```
#### Final Project Data Prep and EDA ####
library(reshape2)
library(ggplot2)
library(corrplot)
library(tree)
library(ISLR)
library(MASS)
library(randomForest)
library(gbm)
library(glmnet)
library(leaps)
library(e1071)
library(splines)
library(gam)
library(leaps)
library(pls)
library(rpart)
library(class)
library(caret)
library(neuralnet)

rm(list = ls()) # clear enironment
# Load Data
charity <- read.csv("C:/Users/Bivek/Documents/NorthWestern MSPA/Predict
422/Predict 422 R Stuff/charity.csv") # load the "charity.csv" file
# EDA on Raw Data ---------------------------------------------------------------------
summary(charity)
str(charity) # all but part are numeric/int
nums <- sapply(charity, is.numeric)
charity.numeric <- charity[nums]

# looking at dist of each variable
d <- melt(charity.numeric)
ggplot(d,aes(x = value)) +
  facet_wrap(~variable,scales = "free_x") +
  geom_histogram()

# Looked at outliers in boxplots
p <- ggplot(data = d, aes(x=variable, y=value)) +
  geom_boxplot(aes())
p + facet_wrap( ~ variable, scales="free")
## Thoughts
  # ID: Won't be used in prediction
  # reg1-reg4: will convert to factors
  # Home: binary
  # child: could create dummy??
```

```
  # hinc: could create dummy, but will most likely leave alone
  # genf: binary
  # wrat: we don't know exactly how this is calculated so probably factor, possible
create binary (wrat<=7, wrat >7)
  # avhv: right skewed, take log
  # incm: right skewed, take log
  # inca: right skewed, take log
  # plow: right skewed, take log
  # npro: ??
  # tgif: lots of 0s, convert to binary?
  # lgif: lots of 0s, convert to binary?
  # rgif: lots of 0s, convert to binary?
  # tdon: ??
  # tlag: right skewed, take log
  # agif: lots of 0s, convert to binary?
  # donr: response, binary
  # damt: response, normal
# Transformations ------------------------------------------------------

### Now looking ar all variables that aren't factors, binary, or response
charity.t <- charity # create transformed dataset, this is what we will use when
splitting

### Looking at histograms of variables to potentially be transformed
d2 <- melt(charity.numeric[11:21])
ggplot(d2,aes(x = value)) +
  facet_wrap(~variable,scales = "free_x") +
  geom_histogram()

## avhv
  par(mfrow=c(1,2))
  # original histogram
  hist(charity$avhv)
  # transformation
  charity.t$avhv <- log(charity.t$avhv)
  # transformed histogram
  hist(charity.t$avhv)
## incm
  # original histogram
  hist(charity$incm)
  # transformation
  charity.t$incm <- log(charity.t$incm)
  # transformed histogram
  hist(charity.t$incm)
## inca
  # original histogram
```

```
  hist(charity$inca)
  # transformation
  charity.t$inca <- log(charity.t$inca)
  # transformed histogram
  hist(charity.t$inca)
## plow
  # original histogram
  hist(charity$plow)
  # transformation
  charity.t$plow <- sqrt(charity$plow) ## log produces -inf values b/c can't do
log(0). so im using sqrt
  # transformed histogram
  hist(charity.t$plow)
## tgif, lgif, rgif, and agif
  # histograms
  hist(charity$tgif)
  hist(charity$lgif)
  # We could make both of these binary, then they would end up as the same variable
and we could delete one
## rgif
  # original histogram
  hist(charity$rgif)
  # transformation
  charity.t$rgif <- log(charity$rgif)
  # transformed histogram
  hist(charity.t$rgif)
## agif
  # original histogram
  hist(charity$agif)
  # transformation
  charity.t$agif <- log(charity$agif)
  # transformed histogram
  hist(charity.t$agif)
## tlag
  # original histogram
  hist(charity$tlag)
  # transformation
  charity.t$tlag <- log(charity$tlag)
  # transformed histogram
  hist(charity.t$tlag)
### Original histograms with transformed variables to show new distributions
  nums2 <- sapply(charity.t, is.numeric)
  charity.t.numeric <- charity.t[nums2]
  d <- melt(charity.t.numeric)
  ggplot(d,aes(x = value)) +
    facet_wrap(~variable,scales = "free_x") +
```

```
   geom_histogram()
par(mfrow=c(1,1))


charity.t2 <- charity.t # second transformed datset for factor dataframe
# Converting to factors for factor dataframe
  # reg1-reg4: will convert to factors
    charity.t2$reg1 <- as.factor(charity.t$reg1)
    charity.t2$reg2 <- as.factor(charity.t$reg2)
    charity.t2$reg3 <- as.factor(charity.t$reg3)
    charity.t2$reg4 <- as.factor(charity.t$reg4)

  # Home: binary
    charity.t2$home <- as.factor(charity.t$home)

  # genf: binary
    charity.t2$genf <- as.factor(charity.t$genf)

  # wrat: we don't know exactly how this is calculated so probably factor, possible
create binary (wrat<=7, wrat >7)
    charity.t2$wrat <- as.factor(charity.t$wrat)

# EDA on Transformed Dataset ----------------------------------------------
  summary(charity.t)
  str(charity.t)
  # Correlation of Numeric variables
  res <- cor(charity.t.numeric[!charity$part=="test",])
  cex.before <- par("cex")
  par(cex = 0.7)
  corrplot.mixed(res, mar=c(3,3,5,3), tl.pos="lt",  tl.col = "black", tl.cex = .8, diag="u")
  par(cex = cex.before)
  # Thoughts
  # Number of children as largest correlation with response
  # As expected, plow has a strong negative correlation with variables related to
income (avhv, incm, inca)
  # Tgif and lgif have a low correlation (.15), should keep both in model

  # Frequency tables for categorical/bianry variables
  nums3 <- sapply(charity.t2, is.numeric)
  ft <- apply(charity.t[!nums3],2,ftable)
  plot(ft)
  ## End EDA on Transformed Dataset

# Splitting up and Standardizing Data ---------------------------------------

# Traing Set
```

