CHARITY ASSIGNMENT – GROUP PROJECT

INTRODUCTION

A charitable organization wishes to develop a machine learning model to improve the cost-effectiveness of their direct marketing campaigns to previous donors. According to their recent mailing records, the typical overall response rate is 10%.

The objective of this machine learning analysis is to develop a machine learning model to improve the cost-effectiveness, (profit) of their direct marketing campaigns to their previous donors. It is not cost effective to mail all previous donors based upon an overall response rate of 10% driving the expected profit from each mailing to ($0.55). This result of this analysis will provide a model which will predict the expected gift amounts from a list of likely donors in order to maximize profit from their mailing campaign.

To solve this problem, we will complete the assignment using the following steps.

1. EDA:- We will conduct Exploratory Data Analysis (EDA) on the data provided prior to building classification and prediction models.
2. Classification Model:- We will then develop a classification model for the DONR variable using other variables as predictors. We will build various classification models and work directly with the best of the model we design.
3. Prediction Models:- We will then develop a prediction model for the DAMT variable using the other variables as predictors. We will fit all candidate models using our training data and evaluate the fitted models using the validation data.
4. Submission:- We will conclude our analysis by preparing a test set classification and predictions into a separate file.

ANALYSIS

Exploratory Data Analysis (EDA)

The exploratory data analysis provides an approach towards analyzing the data prior to beginning the classification and prediction models. The charity data set contains 8,009 observations with 24 variables.

There are three main components in developing this predictive model:

- **Training data set** – utilized for exploratory data analysis, data preparation, building and selecting a predictive model. This data-set contains 3,984 observations utilizing up to 36 variables.
- **Validation data set** – utilized to evaluate the model selected in the training and testing phases of this analysis. This data-set contains 2,018 observations utilizing 33 variables.
- **Test data set** – utilized to score the model selected in the training phase of this analysis. This data-set contains 2,007 observations utilizing up to 36 variables.

Table 1 below is the data dictionary, which identifies each variable, type, and definition.

VARIABLE NAME	TYPE	DEFINITION
ID number	Continuous	Observation row count
REG1	Categorical	Region
REG2	Categorical	Region
REG3	Categorical	Region
REG4	Categorical	Region
HOME	Categorical	1 = homeowner, 0 = not a homeowner
CHLD	Categorical	Number of children
HINC	Categorical	Household income
GENF	Categorical	Gender (0 = Male, 1 = Female)
WRAT	Continuous	Wealth Rating
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	Continuous	Classification Response Variable (1 = Donor, 0 = Non-donor)
DAMT	Continuous	Prediction Response Variable (Donation Amount in $)

Table 1: Data Dictionary

Predictor Transformations

Prior to working on the classification models and prediction models, we will analyze some of the variables and apply various transformations to bring the data-set to 'normal' levels. The results below are reflective of various transformations on the variables applied, with key transformations being highlighted. Based upon the skewed distributions of 12 variables within this data-set we are performing a combination of log and power transformations to restore symmetry to these variables as well as to improve model performance as shown in table 2 below.

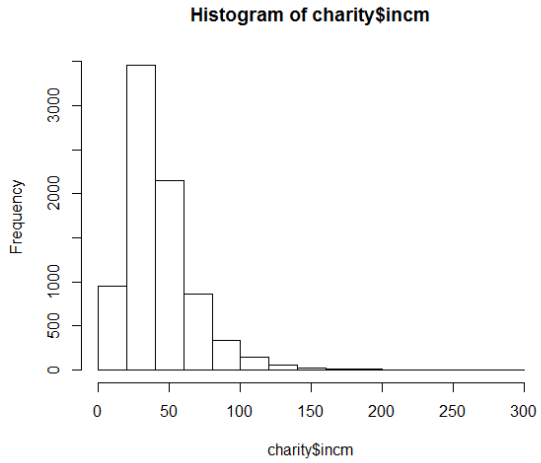
Variable	Transformation
INCM	Log
INCA	Log
TGIF	Log
AGIF	Log
TDON	Log
AVHV	Log
TLAG	Log
NPRO	Power, 2/3rd Root
PLOW	Power, 3rd Root
LGIF	Power, 5th Root
RGIF	Power, 7th Root
TLAG	Power, 5th Root

Table 2: Variable Transformations

Variable histograms below demonstrate the prior and post effect of log and power transformations. Both histograms demonstrate clear skewness prior and symmetry after the transformations were performed, as shown in tables 3 and 4 below.

Table 3 :- Median Family Income Transformation

Median Family Income, INCM



Log Median Family Income, INCM

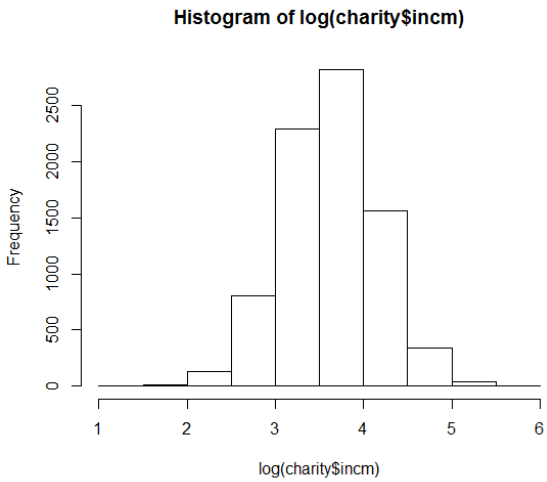
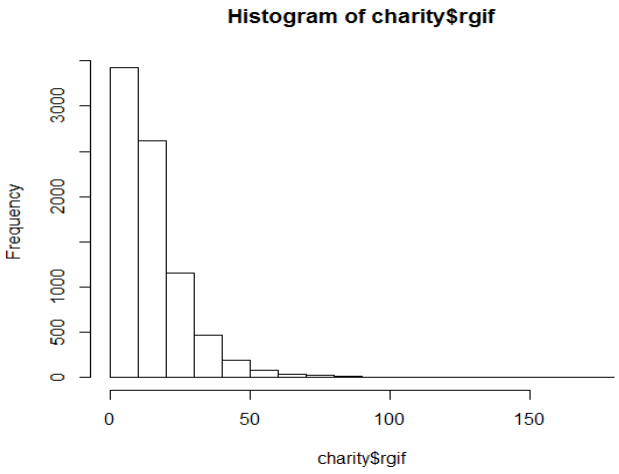
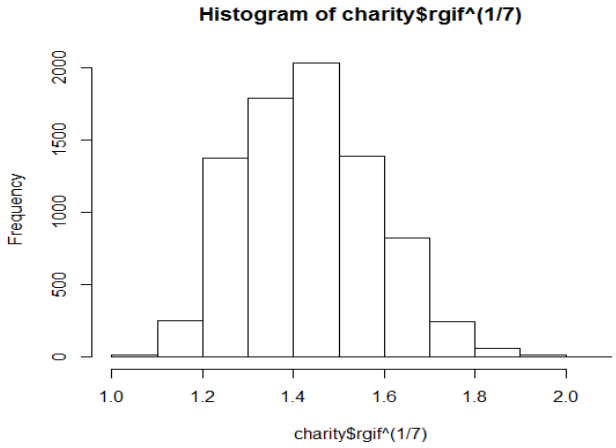


