "skewed" color transition

col\_breaks = c(seq(0,0.799,length=100), # for blue

seq(0.80,0.929,length=100), # for yellow

seq(0.93,1.0,length=100)) # for red

# creates a 5 x 5 inch image

png("classheatmap.png", # create PNG for the heat map

width = 11\*300, # 5 x 300 pixels

height = 17\*300,

res = 300, # 300 pixels per inch

pointsize = 10) # smaller font size

heatmap.2(d1,

cellnote = d2, # same data set for cell labels

main = "Classification Metrics", # heat map title

notecol="black", # change font color of cell labels to black

density.info="none", # turns off density plot inside color legend

trace="none", # turns off trace lines inside the heat map

margins =c(10,8), # widens margins around plot

col=my\_palette, # use on color palette defined earlier

breaks=col\_breaks, # enable color transition at specified limits

dendrogram="none", # only draw a row dendrogram

Colv=FALSE, # turn off column clustering

Rowv=FALSE, # turn off row clustering

keysize=25,

lmat=rbind(c(2),c(3),c(1),c(4)),

lhei=c(5,5,18,0),

lwid=c(1))

dev.off()

# Colormap for prediction models

library(gplots)

library(RColorBrewer)

d3 = read.csv("predmap.csv", row.names = 1, header=T)

d4 = read.csv("predactual.csv", row.names = 1, header=T)

rownames = rownames(d3)

d3 = as.matrix(d3)

d4 = as.matrix(d4)

# creates a own color palette from blue to red

my\_palette <- colorRampPalette(c("red","yellow","blue"))(n = 299)

# (optional) defines the color breaks manually for a "skewed" color transition

col\_breaks = c(seq(0,0.03,length=100), # for red

seq(0.031,0.08,length=100), # for yellow

seq(0.081,1.0,length=100)) # for blue

# creates a 5 x 5 inch image

png("predheatmap.png", # create PNG for the heat map

width = 2.2\*300, # 5 x 300 pixels

height = 14\*300,

res = 300, # 300 pixels per inch

pointsize = 9) # smaller font size

heatmap.2(d3,

cellnote = d4, # same data set for cell labels

main = "Classification Metrics", # heat map title

notecol="black", # change font color of cell labels to black

density.info="none", # turns off density plot inside color legend

trace="none", # turns off trace lines inside the heat map

margins =c(12,10), # widens margins around plot

col=my\_palette, # use on color palette defined earlier

breaks=col\_breaks, # enable color transition at specified limits

dendrogram="none", # only draw a row dendrogram

Colv=FALSE, # turn off column clustering

Rowv=FALSE, # turn off row clustering

keysize=25,

lmat=rbind(c(2),c(3),c(1),c(4)),

lhei=c(5,5,18,0),

lwid=c(1))

dev.off()

# select model.ls2 since it has minimum mean prediction error in the validation sample

model.best.y = model.ns1

yhat.test <- predict(model.best.y, newdata = data.test.std) # test predictions

# FINAL RESULTS

# Save final results for both classification and regression

length(chat.test) # check length = 2007

length(yhat.test) # check length = 2007

chat.test[1:10] # check this consists of 0s and 1s

yhat.test[1:10] # check this consists of plausible predictions of damt

ip <- data.frame(chat=chat.test, yhat=yhat.test) # data frame with two variables: chat and yhat

write.csv(ip, file="JP.csv", row.names=FALSE) # use your initials for the file name

# submit the csv file in Angel for evaluation based on actual test donr and damt values