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

# Validation Set

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

# Test Set

data.test <- charity.t[charity$part=="test",]
n.test <- dim(data.test)[1] # 2007
x.test <- data.test[,2:21]

# Standardization

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)
# Factors Dataframe Creation ---------------------------------------------
```

```
# Splitting Factor Dataframe
data.train.F <- charity.t2[charity$part=="train",]
x.train.F <- data.train.F[,2:21]
c.train.F <- data.train.F[,22] # donr
n.train.c.F <- length(c.train.F) # 3984
y.train.F <-  data.train.F[c.train.F==1,23]# damt for observations with donr=1
n.train.y.F <- length(y.train.F) # 1995data.train.F[c.train.F==1,23]

# Validation Set - Factor Dataframe

data.valid.F <- charity.t2[charity$part=="valid",]
x.valid.F <- data.valid.F[,2:21]
c.valid.F <- data.valid.F[,22] # donr
n.valid.c.F <- length(c.valid.F) # 2018
y.valid.F <- data.valid.F[c.valid.F==1,23] # damt for observations with donr=1
n.valid.y.F <- length(y.valid.F) # 999

# Test Set - Factor Dataframe

data.test.F <- charity.t2[charity$part=="test",]
n.test.F <- dim(data.test.F)[1] # 2007
x.test.F <- data.test.F[,2:21]

# Standardization - Factor Dataframe

train.factors <- sapply(x.train.F, is.factor) # all variables converted to factors earlier
x.train.mean.F <- apply(x.train.F[,!train.factors], 2, mean) # take mean of all
variables except factors
x.train.sd.F <- apply(x.train.F[,!train.factors], 2, sd)
x.train.std.F <- t((t(x.train.F[,!train.factors])-x.train.mean.F)/x.train.sd.F) #
standardize to have zero mean and unit sd
apply(x.train.std.F, 2, mean) # check zero mean
apply(x.train.std.F, 2, sd) # check unit sd
data.train.std.c.F <- data.frame(x.train.std.F, donr=c.train.F) # to classify donr
data.train.std.y.F <- data.frame(x.train.std.F[c.train.F==1,], damt=y.train.F) # to
predict damt when donr=1

valid.factors <- sapply(x.valid.F, is.factor)
x.valid.std.F <- t((t(x.valid.F[,!valid.factors])-x.train.mean.F)/x.train.sd.F) #
standardize using training mean and sd
data.valid.std.c.F <- data.frame(x.valid.std.F, donr=c.valid.F) # to classify donr
data.valid.std.y.F <- data.frame(x.valid.std.F[c.valid.F==1,], damt=y.valid.F) # to
predict damt when donr=1


test.factors <- sapply(x.test.F, is.factor)
```

```
x.test.std.F <- t((t(x.test.F[,!test.factors])-x.train.mean.F)/x.train.sd.F) # standardize
using training mean and sd
data.test.std.F <- data.frame(x.test.std.F)