Table 4: Dollar Amount of Recent Gift Transformation

Dollar Amount Recent Gift, RGIF



7th Root of Dollar Amount Recent Gift, RGIF



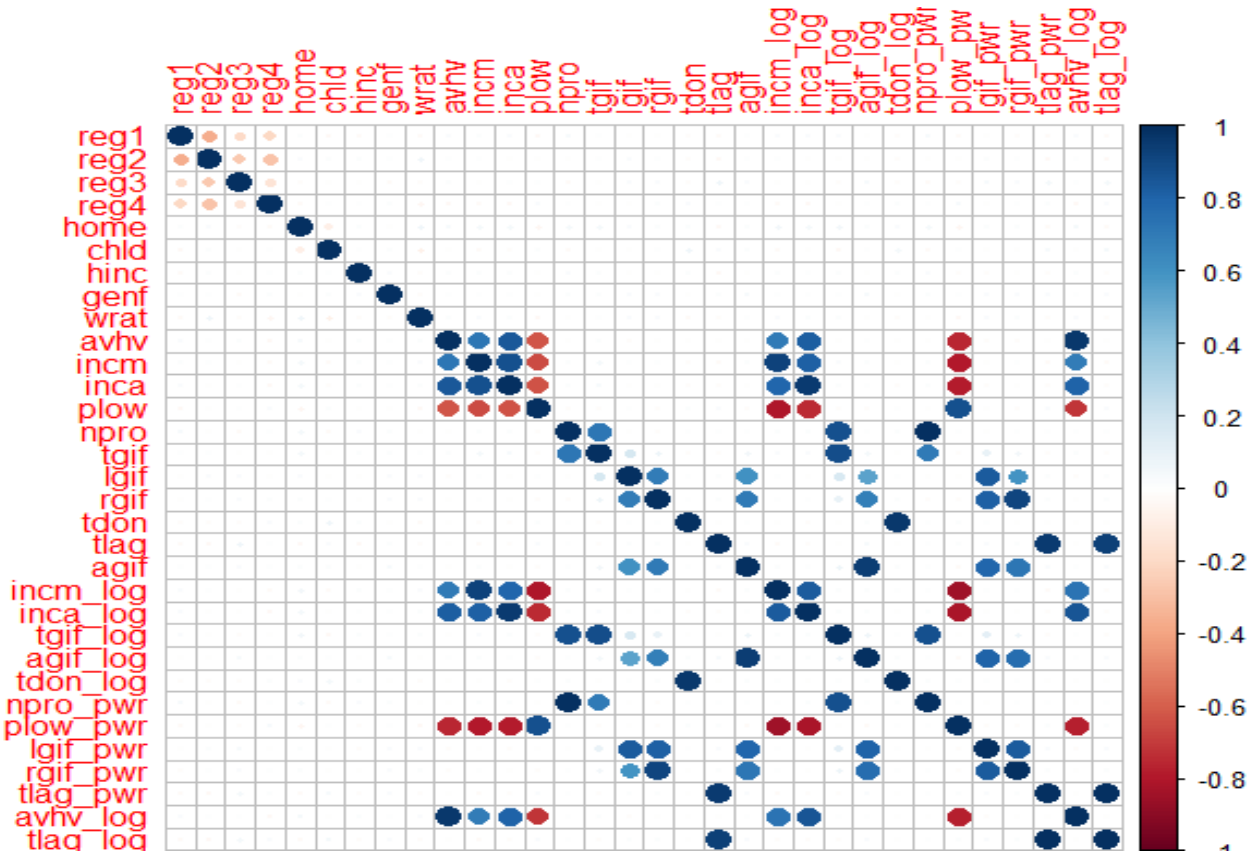
The next step in our EDA will be to look at the correlation matrix for the training data-set.

We note that some variables are close to having significant correlations based on the above plot. As such, certain 'key' variables will not be excluded through the model building analysis that we will perform next. The variables are listed in table 5 below.

Variable Y Axis	Variable X Axis
INCM	AVHV
INCA	INCM
PLOW	INCA
TGIF	NPRO
RGIF	LGIF
INCA_LOG	INCM_LOG
RGIF_PWR	LGIF_PWR
TLAG_LOG	TLAG_PWR

Table 5: Corrplot Variables

The package corrplot provides a plot of the correlation matrix for the training data set in order to determine if there is multicollinearity. While some variables are close to having significant correlations, based upon the results from plot 1, the variables in listed table 5, will not be excluded from the model building.



Plot 1: Corrplot

The correlation diagram identifies significant positive correlations between predictor variables. These relationships will be considered in the model building process and additional tests of multicollinearity will be applied where applicable.

Variable Y Axis	Variable X Axis
INCM	AVHV
INCA	INCM
PLOW	INCA
TGIF	NPRO
RGIF	LGIF
INCA_LOG	INCM_LOG
RGIF_PWR	LGIF_PWR
TLAG_LOG	TLAG_PWR

Table 5: Corrplot Variables

Classification Training Model Development

The type of machine learning models developed for the training analysis are known as classification models. The following nine models will be utilized for classification model development: LDA, Logistic Regression, KNN, Decision Tree, Bagging, Random Forest, Boosting, SVM, and Stack or Ensemble models. We will perform analysis on each model and present key information in our findings below, including the respective Mailings, Profit and Accuracy against each model. We will further present the Classification Table of each model to get a better understanding on how the model performs.

LDA Model :- The LDA model presented below was not considered the best model when using qualitative predictors; however, this model is being utilized as a base model for comparison. This model achieved the third ranked highest profit with the fourth ranked lowest prediction error. The following variables will be utilized for the LDA model: reg1, reg2, home, plow, npro, tdon, tlag, incm_log, tgif_log, tdon_log, npro_pwr, tlag_pwr, tlag_log, as well as dummy variables created for each class of chld, hinc, and wrat. Our final results tab depicts the reason why this model as well as other models were not selected. The criteria we are working towards is a model that has the highest profit values.

- **Mailings:** **1,245**
- **Profit:** **$11,865**
- **Accuracy:** **87%**
- **Sensitivity** **80%**

Classification Table			
chat.valid.lda		**0**	**1**
	0	764	9
	1	255	990

Table 6: LDA Classification Table

Logistic Regression Model:- The logistic regression model is suited well for highlighting variables with significance based upon their p-values. This model achieved the first ranked highest profit with the second ranked lowest prediction error. The following variables will be utilized for the logistic model: reg1, reg2, home, plow, npro, tdon, I(tdon^3), tlag, incm_log, tgif_log, tdon_log, npro_pwr, tlag_pwr, tlag_log, as well as dummy variables created for each class of chld, hinc, and wrat. This model had a *AIC value of 1673.3*. The following variables demonstrated high significance based on their p-value.

Variable	P-Value
reg1	< 2e-16 ***
reg2	< 2e-16 ***
home	< 2e-16 ***
tdon	< 2e-16 ***
tlag	0.000461 ***
incm_log	6.57e-10 ***
tgif_log	0.000930 ***
tdon_log	< 2e-16 ***
npro_pwr	0.034301 *
tlag_pwr	8.37e-05 ***
tlag_log	0.000116 ***
factor(chld)	< 2e-16 ***
factor(hinc)	< 2e-16 ***
factor(wrat)	2.52e-15 ***

Table 7: Logistic Regression Variables

- **Mailings: 1,238**
- **Profit: $11,923**
- **Accuracy: 88%**
- **Sensitivity 80%**

Classification Table			
chat.valid.knn		**0**	**1**
	0	774	6
	1	245	993

Table 8: Logistic Classification Table

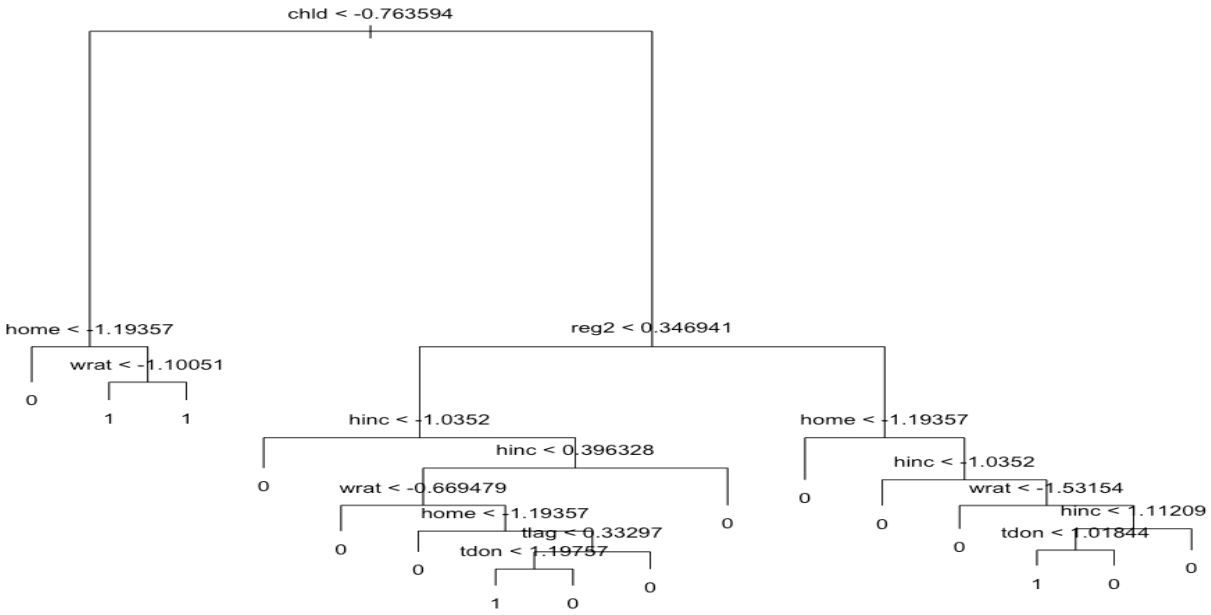
KNN Model:- The KNN model achieved the ninth ranked highest profit with the seventh ranked lowest prediction error. Cross validation was used to select five as a number of nearest neighbor values to consider for each decision point. The following variables will be utilized for the KNN model: reg1, reg2, home, avhv, incm, inca, npro, tdon, tlag, incm_log, inca_log, tgif_log, tdon_log, npro_pwr, tlag_pwr, avhv_log, tlag_log, chld, hinc, wrat.

- **Mailings: 1,189**
- **Profit: $10,035**
- **Accuracy: 83%**
- **Sensitivity 78%**

Classification Table			
chat.valid.knn		**0**	**1**
	0	758	75
	1	261	924

Table 9: KNN Classification Table

Decision Tree Model:- The decision tree model achieved the eighth ranked highest profit with the ninth ranked lowest prediction error. A large tree was fit first, and then pruned to a subtree with a smaller number of terminal nodes using the deviance produced from cross validation. The following variables will be utilized for the dec tree model: reg1, reg2, reg3, reg4, home, chld, hinc, genf, wrat, avhv_log, incm, inca, plow, npro, tgif, lgif, rgif, tdon, tlag, agif. The figure below shows that the variables chld, reg2, and home reduce a significant amount of node impurity and can be deemed as important.



- **Mailings: 1,362**
- **Profit: $11,414**
- **Accuracy: 81%**
- **Sensitivity 76%**

Classification Table			
chat.valid.tree		**0**	**1**
	0	667	32
	1	352	967

Table 10: DT Classification Table

Bagging Model:- The bagging model achieved the sixth ranked highest profit with the sixth ranked lowest prediction error. All predictor variables were considered at each split of the tree for each bootstrap sample to create the bagging model. The following variables will be utilized for the bagging model: reg1, reg2, reg3, reg4, home, chld, hinc, genf, wrat, avhv_log, incm, inca, plow, npro, tgif, lgif, rgif, tdon, tlag, agif.

- **Mailings: 1,309**
- **Profit: $11,708**
- **Accuracy: 84%**
- **Sensitivity 76%**

Classification Table			
chat.valid.bag		**0**	**1**
	0	699	13
	1	320	986

Table 11: Bagging Classification Table

Random Forest Model:- The random forest model achieved the fifth ranked highest profit with the third ranked lowest prediction error. The number of predictor variables for each split of the tree was set to 4 (p/3). The following variables will be utilized for the random forest model: reg1, reg2, reg3, reg4, home, chld, hinc, genf, wrat, avhv_log, incm, inca, plow, npro, tgif, lgif, rgif, tdon, tlag, agif.

- **Mailings: 1,223**
- **Profit: $11,764**
- **Accuracy: 87%**
- **Sensitivity 80%**

Classification Table			
chat.valid.rf		**0**	**1**
	0	777	20
	1	242	979

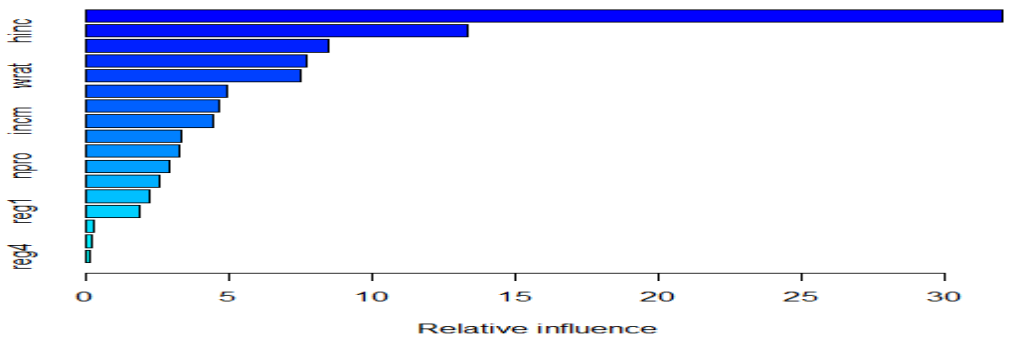
Table 12: RF Classification Table

Boosting Model:- The boosting model achieved the second ranked highest profit with the first ranked lowest prediction error. Cross validation was used to select the interaction depth and shrinkage of the model, which limits the depth of each tree and the rate at which the model learns. The model selected was rather complex with a interaction depth of 6 and a shrinkage parameter of 0.01. The following variables will be utilized for the boosting model: reg1, reg2, reg3, reg4, home, chld, hinc, genf, wrat, avhv, incm, inca, plow, npro, tgif, tdon, tlag.

- **Mailings: 1,215**
- **Profit: $11,911**
- **Accuracy: 88%**
- **Sensitivity 81%**

Classification Table			
chat.valid.boost		**0**	**1**
	0	793	10
	1	226	989

Table 13: Boosting Classification Table



Plot 2: Boosting Variable Influence

SVM Model:- The SVM model achieved the seventh ranked highest profit with the eighth ranked lowest prediction error. Several kernels were used to fit the data, with the radial kernel performing the best. Gamma and the 'cost' variable, which determines the number of support vectors used, were selected through cross validation methods. The following variables will be utilized for the SVM model: reg1, reg2, reg3, reg4, home, chld, hinc, genf, wrat, avhv, incm, inca, plow, npro, tgif, lgif, rgif, tdon, tlag, agif.

- **Mailings: 1,308**
- **Profit: $11,638**
- **Accuracy: 83%**
- **Sensitivity 75%**

Classification Table			
chat.valid.svm		**0**	**1**
	0	694	16
	1	325	983

Table 14: SVM Classification Table

Stack or Ensemble Model:- The results of three models (log, boosting, svm), one which had results that presented a weak correlation with other models, were combined to create a stack model. The stack model achieved the fourth ranked highest profit with the fifth ranked lowest prediction error.

- **Mailings: 1,253**
- **Profit: $11,864**
- **Accuracy: 87%**
- **Sensitivity 79%**

Classification Table			
chat.valid.stack		**0**	**1**
	0	757	8
	1	262	991

Table 15: Stack Classification Table

RESULTS:- Summary of Models

Below we will populate all the model based on their highest profit achieved as well as the lowest errors achieved. Lastly, we will present a combination model depiction based on validation set least error and greatest profit.

Models ranked by increased Profit on validation set

Rank	Model	N.Mail	Profit	Error
9	Log1	1238	11923	0.1244
4	Boosting	1215	11911	0.1169
8	LDA1	1245	11865	0.1308
1	Stack	1253	11864	0.1338
7	Random Forest	1223	11764	0.1298
5	Bagging	1309	11708	0.165
2	SVM	1308	11638	0.169
6	Decision Tree	1362	11414	0.1903
3	KNN	1189	11035	0.1665