# Add back in factors: Classification

data.train.std.c.F$reg1 <- x.train.F$reg1
data.train.std.c.F$reg2 <- x.train.F$reg2
data.train.std.c.F$reg3 <- x.train.F$reg3
data.train.std.c.F$reg4 <- x.train.F$reg4
data.train.std.c.F$home <- x.train.F$home
data.train.std.c.F$genf <- x.train.F$genf
data.train.std.c.F$wrat <- x.train.F$wrat


data.valid.std.c.F$reg1 <- x.valid.F$reg1
data.valid.std.c.F$reg2 <- x.valid.F$reg2
data.valid.std.c.F$reg3 <- x.valid.F$reg3
data.valid.std.c.F$reg4 <- x.valid.F$reg4
data.valid.std.c.F$home <- x.valid.F$home
data.valid.std.c.F$genf <- x.valid.F$genf
data.valid.std.c.F$wrat <- x.valid.F$wrat

data.test.std.F$reg1 <- x.test.F$reg1
data.test.std.F$reg2 <- x.test.F$reg2
data.test.std.F$reg3 <- x.test.F$reg3
data.test.std.F$reg4 <- x.test.F$reg4
data.test.std.F$home <- x.test.F$home
data.test.std.F$genf <- x.test.F$genf
data.test.std.F$wrat <- x.test.F$wrat

# Add back in factors: Predicticting

data.train.std.y.F$reg1 <- x.train.F[c.train.F==1,'reg1']
data.train.std.y.F$reg2 <- x.train.F[c.train.F==1,'reg2']
data.train.std.y.F$reg3 <- x.train.F[c.train.F==1,'reg3']
data.train.std.y.F$reg4 <- x.train.F[c.train.F==1,'reg4']
data.train.std.y.F$home <- x.train.F[c.train.F==1,'home']
data.train.std.y.F$genf <- x.train.F[c.train.F==1,'genf']
data.train.std.y.F$wrat <- x.train.F[c.train.F==1,'wrat']


data.valid.std.y.F$reg1 <- x.valid.F[c.valid.F==1,'reg1']
data.valid.std.y.F$reg2 <- x.valid.F[c.valid.F==1,'reg2']
data.valid.std.y.F$reg3 <- x.valid.F[c.valid.F==1,'reg3']
```

```
data.valid.std.y.F$reg4 <- x.valid.F[c.valid.F==1,'reg4']
data.valid.std.y.F$home <- x.valid.F[c.valid.F==1,'home']
data.valid.std.y.F$genf <- x.valid.F[c.valid.F==1,'genf']
data.valid.std.y.F$wrat <- x.valid.F[c.valid.F==1,'wrat']
#### Classification Models ----------------------------------------------
  # Response variable is DONR
  # 2 Different transformed/cleaned datasets:
str(data.train.std.c) # All data left as numeric and standardized
str(data.train.std.c.F) # reg1-4, home, genf, and wrat were converted to factors
# Logistic ------------------------------------------------------------

# All variables - factor dataset gave better results
model.log1 <- glm(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc +
I(hinc^2) + genf + wrat +
                    avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,
                  data.train.std.c.F, family=binomial("logit"))

post.valid.log1 <- predict(model.log1, data.valid.std.c.F, 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.F[order(post.valid.log1, decreasing=T)]-2)
plot(profit.log1) # see how profits change as more mailings are made
n.mail.valid.1 <- which.max(profit.log1) # number of mailings that maximizes
profits
c(n.mail.valid.1, max(profit.log1)) # report number of mailings and maximum profit
# 1251 11650

cutoff.log1 <- sort(post.valid.log1, decreasing=T)[n.mail.valid.1+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
mean(chat.valid.log1 != c.valid) # Error rate
#                  c.valid
# chat.valid.log1    0   1
#               0 744  23
#               1 275 976
# check n.mail.valid = 275+976 = 1251
# check profit = 14.5*976-2*1251 = 11650

# Backwards Selection - factor dataset gave better results
```

```
model.logB <- step(glm(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc +
I(hinc^2) + genf + wrat +
                avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,
              data.train.std.c.F, family=binomial("logit")))

post.valid.logB <- predict(model.logB, data.valid.std.c.F, type="response")