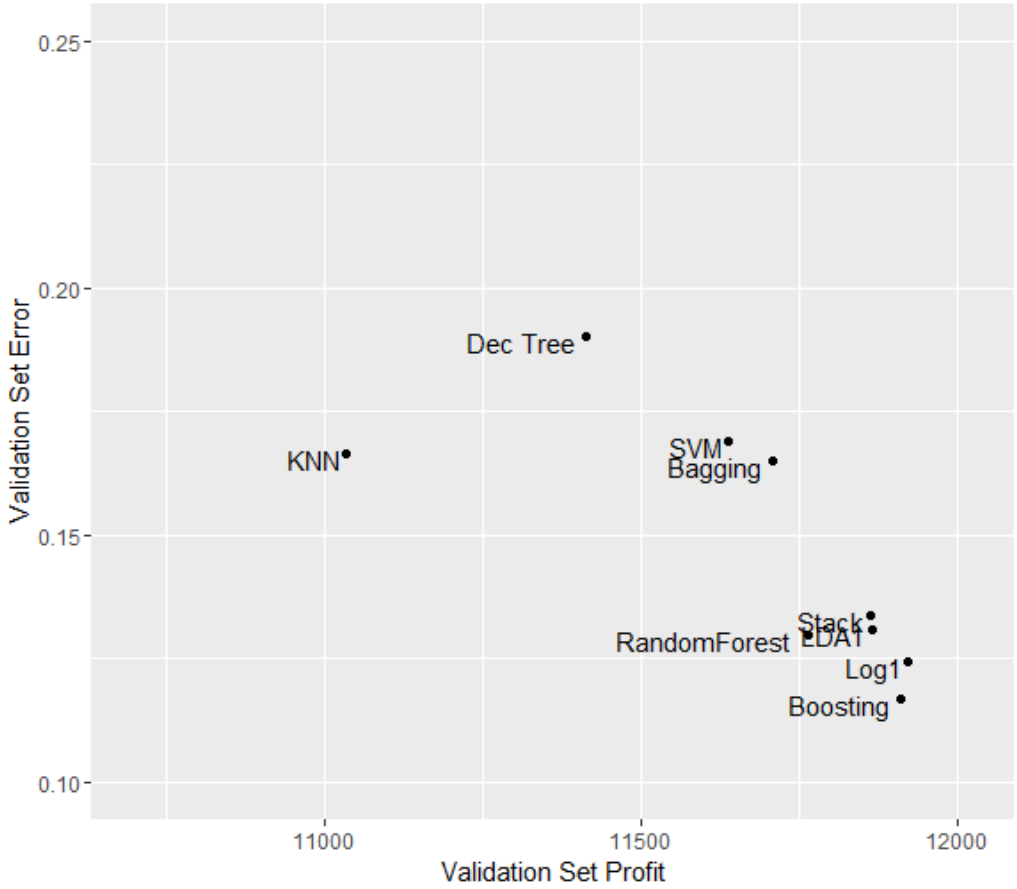
Table 16: Models by Profit

Models ranked by decreased error on validation set

Rank	Model	N.Mail	Profit	Error
4	Boosting	1215	11911	0.1169
9	Log1	1238	11923	0.1244
7	Random Forest	1223	11764	0.1298
8	LDA1	1245	11865	0.1308
1	Stack	1253	11864	0.1338
5	Bagging	1309	11708	0.165
3	KNN	1189	11035	0.1665
2	SVM	1308	11638	0.169
6	Decision Tree	1362	11414	0.1903

Table 17: Models by Error

Best combined model based upon validation set least error and greatest profit.



Plot 3: Best combined Model

The selected model based upon greatest profit achieved on the training and validation data sets is the logistic model, which indicates to mail to the 302 highest posterior probabilities. Within this model, we would generate a profit of 11,923.

Classification Table			
Chat.test		**0**	**1**
		1705	302

Table 18: Posterior Probabilities

ANALYSIS:- Predictive Model Development

Our next analysis will focus on building various prediction models for the DAMT variable using other variables as predictors. The following seven models will be utilized for predictive model development: least squares, ridge, lasso, PCR, GAM with non-linear splines, boosting, and random forest regression models.

Least Squares Model

The least squares regression predictive model achieved second ranked lowest mean prediction error of 1.3373 as shown in the summary table # 23. The final least squares model utilized the backwards step model, the models were cross validated using forward, backward, and stepwise variable selection.

Variable	GVIF	DF	GVIF^(1/(2*Df))
REG3	1.0622	1	1.0306
REG4	1.0621	1	1.0306
HOME	1.0432	1	1.0213
GENF	1.0133	1	1.0066
PLOW	4.4128	1	2.1007
LGIF	5.2735	1	2.2964
RGIF	8.8556	1	2.9758
TDON	1.0182	1	1.0090
AGIF	11.4014	1	3.3766
INCM_LOG	3.6803	1	1.9184
TGIF_LOG	1.0726	1	1.0356
AGIF_LOG	13.5878	1	3.6861
PLOW_PWR	5.5447	1	2.3547
LGIF_PWR	11.4015	1	3.3766
RGIF_PWR	11.2518	1	3.3543
Factor(CHLD)	1.3065	5	1.0270
Factor(HINC)	1.2043	6	1.0156
Factor(WRAT)	1.1413	9	1.0073

Table 19: VIF

Ridge Model

The ridge predictive model achieved fifth ranked lowest mean prediction error of 1.4418 as shown in the summary table # 23. An important benefit of using the ridge model is that it automatically transforms any qualitative variables into dummy variables. The largest lambda within one standard error of the minimum was explored but had a higher validation error than the best lambda. Additional significant measures and results are shown below in table 20.

Measure	Result
Best Lambda	0.1224564
Best Mean	1.441831
Largest Lambda	0.8639055
Largest Mean	1.754513

Table 20: Ridge Model

Lasso Model

The lasso predictive model achieved third ranked lowest mean prediction error of 1.3435 as shown in the summary table # 23. The lasso model performed variable selection by shrinking variable coefficients to zero or near zero. The largest lambda within one standard error of the minimum was explored but had a higher validation error than the best lambda. Additional significant measures and results are shown below in table 21.

Measure	Result
Best Lambda	0.002895468
Best Mean	1.399952
Largest Lambda	0.0751381
Largest Mean	1.700953

Table 21: Lasso Model

PCR Model

The principal component regression model achieved fourth ranked lowest mean prediction error of 1.3993 as shown in the summary table # 23. MSE was used as a metric to select the a number of principal components in the model (38).

Non-Linear Spline Model

The predictive GAM using non-linear natural splines as basis functions achieved first ranked lowest mean prediction error of 1.3127 as shown in the summary table # 23. Additional significant measures and results are shown below in table 22.

Measure	Result
MPE	1.312746
Adj-R^2	0.6897

Table 22: Spline Model

Boosting Model

The Boosting predictive model achieved sixth ranked lowest mean prediction error of 1.4771 as shown in the summary table # 23. Cross validation was leveraged to select the interaction depth of 7 and shrinkage value of 0.01 in the model.

Random Forest Model

The random forest predictive model achieved seventh ranked lowest mean prediction error of 1.6201 as shown in the summary table # 23.

RESULTS

The best predictive model selected the non-linear spline regression model based upon the lowest mean prediction error of 1.3127 and an adjusted-R^2 of 0.6897.

Model #	Model	MPE	STD Error	Coefficients
3	Spline	1.3127	0.1527	45
7	LS1	1.3373	0.1552	36
5	Lasso	1.3435	0.1545	43
4	PCR	1.3993	0.1559	NA
6	Ridge	1.4418	0.1580	54
2	Boost	1.4771	0.1621	14
1	Random Forest	1.6201	0.1713	14

Table 23: Predictive Model Summary

The logistic regression classification in conjunction with the GAM using non-linear natural splines predicted a profit of $3,704 on the test set, through mailing to 301 potential donors as shown in table 25, and having a mean expected donation of $13.42 as shown in table 25.

Classification Table			
Chat.test		**0**	**1**
		1706	301

Table 24: Posterior Probabilities

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
6.354	12.130	13.410	13.420	14.750	19.280

Table 25: Donation Summary

CONCLUSION

The logistic regression classification model and the GAM using non-linear natural splines demonstrated the best performance in this machine learning model to improve the cost-effectiveness, (profit) of their direct marketing campaigns to their previous donors. A key takeaway is that when developing classification models to predict profit, key in on sensitivity, because false positives are problematic. An area for potential classification and predictive model improvement may involve in further refinement of the exploratory data analysis. Although a substantial amount of the analysis was allocated towards EDA, there is always room for improvement. One cautionary note is to not over-manipulate the source data which could lead to a greater potential for over-fitting the model.

APPENDIX

R Code :-

```
# Course Project - Working R Script File
# OBJECTIVE: A charitable organization wishes to develop a machine learning
# model to improve the cost-effectiveness of their direct marketing campaigns
# to previous donors. - test
# 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, we will have to each add our own code here
charity <- read.csv("/Users/charity.csv") # load the "charity.csv" file
# charity <- read.csv("/Users/charity.csv")
#transformation summary
par(mfcol=c(1,2))
hist(charity$avhv)
hist(log(charity$avhv))
hist(charity$incm)
hist(log(charity$incm))
hist(charity$plow)
hist(log(charity$plow))
hist(charity$inca)
hist(log(charity$inca))
hist(charity$tgif)
hist(log(charity$tgif))
hist(charity$agif)
hist(log(charity$agif))
hist(charity$rgif)
hist(log(charity$rgif))
hist(charity$tdon)
hist(log(charity$tdon))
hist(charity$tlag)
hist(log(charity$tlag))
hist(charity$plow)
hist(log(charity$plow))
hist(charity$npro)
hist((charity$npro)^(2/3))  #this one may not be worthwhile
hist(charity$plow)
```

```r
hist(charity$plow^(1/3)) #this does not look normal, but definitely better
hist(charity$lgif)
hist(charity$lgif^(1/5))
hist(charity$rgif)
hist(charity$rgif^(1/7))
hist(charity$hinc)
hist(charity$hinc^(2))
hist(charity$tlag)
hist(charity$tlag^(1/5))
par(mfcol=c(1,1))
# predictor transformations
charity.t <- charity
# charity.t$avhv <- log(charity.t$avhv)
# add further transformations if desired
# for example, some statistical methods can struggle when predictors are highly skewed
#these are based on the histograms above
charity.t$incm_log <- log(charity.t$incm)
charity.t$inca_log <- log(charity.t$inca)
charity.t$tgif_log <- log(charity.t$tgif)
charity.t$agif_log <- log(charity.t$agif)
charity.t$tdon_log <- log(charity.t$tdon)
charity.t$npro_pwr <- (charity.t$npro)^(2/3)
charity.t$plow_pwr <- charity.t$plow^(1/3)
charity.t$lgif_pwr <- charity.t$lgif^(1/5)
charity.t$rgif_pwr <- charity.t$rgif^(1/7)
charity.t$tlag_pwr <- charity.t$tlag^(1/5)
charity.t$avhv_log <- log(charity.t$avhv)
charity.t$tlag_log <- log(charity.t$tlag)
######## Interesting Tables for EDA ########
# Please feel free to use/add at your discretion
# no need to include all #
(table(charity.t$donr,charity.t$chld))
hist(charity.t$chld[charity.t$donr==1])
(table(charity.t$donr,charity.t$hinc>mean(charity.t$hinc)))
(table(charity.t$donr,charity.t$hinc))
hist(charity.t$hinc[charity.t$donr==1])
hist(charity.t$hinc[charity.t$donr==0])
(table(charity.t$donr,charity.t$wrat))
```

```r
hist(charity.t$wrat[charity.t$donr==1])
hist(charity.t$wrat[charity.t$donr==0])
# set up data for analysis
# added tranformed variables into the data set by replacing 2;21 with c(2:21,25:36)
data.train <- charity.t[charity$part=="train",]
x.train <- data.train[,c(2:21,25:36)]
# x.train <- data.train[,2:21]
c.train <- data.train[,22] # donr
n.train.c <- length(c.train) # 3984
y.train <- data.train[c.train==1,23] # damt for observations with donr=1
n.train.y <- length(y.train) # 1995
data.valid <- charity.t[charity$part=="valid",]
x.valid <- data.valid[,c(2:21,25:36)]
# x.valid <- data.valid[,2:21]
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.t[charity$part=="test",]
n.test <- dim(data.test)[1] # 2007
x.test <- data.test[,c(2:21,25:36)]
# x.test <- data.test[,2:21]
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) # Standardized data to classify donr
data.train.std.y <- data.frame(x.train.std[c.train==1,], damt=y.train) #  Standardized data to predict
damt when donr=1
#Validation set
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)
#check correlations
library(corrplot)
```

```
corrplot::corrplot(cor(x.train))
##### CLASSIFICATION MODELING ######
####################
# LDA 1
####################
library(MASS)
model.lda1 <- lda(donr ~ reg1 + reg2 + home + plow + npro + tdon + tlag + incm_log +
                    tgif_log + tdon_log + npro_pwr + tlag_pwr + tlag_log + factor(chld) +
                    factor(hinc) + factor(wrat),
                  data.train.std.c)
# 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
error.lda1 <- round(mean(chat.valid.lda1!=c.valid),4)
####################
# Logestic Reg
####################
model.log1 <- glm(donr ~ reg1 + reg2 + home + plow + npro +
                    tdon +
                    I(tdon^3)+
                    tlag + incm_log +
                    tgif_log +
                    tdon_log +
                    npro_pwr + tlag_pwr + tlag_log +
                    factor(chld) + factor(hinc) + factor(wrat)  ,
                  data.train.std.c, family=binomial("logit"))
summary(model.log1)
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
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
error.log1 <- round(mean(chat.valid.log1!=c.valid),4)
####################
# KNN
####################
#creating prediction and response matricies for the knn model
knn.data.train.std.c<-
as.matrix((data.train.std.c[,c("reg1","reg2","home","avhv","incm","inca","npro","tdon","tlag","incm_lo
g","inca_log",
                                                    "tgif_log","tdon_log","npro_pwr","tlag_pwr",
                                                    "avhv_log","tlag_log","chld","hinc","wrat")]))
knn.data.valid.std.c<-
as.matrix(data.valid.std.c[,c("reg1","reg2","home","avhv","incm","inca","npro","tdon","tlag","incm_log
","inca_log",
                                                   "tgif_log","tdon_log","npro_pwr","tlag_pwr",
                                                   "avhv_log","tlag_log","chld","hinc","wrat")])
knn.data.train.std.c.target<-data.train.std.c[,33]
library(class)
set.seed(1)
# knn.pred.cv=knn.cv(data.train.std.c[,-33],data.train.std.c[,33],k=3,prob=TRUE)
for (i in 1:10)
{
  knn.pred=knn(knn.data.train.std.c,knn.data.valid.std.c,knn.data.train.std.c.target,k=i,prob=TRUE)
  success<-(mean(knn.pred==c.valid))
  print(success)
}
knn.pred=knn(knn.data.train.std.c,knn.data.valid.std.c,knn.data.train.std.c.target,k=5,prob=TRUE)
#confusion matrix
table(knn.pred,c.valid)
mean(knn.pred==c.valid)
knn.attributes<-attributes(knn.pred)
knn.pred.num<-as.numeric(knn.pred)-1
```

```
post.valid.knn<-knn.attributes$prob*knn.pred.num
profit.knn <- cumsum(14.5*c.valid[order(post.valid.knn, decreasing=T)]-2)
plot(profit.knn)
n.mail.valid <- which.max(profit.knn) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.knn)) # report number of mailings and maximum profit
n.mail.valid
max(profit.knn)
cutoff.knn <- sort(post.valid.knn, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid
chat.valid.knn <- ifelse(post.valid.knn>cutoff.knn, 1, 0) # mail to everyone above the cutoff
table(knn.pred, c.valid) # classification table
table(chat.valid.knn, c.valid) # sanity check
error.knn <- round(mean(chat.valid.knn!=c.valid),4)
####################
# Dec Tree Model
####################
library(tree)
tree.fit=tree(factor(data.train.std.c$donr) ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf +
wrat +
                avhv_log + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,
              data=data.train.std.c)
plot(tree.fit)
text(tree.fit,pretty=1)
cv.tree.fit=cv.tree(tree.fit,FUN=prune.misclass)
names(cv.tree.fit)
cv.tree.fit
par(mfrow=c(1,2))
plot(cv.tree.fit$size,cv.tree.fit$dev,type="b")
plot(cv.tree.fit$k,cv.tree.fit$dev,type="b")
# We now apply the prune.misclass() function in order to prune the tree to the nodes with the lowest
dev
lowest.dev.node<-cv.tree.fit$size[which.min(cv.tree.fit$dev)]
prune.tree=prune.misclass(tree.fit,best=lowest.dev.node)
plot(prune.tree)
text(prune.tree,pretty=1)