# Backwards Profit

profit.logB <- cumsum(14.5*c.valid[order(post.valid.logB, decreasing=T)]-2)
plot(profit.logB) # see how profits change as more mailings are made
n.mail.valid.B <- which.max(profit.logB) # number of mailings that maximizes
profits
c(n.mail.valid.B, max(profit.logB)) # report number of mailings and maximum profit
# 1349.0 11642.5


# Forwards Selection

model.logF <- step(glm(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc +
I(hinc^2) + genf + wrat +
                  avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,
                data.train.std.c.F, family=binomial("logit")),direction="forward")

post.valid.logF <- predict(model.logF, data.valid.std.c.F, type="response")

# Forwards Profit - same results as original model, used factor dataset

profit.logF <- cumsum(14.5*c.valid[order(post.valid.logF, decreasing=T)]-2)
plot(profit.logF) # see how profits change as more mailings are made
n.mail.valid.F <- which.max(profit.logF) # number of mailings that maximizes profits
c(n.mail.valid.F, max(profit.logF)) # report number of mailings and maximum profit
# 1251 11650


# Stepwise Selection - factor dataset was best

model.logS <- step(glm(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc +
I(hinc^2) + genf + wrat +
                  avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,
                data.train.std.c.F, family=binomial("logit")),direction="both")

post.valid.logS <- predict(model.logS, data.valid.std.c.F, type="response")

# Stepwise Profit
```

```
profit.logS <- cumsum(14.5*c.valid[order(post.valid.logS, decreasing=T)]-2)
plot(profit.logS) # see how profits change as more mailings are made
n.mail.valid.S <- which.max(profit.logS) # number of mailings that maximizes profits
c(n.mail.valid.S, max(profit.logS)) # report number of mailings and maximum profit
# 1350.0 11640.5

## Of all variable selection methods, the full model performed the best
# LDA Model ----------------------------------------------------------------

# Slightly better than example code when using factor dataset. Not as good as
logistic

model.lda1 <- lda(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2)
+ genf + wrat +
                    avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,
                  data.train.std.c.F)

post.valid.lda1 <- predict(model.lda1, data.valid.std.c.F)$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.F[order(post.valid.lda1, decreasing=T)]-2)
plot(profit.lda1) # see how profits change as more mailings are made
n.mail.valid.lda1 <- which.max(profit.lda1) # number of mailings that maximizes
profits
c(n.mail.valid, max(profit.lda1)) # report number of mailings and maximum profit
# 1103.0 11646.5

cutoff.lda1 <- sort(post.valid.lda1, decreasing=T)[n.mail.valid.lda1+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.F) # classification table
mean(chat.valid.lda1 != c.valid.F) # Error rate
# 0.1719524
# QDA ----------------------------------------------------------------
## Build QDA Model

# Initial model contains all variables
qda.fit1 <- qda(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc
                + genf + wrat + avhv + incm + inca + plow + npro
                + tgif + lgif + rgif + tdon + tlag + agif,
                data = data.train.std.c)
qda.fit1
```

```
# Create predictions based on qda.fit1 and examine confusion matrix.
qda.fit1.preds <- predict(qda.fit1, data.valid.std.c)
qda.fit1.preds.class <- qda.fit1.preds$class
table(qda.fit1.preds.class, c.valid)
mean(qda.fit1.preds.class != c.valid)
qda.fit1.preds.posterior <- qda.fit1.preds$posterior[,2]
# Error  0.1620416


# Attempt reduced predictor set from above model to determine if performance is
improved
qda.fit2 <- qda(donr ~ reg2 + reg3 + reg4 + home*chld + I(hinc^2) + wrat +
incm*inca + avhv + plow + tdon + npro + tlag, data = data.train.std.c)
qda.fit2

# Create predictions based on qda.fit2 and examine confusion matri
qda.fit2.preds <- predict(qda.fit2, data.valid.std.c)
qda.fit2.preds.class <- qda.fit2.preds$class
table(qda.fit2.preds.class, c.valid)
mean(qda.fit2.preds.class != c.valid)
qda.fit2.preds.posterior <- qda.fit2.preds$posterior[,2]
# Error 0.2180377

### QDA1 Profit
profit.qda1 = cumsum(14.5 * c.valid[order(qda.fit1.preds.posterior, decreasing = T)]
-2)
plot(profit.qda1, col = "darkolivegreen4", main = "QDA Model 1 - Profit") # see how
profits change as more mailings are made
n.mail.qda1 = which.max(profit.qda1) # number of mailings that maximizes profits
c(n.mail.qda1, max(profit.qda1))
# 1390.0 11241.5
View(qda.fit2.valid.preds.post)