post.valid.tree <- predict(prune.tree, data.valid.std.c, type="vector")[,2] # n.valid post probs
profit.tree <- cumsum(14.5*c.valid[order(post.valid.tree, decreasing=T)]-2)
plot(profit.tree) # see how profits change as more mailings are made
n.mail.valid <- which.max(profit.tree) # number of mailings that maximizes profits
```

```
c(n.mail.valid, max(profit.tree)) # report number of mailings and maximum profit
n.mail.valid
max(profit.tree)
cutoff.tree <- sort(post.valid.tree, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid
chat.valid.tree <- ifelse(post.valid.tree>cutoff.tree, 1, 0) # mail to everyone above the cutoff
table(chat.valid.tree, c.valid) # classification table
error.tree <- round(mean(chat.valid.tree!=c.valid),4)
####################
# Bagging Model
####################
model.bag <- randomForest::randomForest(factor(data.train.std.c$donr) ~ reg1 + reg2 + reg3 + reg4 +
home + chld + hinc + genf + wrat +
                                          avhv_log + incm + inca + plow + npro + tgif + lgif + rgif +
tdon + tlag + agif,
                                        data.train.std.c,mtry=20,importance=TRUE)
randomForest::importance(model.bag)
randomForest::varImpPlot(model.bag)
post.valid.bag <- predict(model.bag, data.valid.std.c,type='prob')[,2] # n.valid.c post probs
# calculate ordered profit function using average donation = $14.50 and mailing cost = $2
profit.bag <- cumsum(14.5*c.valid[order(post.valid.bag, decreasing=T)]-2)
plot(profit.bag) # see how profits change as more mailings are made
n.mail.valid <- which.max(profit.bag) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.bag)) # report number of mailings and maximum profit
cutoff.bag <- sort(post.valid.bag, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid
chat.valid.bag <- ifelse(post.valid.bag>cutoff.bag, 1, 0) # mail to everyone above the cutoff
table(chat.valid.bag, c.valid) # classification table
error.bag <- round(mean(chat.valid.bag!=c.valid),4)
####################
# Random Forest Model
####################
set.seed(3)
model.rf <- randomForest::randomForest(factor(data.train.std.c$donr) ~ reg1 + reg2 + reg3 + reg4 +
home + chld + hinc + genf + wrat +
                                         avhv_log + incm + inca + plow + npro + tgif + lgif + rgif +
tdon + tlag + agif,
                                       data.train.std.c)
model.rf
#Var importance stats and plot
```

```
randomForest::importance(model.rf)
randomForest::varImpPlot(model.rf)
post.valid.rf <- predict(model.rf, data.valid.std.c,type='prob')[,2] # n.valid.c post probs
# calculate ordered profit function using average donation = $14.50 and mailing cost = $2
profit.rf <- cumsum(14.5*c.valid[order(post.valid.rf, decreasing=T)]-2)
plot(profit.rf) # see how profits change as more mailings are made
n.mail.valid <- which.max(profit.rf) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.rf)) # report number of mailings and maximum profit
cutoff.rf <- sort(post.valid.rf, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid
chat.valid.rf <- ifelse(post.valid.rf>cutoff.rf, 1, 0) # mail to everyone above the cutoff
table(chat.valid.rf, c.valid) # classification table
error.rf <- round(mean(chat.valid.rf!=c.valid),4)
####################
# Boosting Model
####################
library(gbm)
# gbm() with the option distribution="gaussian" for a regression problem;
# gbm() for a classification problem, we would use distribution="bernoulli".
# The argument n.trees=5000 indicates that we want 5000 trees, and the option interaction.depth=x
limits the depth of each tree.
#Using this code to tune the GBM model.
#Commented out due to the time it takes to run the code
# library(caret)
# myTuneGrid <- expand.grid(n.trees = 500,interaction.depth = c(6,7),shrinkage =
c(.001,.01,.1),n.minobsinnode=10)
# fitControl <- trainControl(method = "repeatedcv", number = 3,repeats = 1, verboseIter =
FALSE,returnResamp = "all")
# myModel <- train(data.train.std.c$donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf +
wrat +
#                    avhv + incm + inca + plow + npro + tgif  + tdon + tlag ,
#                  data=data.train.std.c,method = "gbm",trControl = fitControl,tuneGrid = myTuneGrid)
set.seed(1)
model.boost=gbm(data.train.std.c$donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +
                  avhv + incm + inca + plow + npro + tgif  + tdon + tlag ,

data=data.train.std.c,distribution="bernoulli",n.trees=5000,interaction.depth=6,shrinkage = .01)
# The summary() function produces a relative influence plot and also outputs the relative influence
statistics.
```

```
summary(model.boost)
post.valid.boost <- predict(model.boost, data.valid.std.c,type='response',n.trees=5000) # n.valid.c
post probs
# calculate ordered profit function using average donation = $14.50 and mailing cost = $2
profit.boost <- cumsum(14.5*c.valid[order(post.valid.boost, decreasing=T)]-2)
plot(profit.boost) # see how profits change as more mailings are made
n.mail.valid <- which.max(profit.boost) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.boost)) # report number of mailings and maximum profit
cutoff.boost <- sort(post.valid.boost, decreasing=T)[n.mail.valid+1] # set cutoff based on
n.mail.valid
chat.valid.boost <- ifelse(post.valid.boost>cutoff.boost, 1, 0) # mail to everyone above the cutoff
table(chat.valid.boost, c.valid) # classification table
error.boost <- round(mean(chat.valid.boost!=c.valid),4)
####################
# SVM
####################
library(e1071)
svmfit=svm(as.factor(data.train.std.c$donr) ~ reg1 + reg2 + reg3 + reg4 + home + factor(chld) + hinc +
genf + wrat +
             avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,
           data=data.train.std.c, kernel="radial",  gamma=.03125, cost=10,probability =TRUE)
summary(svmfit)
#use this code to tune SVM using cross validation.
#commented out due to the time it requires to run # set.seed(1)
# myTuneGrid <- expand.grid(sigma=2^c(-25,-5,-1),C=10)
# fitControl <- trainControl(method = "cv", number = 5,repeats = 1, verboseIter = FALSE,returnResamp =
"all",classProbs = TRUE)
# myModel <- train(as.factor(data.train.std.c$donr) ~ reg1 + reg2 + reg3 + reg4 + home + factor(chld)
+ hinc + genf + wrat +
#                    avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,
#                  data=data.train.std.c , method = "svmRadial",trControl = fitControl,tuneGrid =
myTuneGrid)
pred<-predict(svmfit, data.valid.std.c,probability = TRUE)
post.valid.svm <- attr(pred, "probabilities")[,2] # n.valid.c post probs
profit.svm <- cumsum(14.5*c.valid[order(post.valid.svm, decreasing=T)]-2)
plot(profit.svm) # see how profits change as more mailings are made
n.mail.valid <- which.max(profit.svm) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.svm)) # report number of mailings and maximum profit
```

```
cutoff.svm <- sort(post.valid.svm, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid
chat.valid.svm <- ifelse(post.valid.svm>cutoff.svm, 1, 0) # mail to everyone above the cutoff
table(chat.valid.svm, c.valid) # classification table
error.svm <- round(mean(chat.valid.svm!=c.valid),4)
####################
# Stack
####################
stack.df<-DF <- data.frame(log1=as.numeric(post.valid.log1),  # as many cols as you need
                           stringsAsFactors=FALSE)
stack.df<-cbind("RandomForest"=as.numeric(post.valid.rf),stack.df)
stack.df<-cbind("Dec Tree"=as.numeric(post.valid.tree),stack.df)
stack.df<-cbind("Bagging"=as.numeric(post.valid.bag),stack.df)
stack.df<-cbind("Boosting"=as.numeric(post.valid.boost),stack.df)
stack.df<-cbind("KNN"=as.numeric(post.valid.knn),stack.df)
stack.df<-cbind("SVM"=as.numeric(post.valid.svm),stack.df)
stack.df<-cbind("log1"=as.numeric(post.valid.log1),stack.df)
head(stack.df)
stack.df.subset<-(stack.df[,c("log1","Boosting", "SVM")])
post.valid.stack<-rowMeans(stack.df.subset)
# post.valid.stack<-Reduce(`*`, stack.df.subset)
profit.stack <- cumsum(14.5*c.valid[order(post.valid.stack, decreasing=T)]-2)
plot(profit.stack) # see how profits change as more mailings are made
n.mail.valid <- which.max(profit.stack) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.stack)) # report number of mailings and maximum profit
cutoff.stack <- sort(post.valid.stack, decreasing=T)[n.mail.valid+1] # set cutoff based on
n.mail.valid
chat.valid.stack <- ifelse(post.valid.stack>cutoff.stack, 1, 0) # mail to everyone above the cutoff
table(chat.valid.stack, c.valid) # classification table
error.stack <- round(mean(chat.valid.stack!=c.valid),4)
#################
#Results DataFrame
#################
model<-c("LDA1","Log1")
n.mail<-c(which.max(profit.lda1),which.max(profit.log1))
profit<-c(max(profit.lda1),max(profit.log1))
error<-c(error.lda1,error.log1)
results<-data.frame(model,n.mail,profit,error,stringsAsFactors = FALSE)
# adding results for each new model
```

```
results<-rbind(c("RandomForest",which.max(profit.rf),max(profit.rf),error.rf),results)
results<-rbind(c("Dec Tree",which.max(profit.tree),max(profit.tree),error.tree),results)
results<-rbind(c("Bagging",which.max(profit.bag),max(profit.bag),error.bag),results)
results<-rbind(c("Boosting",which.max(profit.boost),max(profit.boost),error.boost),results)
results<-rbind(c("KNN",which.max(profit.knn),max(profit.knn),error.knn),results)
results<-rbind(c("SVM",which.max(profit.svm),max(profit.svm),error.svm),results)
results<-rbind(c("Stack",which.max(profit.stack),max(profit.stack),error.stack),results)
#Models ranked by Profit on validation set
results[order(results$profit,decreasing = TRUE),]
#Models ranked by error on validation set
results[order(results$error,decreasing = FALSE),]
dev.off()
require(ggplot2)
a<-ggplot(data=results, aes(as.numeric(results$profit),as.numeric(results$error))) +
  geom_point() + geom_text(aes(label=results$model), vjust=.75,hjust=1.1) + ylim (.10,.25) +
xlim(10700,12025) +xlab("Validation Set Profit") +
  ylab("Validation Set Error")
a
#Change these 2 variables to whichever model we choose.
# select model.log1 since it has maximum profit in the validation sample
best.model<-model.log1
best.model.profit<-profit.log1
post.test <- predict(best.model, data.test.std, type="response") # post probs for test data
# Oversampling adjustment for calculating number of mailings for test set
n.mail.valid <- which.max(best.model.profit)
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
# 1705  302
# based on this model we'll mail to the 302 highest posterior probabilities
# See below for saving chat.test into a file for submission
```

```
################################
##### PREDICTION MODELING ######
################################
################
# Least squares regression
################
fullmod<-lm(damt ~ . + I(hinc^2)+ factor(chld)+factor(hinc)+factor(wrat),
            data.train.std.y)
# Using the step function below to perform backward variable selection
backwards_lm = step(fullmod,trace=0)
forward_lm = step(fullmod,trace=0,direction = "forward")
step_lm = step(fullmod,trace=0,direction = "both")
formula(backwards_lm)
formula(forward_lm)
formula(step_lm)
#Automated variable selection
# Commented out due to the time it takes to run
# model.backward <- lm(formula(backwards_lm),
#                 data.train.std.y)
# model.forward <- lm(formula(forward_lm),
#                     data.train.std.y)
# model.step <- lm(formula(step_lm),
#                     data.train.std.y)
#Can use this code for cross validation to compare models
#comment out for now, feel free to test.
# #K-fold cross validation
# k=5
# set.seed(1)
# #seperates the data into k folds
# folds=sample(1:k,nrow(data.train.std.y),replace=TRUE)
#
# #loop that performs cross-validation
# backward.cv.errors=matrix(NA,k,1)
# for(j in 1:k){
#     pred=predict(model.backward,data.train.std.y[folds==j,])
#     backward.cv.errors[j]=mean((data.train.std.y$damt[folds==j]-pred)^2)
# }
# mean(backward.cv.errors)
```

```r
#Backward/step model
model.ls1 <- lm(damt ~ reg3 + reg4 + home + genf + plow + lgif + rgif + tdon +
                  agif + incm_log + tgif_log + agif_log + plow_pwr + lgif_pwr +
                  rgif_pwr + factor(chld) + factor(hinc) + factor(wrat),
                data.train.std.y)
pred.valid.ls1 <- predict(model.ls1, newdata = data.valid.std.y) # validation predictions
mpe.ls1<-mean((y.valid - pred.valid.ls1)^2) # mean prediction error
std.error.ls1<-sd((y.valid - pred.valid.ls1)^2)/sqrt(n.valid.y) # std error
ls1.num.coef<-length(model.ls1$coefficients)
#evaluate multi-collinearity
library(car)
vif(model.ls1)
#################
# Ridge Regression
#################
# The model.matrix() function is particularly useful for creating x; not only
# does it produce a matrix corresponding to the predictors but it also
# automatically transforms any qualitative variables into dummy variables
x.matrix.train=model.matrix(damt ~ . + I(hinc^2)+ factor(chld)+factor(hinc)+factor(wrat),
                            data.train.std.y)
y.matrix.train=data.train.std.y$damt
x.matrix.valid=model.matrix(damt ~ . + I(hinc^2)+ factor(chld)+factor(hinc)+factor(wrat),
                            data.valid.std.y)
y.matrix.valid<-data.valid.std.y$damt
#glmnet package
library(glmnet)
# By default the glmnet() function performs ridge regression for an automatically
# selected range of lambda values. However, here we have chosen to implement
# the function over a grid of values ranging from lambda = 10^10 to lambda = 10e2, essentially
# covering the full range of scenarios from the null model containing
# only the intercept, to the least squares fit.
grid=10^seq(10,-2,length=100)
# #run ridge regression with lambda
# Note that by default, the glmnet() function standardizes the
# variables so that they are on the same scale. To turn off this default setting,
# use the argument standardize=FALSE
ridge.mod=glmnet(x.matrix.train,y.matrix.train,alpha=0,lambda=grid,standardize = FALSE)
plot(ridge.mod, xvar="lambda", label=T)
```

```
#matrix with 55 rows (one for each predictor, plus an intercept) and 100
# columns (one for each value of lambda).
dim(coef(ridge.mod))
# cross-validation to choose the tuning parameter lambda. We can do this using
# the built-in cross-validation function, cv.glmnet(). By default, the function cv.glmnet() performs
ten-fold cross-validation,
#though this can be changed using the argument nfolds
set.seed(1)
cv.out=cv.glmnet(x.matrix.train,y.matrix.train,alpha=0)
plot(cv.out)
#identifying the best tuning parameter lambda
bestlam=cv.out$lambda.min
bestlam
#Test MSE for the best lambda
ridge.pred.best=predict(ridge.mod,s=bestlam,newx=(x.matrix.valid))
mean((ridge.pred.best-y.matrix.valid)^2)
#lagest lambda within 1 sd
largestlam=cv.out$lambda.1se
largestlam
#Test MSE for the largest lambda to see if we get lower error
ridge.pred.largest=predict(ridge.mod,s=largestlam,newx=(x.matrix.valid))
mean((ridge.pred.largest-y.matrix.valid)^2)
#fitting the ridge model with best lambda
ridge.mod=glmnet(x.matrix.train,y.matrix.train,alpha=0,lambda=bestlam,standardize = FALSE)
pred.valid.ridge <- predict(ridge.mod,newx=(x.matrix.valid)) # validation predictions
mpe.ridge<-mean((y.valid - pred.valid.ridge)^2) # mean prediction error