### QDA2 Profit
profit.qda2 = cumsum(14.5 * c.valid[order(qda.fit2.preds.posterior, decreasing = T)]
-2)
plot(profit.qda2, col = "darkolivegreen4", main = "QDA Model 2 - Profit") # see how
profits change as more mailings are made
n.mail.qda2 = which.max(profit.qda2) # number of mailings that maximizes profits
c(n.mail.qda2, max(profit.qda2))
# 1378 11280
View(qda.fit2.valid.preds.post)
# Random Trees ------------------------------------------------------------
## Basic Tree
```

```
data.train.std.c.F$donr <- as.factor(data.train.std.c.F$donr) # Make response
variable a factor for classification tree
myTree <- tree(donr~.,data.train.std.c.F)
summary(myTree)
plot(myTree)
text(myTree, pretty=0)
myTree

## Bagging
set.seed(1)
bag1 <- randomForest(donr~.,data=data.train.std.c.F, mtry=20,importance=TRUE)
bag1

# predict on validation set
data.valid.std.c.F$donr <- as.factor(data.valid.std.c.F$donr) # convert validation
response to factor
yhat.bag = predict(bag1,newdata=data.valid.std.c.F)

# Bag Profit
profit.bag <- cumsum(14.5*c.valid.F[order(yhat.bag, decreasing=T)]-2)
plot(profit.bag) # see how profits change as more mailings are made
n.mail.valid.bag <- which.max(profit.bag) # number of mailings that maximizes
profits
c(n.mail.valid.bag, max(profit.bag))
# 1038 11032

# Bag Accuracy
table(yhat.bag, c.valid.F) # classification table
mean(yhat.bag != c.valid.F) # Error rate
# 0.1134787

## Random Forest
set.seed(1)
RF1 <- randomForest(donr~.,data=data.train.std.c.F, mtry= 5,importance=TRUE)
RF1

# predict on validation set
yhat.RF1 = predict(RF1,newdata=data.valid.std.c.F)

# RF Profit
profit.RF1 <- cumsum(14.5*c.valid.F[order(yhat.RF1, decreasing=T)]-2)
plot(profit.RF1) # see how profits change as more mailings are made
n.mail.valid.RF1 <- which.max(profit.RF1) # number of mailings that maximizes
profits
c(n.mail.valid.RF1, max(profit.RF1))
# 1053.0 11132.5
```

```
# RF Accuracy
table(yhat.RF1, c.valid.F) # classification table
mean(yhat.RF1 != c.valid.F) # Error rate
# 0.1119921


## Boosting

set.seed(1)
boost1=gbm(donr~.,data=data.train.std.c.F,distribution=
"gaussian",n.trees=5000,interaction.depth=6)
summary(boost1)

par(mfrow=c(1,2))
plot(boost1,i="chld")
plot(boost1,i="hinc")
par(mfrow=c(1,1))

# Predict Boost
set.seed(1)
yhat.boost1 = predict.gbm(boost1, newdata = data.valid.std.c.F,n.trees = 5000, type
= "response")

# Boost Profit
profit.boost1 <- cumsum(14.5*c.valid.F[order(yhat.boost1, decreasing=T)]-2)
plot(profit.boost1) # see how profits change as more mailings are made
n.mail.valid.boost <- which.max(profit.boost1) # number of mailings that maximizes
profits
c(n.mail.valid.boost, max(profit.boost1))
# 1271.0 11885.5, depth 6 and default shrinkage


# Boost Accuracy
cutoff.boost <- sort(yhat.boost1, decreasing=T)[n.mail.valid.boost+1] # set cutoff
based on n.mail.valid
chat.valid.boost <- ifelse(yhat.boost1>cutoff.boost, 1, 0) # mail to everyone above
the cutoff
table(chat.valid.boost, c.valid.F) # classification table
mean(chat.valid.boost != c.valid.F) # Error rate
# 0.1387512389
# SVM ---------------------------------------------------------------------

set.seed(1)
# Use cross-validation to select the best parameters for the SVC.
tune.out = tune(svm, donr ~., data=data.train.std.c.F, kernel = "linear",
```

```
            ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10)))
summary(tune.out)

# Best cost parameter is 5
set.seed(1)
svm.fit1 = svm(donr ~., data=data.train.std.c.F, kernel = "linear",cost = 5, scale =
FALSE)
summary(svm.fit1)
svmpred = predict(svm.fit1, data.valid.std.c.F)
table(predict=svmpred, truth=data.valid.std.c.F$donr)
svmError <- mean(svmpred != c.valid)
svmError
# Error 0.1555996036

# SVM Profit
profit.svm <- cumsum(14.5*c.valid[order(svmpred, decreasing=T)]-2)
plot(profit.svm) # see how profits change as more mailings are made
n.mail.valid.svm <- which.max(profit.svm) # number of mailings that maximizes
profits
c(n.mail.valid.svm, max(profit.svm))
# 1085 10561
# KNN ---------------------------------------------------------------------

# Cross validation using train() to find best value of K
set.seed(1)
data.train.std.c.F$donr <- as.factor(data.train.std.c.F$donr)
knn.t <- train(
  donr~.,
  data=data.train.std.c.F,
  method='knn',
  tuneGrid=expand.grid(.k=1:15),
  metric='Accuracy',
  trControl=trainControl(
    method='repeatedcv',
    number=10,
    repeats=15))

knn.t
plot(knn.t) # looks to level off around 11, best around 13/14
confusionMatrix(knn.t)

dim(data.train.std.c)
dim(data.valid.std.c)
# Perform K-nearest neighbors classification and train on validation set
set.seed(1)
```

```
knn.pred=knn(data.train.std.c.F, data.valid.std.c.F,c.train,k=13) # Profit will be
different every time we run it with the same value of K

# Check prediction accuracy
table(knn.pred, c.valid)
(887 + 971)/2018 # Percent correctly classified Prediction Error
(132 + 28)/2018 # Percent incorrectly classified Classification Error
# 0.9207136 Accuracy
# 0.07928642 Error
# KNN Profit
profit.knn <- cumsum(14.5*c.valid[order(knn.pred, decreasing=T)]-2)
plot(profit.knn) # see how profits change as more mailings are made
n.mail.valid.knn <- which.max(profit.knn) # number of mailings that maximizes
profits
c(n.mail.valid.knn, max(profit.knn))
# 1103.0 11873.5
# Neural Network ------------------------------------------------------------

n <- names(data.train.std.c)
f <- as.formula(paste("donr ~", paste(n[!n %in% "donr"], collapse = " + ")))
set.seed(1)
neural.model <- neuralnet(f,data=data.train.std.c, hidden = 5, lifesign="minimal",
linear.output=FALSE, threshold=0.1)

plot(neural.model, rep = "best")

set.seed(1)
neural.model.results <- compute(neural.model, data.valid.std.c[,-21])

results <- data.frame(actual = data.valid.std.c$donr, prediction =
neural.model.results$net.result)

error = mean(results$actual - results$prediction)^2 #mean prediction error
error
# 0.00009876081308

# Neural Net Profit
profit.neural <- cumsum(14.5*c.valid[order(results[,2], decreasing=T)]-2)
plot(profit.neural) # see how profits change as more mailings are made
n.mail.valid <- which.max(profit.neural) # number of mailings that maximizes
profits
c(n.mail.valid, max(profit.neural))
# 1291 11715
#### Prediction Models ------------------------------------------------------
str(data.train.std.y) # All data left as numeric and standardized
str(data.train.std.y.F) # reg1-4, home, genf, and wrat were converted to factors
```

```
# Ridge Regression/LASSO ----------------------------------------------------

###LASSO w/ BestLam is best model in this section. MSE: 1.512279 SD: 0.1546922

# Building Rideg/LASSO Model
x.train.matrix=model.matrix(damt~.,data.train.std.y.F)[,-1]
y.train.matrix=data.train.std.y.F$damt # not a matrix, just simpler naming
convention
grid=10^seq(10,-2, length =100)
lasso.mod=glmnet(x.train.matrix,y.train.matrix,alpha=1,lambda=grid) # lasso
ridge.mod=glmnet(x.train.matrix,y.train.matrix,alpha=0,lambda=grid) # ridge

plot(lasso.mod, "lambda")
plot(ridge.mod, "lambda")

# Cross Validation to Select Lambda
set.seed(1)
cv.out.lasso <- cv.glmnet(x.train.matrix,y.train.matrix,alpha=1) # lasso
cv.out.ridge <- cv.glmnet(x.train.matrix,y.train.matrix,alpha=0) # ridge

plot(cv.out.lasso)
plot(cv.out.ridge)