std.error.ridge<-sd((y.valid - pred.valid.ridge)^2)/sqrt(n.valid.y) # std error
ridge.coef <- predict(ridge.mod,newx=(x.matrix.valid),type = "coefficients")
ridge.num.coef<-sum(ridge.coef!=0)
#################
# Lasso
#################
#alpha=1 for lasso model
lasso.mod=glmnet(x.matrix.train,y.matrix.train,alpha=1,lambda=grid,standardize = FALSE)
plot(lasso.mod)
# perform cross-validation and compute the associated test error.
set.seed(1)
cv.out=cv.glmnet(x.matrix.train,y.matrix.train,alpha=1)
```

```
plot(cv.out)
# Best Lamda
bestlam=cv.out$lambda.min
bestlam
lasso.pred=predict(lasso.mod,s=bestlam,newx=(x.matrix.valid))
mean((lasso.pred-y.matrix.valid)^2)
#Largest lam
largestlam=cv.out$lambda.1se
largestlam
lasso.pred=predict(lasso.mod,s=largestlam,newx=(x.matrix.valid))
mean((lasso.pred-y.matrix.valid)^2)
#fitting the lasso model with best lambda
lasso.mod=glmnet(x.matrix.train,y.matrix.train,alpha=1,lambda=bestlam,standardize = FALSE)
pred.valid.lasso <- predict(lasso.mod,newx=(x.matrix.valid)) # validation predictions
mpe.lasso<-mean((y.valid - pred.valid.lasso)^2) # mean prediction error
std.error.lasso<-sd((y.valid - pred.valid.lasso)^2)/sqrt(n.valid.y) # std error
lasso.coef <- predict(lasso.mod,newx=(x.matrix.valid),type = "coefficients")
lasso.num.coef<-sum(lasso.coef!=0)
#################
# PCR
#################
library(pls)
set.seed(2)
pcr.fit=pcr(damt ~ . + I(hinc^2)+ factor(chld)+factor(hinc)+factor(wrat),
data=data.train.std.y,scale=TRUE,validation="CV")
# Setting scale=TRUE has the effect of standardizing each
# predictor, so that the scale on which each variable is measured will not have an effect. Setting
# validation="CV" causes pcr() to compute the ten-fold cross-validation error
# for each possible value of M, the number of principal components used.
summary(pcr.fit)
validationplot(pcr.fit,val.type="MSEP")
#MSE values
MSEP(pcr.fit)
#MSE for 38
pred.valid.pcr=predict(pcr.fit,data.valid.std.y,ncomp=38)
mean((pred.valid.pcr-y.valid)^2)
mpe.pcr<-mean((y.valid - pred.valid.pcr)^2) # mean prediction error
std.error.pcr<-sd((y.valid - pred.valid.pcr)^2)/sqrt(n.valid.y) # std error
```

```
# plot(pcr.fit, ncomp=1:38)
#################
# Non Linear Spline
#################
#fit a GAM to predict wage using natural spline functions
fit.incm_log<-smooth.spline(x=data.train.std.y$incm_log,y=data.train.std.y$damt,cv=TRUE)
plot(fit.incm_log)
fit.incm_log
fit.tgif_log<-smooth.spline(x=data.train.std.y$tgif_log,y=data.train.std.y$damt,cv=TRUE)
fit.tgif_log
plot(fit.tgif_log)
fit.plow_pwr<-smooth.spline(x=data.train.std.y$plow_pwr,y=data.train.std.y$damt,cv=TRUE)
fit.plow_pwr
plot(fit.plow_pwr)
library(splines)
model.lm_spline<-lm(damt ~ reg3 + reg4 + home + genf + plow + lgif + rgif + tdon +
                      agif + ns(incm_log,4) + ns(tgif_log,4) + agif_log + ns(plow_pwr,4) + lgif_pwr +
                      rgif_pwr + factor(chld) + factor(hinc) + factor(wrat),
                    data=data.train.std.y)
pred.valid.lm_spline <- predict(model.lm_spline, newdata = data.valid.std.y) # validation predictions
mpe.lm_spline<-mean((y.valid - pred.valid.lm_spline)^2) # mean prediction error
std.error.lm_spline<-sd((y.valid - pred.valid.lm_spline)^2)/sqrt(n.valid.y) # std error
lm_spline.num.coef<-length(model.lm_spline$coefficients)
mpe.lm_spline
ls_spline.num.coef<-length(model.lm_spline$coefficients)
summary(model.lm_spline)
#################
# Boosting Regression
#################
# We run gbm() with the option distribution="gaussian" for a regression problem;
# if it were a binary classification problem, we would use distribution="bernoulli".
# The argument n.trees=5000 indicates that we want 5000 trees, and the option interaction.depth=4
limits the depth of each tree.
library(caret)
##using this code to tune the GBM model. This takes a long time so I have commented it out
# myTuneGrid <- expand.grid(n.trees = 500,interaction.depth = c(7),shrinkage =
c(.01,.1,.5,.9),n.minobsinnode=10)
```

```
# fitControl <- trainControl(method = "repeatedcv", number = 5,repeats = 1, verboseIter =
FALSE,returnResamp = "all")
# myModel <- train(data.train.std.y$damt ~ reg3 + reg4 + home + genf + plow + lgif + rgif + tdon +
#                 agif + incm_log + tgif_log + agif_log + plow_pwr + lgif_pwr +
#                 rgif_pwr + factor(chld) + factor(hinc) + factor(wrat) ,
#                  data=data.train.std.y,method = "gbm",trControl = fitControl,tuneGrid = myTuneGrid)
set.seed(1)
model.boost_reg=gbm(data.train.std.y$damt ~ reg3 + reg4 + home + genf + plow + lgif + rgif + tdon +
                      agif + incm_log + tgif_log  +
                      factor(chld) + factor(hinc) + factor(wrat),

data=data.train.std.y,distribution="gaussian",n.trees=5000,interaction.depth=7,shrinkage = .01)
# The summary() function produces a relative influence plot and also outputs the relative influence
statistics.
summary(model.boost_reg)
pred.valid.boost_reg <- predict(model.boost_reg, newdata = data.valid.std.y,n.trees=5000) # validation
predictions
mpe.boost_reg<-mean((y.valid - pred.valid.boost_reg)^2) # mean prediction error
std.error.boost_reg<-sd((y.valid - pred.valid.boost_reg)^2)/sqrt(n.valid.y) # std error
boost_reg.num.coef<-length(model.boost_reg$coefficients)
mpe.boost_reg
boost_reg.num.coef<-length(model.boost_reg$var.names)
summary(model.boost_reg)
#################
# Random Forest Regression
#################
set.seed(3)
model.rf_reg <- randomForest::randomForest((data.train.std.y$damt) ~ reg3 + reg4 + home + genf + tdon
+
                                             incm_log + tgif_log + agif_log + plow_pwr + lgif_pwr +
                                             rgif_pwr + (chld) + (hinc) + (wrat),
                                           data.train.std.y)
model.rf_reg
randomForest::importance(model.rf_reg)
randomForest::varImpPlot(model.rf_reg)
pred.valid.rf_reg <- predict(model.rf_reg, data.valid.std.y) # n.valid.c post probs
mpe.rf_reg<-mean((y.valid - pred.valid.rf_reg)^2) # mean prediction error
std.error.rf_reg<-sd((y.valid - pred.valid.rf_reg)^2)/sqrt(n.valid.y) # std error
```

```
mpe.rf_reg
rf_reg.num.coef<-dim(model.rf_reg$importance)[1]
#Results Data Frame
model_pred<-c("ls1")
mean_prediction_error<-c(mpe.ls1)
std_error<-c(std.error.ls1)
num_of_coef<-c(ls1.num.coef)
results_pred<-data.frame(model_pred,mean_prediction_error,std_error,num_of_coef,stringsAsFactors =
FALSE)
results_pred<-rbind(c("ridge",mpe.ridge,std.error.ridge,ridge.num.coef),results_pred)
results_pred<-rbind(c("lasso",mpe.lasso,std.error.lasso,lasso.num.coef),results_pred)
results_pred<-rbind(c("pcr",mpe.pcr,std.error.pcr,"NA"),results_pred)
results_pred<-rbind(c("lm_wSpline",mpe.lm_spline,std.error.lm_spline,ls_spline.num.coef),results_pred)
results_pred<-rbind(c("boost_reg",mpe.boost_reg,std.error.boost_reg,boost_reg.num.coef),results_pred)
results_pred<-rbind(c("RandomForest_reg",mpe.rf_reg,std.error.rf_reg,rf_reg.num.coef),results_pred)
results_pred$mean_prediction_error<-round(as.numeric(results_pred$mean_prediction_error),4)
results_pred$std_error<-round(as.numeric(results_pred$std_error),4)
results_pred[order(results_pred$mean_prediction_error,decreasing = FALSE),]
best_pred_model<-model.lm_spline
# select model.ls1 since it has minimum mean prediction error in the validation sample
yhat.test <- predict(best_pred_model, 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="ABC.csv", row.names=FALSE) # use your initials for the file name
# submit the csv file in Canvas for evaluation based on actual test donr and damt values
```