# Lasso lambda
bestlam.lasso <- cv.out.lasso$lambda.min
OneSElam.lasso <- cv.out.lasso$lambda.1se

# Ridge Lambda
bestlam.ridge <- cv.out.ridge$lambda.min
OneSElam.ridge <- cv.out.ridge$lambda.1se

# prediction for LASSO based on best lambda and OneSE lambda
x.valid.matrix=model.matrix(damt~.,data.valid.std.y.F)[,-1]
y.valid.matrix=data.valid.std.y.F$damt # not a matrix, just simpler naming
convention
lasso.pred.best=predict(lasso.mod ,s=bestlam.lasso ,newx=x.valid.matrix)
lasso.pred.OneSE=predict(lasso.mod ,s=OneSElam.lasso ,newx=x.valid.matrix)

# Prediction for Ridge
ridge.pred.best=predict(ridge.mod ,s=bestlam.ridge ,newx=x.valid.matrix)
ridge.pred.OneSE=predict(ridge.mod ,s=OneSElam.ridge ,newx=x.valid.matrix)

# MSE for BestLam LASSO Model
mean((y.valid.matrix - lasso.pred.best)^2) # mean prediction error
# 1.512279
sd((y.valid.matrix - lasso.pred.best)^2)/sqrt(n.valid.y) # std error
```

```
# 0.1546922

# MSE for OneSE Lam LASSO Model
mean((y.valid.matrix - lasso.pred.OneSE)^2) # mean prediction error
#  1.654438
sd((y.valid.matrix - lasso.pred.OneSE)^2)/sqrt(n.valid.y) # std error
# 0.1601282

# MSE for BestLam Ridge Model
mean((y.valid.matrix - ridge.pred.best)^2) # mean prediction error
# 1.521255
sd((y.valid.matrix - ridge.pred.best)^2)/sqrt(n.valid.y) # std error
# 0.1571182

# MSE for OneSE Lam Ridge Model
mean((y.valid.matrix - ridge.pred.OneSE)^2) # mean prediction error
# 1.635708
sd((y.valid.matrix - ridge.pred.OneSE)^2)/sqrt(n.valid.y) # std error
# 0.1663205
# Least Squares ----------------------------------------------------------

### All Variables

# Factored Dataframe
model.ls1 <- lm(damt ~ .,
            data.train.std.y.F)
summary(model.ls1)
pred.valid.ls1 <- predict(model.ls1, newdata = data.valid.std.y.F) # validation
predictions
mean((y.valid - pred.valid.ls1)^2) # mean prediction error
# 1.472403
sd((y.valid - pred.valid.ls1)^2)/sqrt(n.valid.y) # std error
# 0.1541609

### Best Subset Selection (k=10 CV)
predict.regsubsets=function(object, newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}
k=10
set.seed(1306)
folds=sample(1:k, nrow(data.train.std.y.F), replace=TRUE)
cv.errors=matrix(NA, k, 20,dimnames=list(NULL, paste(1:20)))
```

```
for(j in 1:k){
  best.fit=regsubsets(damt~., data=data.train.std.y.F[folds!=j,], nvmax=20)
  for(i in 1:20){
    pred=predict(best.fit, data.train.std.y.F[folds==j,],id=i)
    cv.errors[j,i]=mean((data.train.std.y.F$damt[folds==j]-pred)^2)
  }
}
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
which.min(mean.cv.errors)
# 20=1.299088
# chose 14 for simplicity
plot(mean.cv.errors, type="b")
cv.best=regsubsets(damt~., data=data.train.std.y.F, nvmax=20)
coef(cv.best, 14)


pred.bestsub.ls <- predict.regsubsets(cv.best, newdata = data.valid.std.y.F, id=20) #
validation predictions
mean((y.valid -pred.bestsub.ls)^2) # mean prediction error
# 1.483712988
sd((y.valid - pred.bestsub.ls)^2)/sqrt(n.valid.y) # std error
# 0.1533872925



### Partial Least Squares

set.seed(1)
pls.fit=plsr(damt~., data=data.train.std.y.F,scale=TRUE, validation="CV")
summary(pls.fit)
validationplot(pls.fit, val.type="MSEP")
# M18=1.143
pls.pred=predict(pls.fit, newdata=data.valid.std.y.F, ncomp=18)
mean((pls.pred-y.valid)^2)
# 1.472454
sd((y.valid - pls.pred)^2)/sqrt(n.valid.y)
# 0.1541474
# GAM (Not Used)----------------------------------------------------------
gam.charity =gam(damt~., data=data.train.std.y.F)
plot.gam(gam.charity, se=TRUE,col="red")
summary(gam.charity)
pred=predict(gam.charity, newdata=data.valid.std.y.F)
mean((pred-y.valid)^2)
# 1.472403
sd((y.valid - pred)^2)/sqrt(n.valid.y) # std error
```

```
# 0.1541609
gam.charity$coef
# RandomForest ------------------------------------------------------------
set.seed(1)
# RF2 is better model
RF2 <- randomForest(y=data.train.std.y.F[,14],
                    x =data.train.std.y.F[,-14], ntree=1000, replace=TRUE, do.trace = 100,
                    importance = TRUE)
set.seed(1)
RF3 <- randomForest(damt~.,data=data.train.std.y.F, mtry= 4,importance=TRUE)

yhat.RF2 = predict(RF2,newdata=data.valid.std.y.F)
yhat.RF3 = predict(RF3,newdata=data.valid.std.y.F)

mean((yhat.RF2-y.valid)^2)
# 1.64065585
sd((y.valid - yhat.RF2)^2)/sqrt(n.valid.y)
# 0.1715645
mean((yhat.RF3-y.valid)^2)
# 1.646324
sd((y.valid -yhat.RF3)^2)/sqrt(n.valid.y)
# 0.1729773
# Boost -------------------------------------------------------------------

set.seed(1)
boost.damt = gbm(damt ~., data = data.train.std.y, distribution = "gaussian",
                 n.trees=5000, interaction.depth=4)
summary(boost.damt)
set.seed(1)
pred.boost.damt <- predict(boost.damt, newdata = data.valid.std.y, n.trees = 5000)
boost.error <- mean((y.valid - pred.boost.damt)^2)
boost.standardError <- sd((y.valid - pred.boost.damt)^2/sqrt(n.valid.y))

boost.error # Error 1.539575259
boost.standardError # Error 0.1668604649
# PCR ---------------------------------------------------------------------

pcr1 <- pcr(damt~., data=data.train.std.y.F,scale = TRUE, validation = "CV")
summary(pcr1)
validationplot(pcr1, val.type="MSEP")
validationplot(pcr1,val.type="MSEP")
pcr.pred <- predict(pcr1,newdata=data.valid.std.y.F, ncomp=28)
mean((pcr.pred-y.valid)^2)
# 1.472403118
sd((y.valid -pcr.pred)^2)/sqrt(n.valid.y)
# 0.1541608612
```

```
# RPart (Not Used)------------------------------------------------------------

rpart1 <- rpart(damt~., data=data.train.std.y.F)
summary(rpart1)
str(rpart1)
rpart.pred <- predict(rpart1, newdata=data.valid.std.y.F)
mean((rpart.pred-y.valid)^2)
# 2.211895
sd((y.valid -rpart.pred)^2)/sqrt(n.valid.y)
# 0.1900094
#### Results ------------------------------------------------------------
# Best Classification model: boost1
summary(boost1)
# Best Predictive Model: model.ls1
summary(model.ls1)
# Classification on Test Data with Boost1 Model-------------------------------
post.test <- predict(boost1, data.test.std.F, n.trees=5000,type="response") # post
probs for test data

# Boost Profit
profit.boost1 <- cumsum(14.5*c.valid.F[order(yhat.boost1, decreasing=T)]-2)
plot(profit.boost1) # see how profits change as more mailings are made
n.mail.valid.boost <- which.max(profit.boost1) # number of mailings that maximizes
profits
c(n.mail.valid.boost, max(profit.boost1))
# 1271.0 11885.5, depth 6 and default shrinkage

# Oversampling adjustment for calculating number of mailings for test set

n.mail.valid <- which.max(profit.boost1)
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)

# based on this model we'll mail to the 319 highest posterior probabilities
# Prediction on Test Data with Least Squares Model ------------------------
yhat.test <- predict(model.ls1, newdata = data.test.std.F) # 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="CTG5_TV_MK_JZ_BA.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

# Calculate total and average
hat.data.frame <- data.frame(chat = chat.test, yhat = yhat.test)
View(hat.data.frame)
# Check expected profit from mailing.
PredictedDonors <- subset(hat.data.frame, chat==1)
FinalRev <- sum(PredictedDonors$yhat)
FinalProfit <- FinalRev - (2*length(PredictedDonors[,1]))
# 3990.797675
AverageDonation <- FinalRev/length(PredictedDonors[,1])
# 14.51033754
```