Bruckner, Funk, Sheets, Zimmerman
Predict 422, Sec 56

Final Project

Introduction

In this report we are developing machine-learning models to optimize a charitable organization's direct marketing campaign. Our goal is two-fold: to identify likely donors using a classification model in order to maximize the net profit from the mailings, and to estimate the gift amounts from donors using a prediction model. We created several classification models, including logistic regression, logistic regression GAM, LDA, QDA, k-nearest neighbors, decision tree, boosts and random forest, and support vector machine, in order to determine which model best identifies likely donors. We also created several prediction models, including least squares regression, best subset selection with k-fold cross-validation, principal components regression, partial least squares, ridge regression, and lasso (Are we doing a decision tree model too for the prediction models?), in order to predict the amount of donations.

Data Exploration

The data we are using for this report is the charity data set. This data set contains 3984 training observations, 2018 validation observations, and 2007 test observations. The data has been weighted in order to make the training and validation samples each contain roughly an equivalent number of donors and non-donors. The charity data set includes two response variables, donr (donor) and damt (donation amount), for the classification models and prediction models, respectively. It also contains 20 predictor variables, including the geographic regions of previous donors, whether the donor is a homeowner or not, the number of children the donor has, the donor's income, the donor's gender, the amounts of previous donations, and various other wealth, income, and donation factors that may help create strong classification and prediction machine-learning models.

Prior to developing any models, we examined our data in order to see if there were any missing values or abnormalities in the data. Although the data set did not contain any missing values, we noticed that several of the variables might benefit from transformations. We decided to apply logarithmic transformations to seven right-skewed variables: avhv, incm, inca, tgif, lgif, rgif, and agif. The top row of **Figure 1** shows the variables prior to transformation, while the bottom row shows the variables after transformation.[i] As the figure shows, the logarithmic transformation helped normalize the distributions of these seven variables.

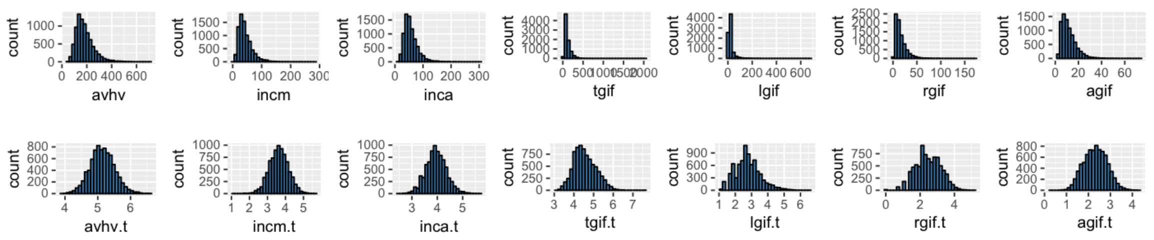


Figure 1: Transformation of Variables

[i] Please note: we kept the transformed variable names the same as the original variable names for our models, but we added a ".t" to the end of each variable name in Figure 1 to clearly distinguish between the original and transformed variables.

After transforming our variables, we created a correlation matrix to gain further insight into the data. **Figure 2** shows a visualization of the correlation matrix, while **Table 1** shows the values of the most statistically significant correlations.

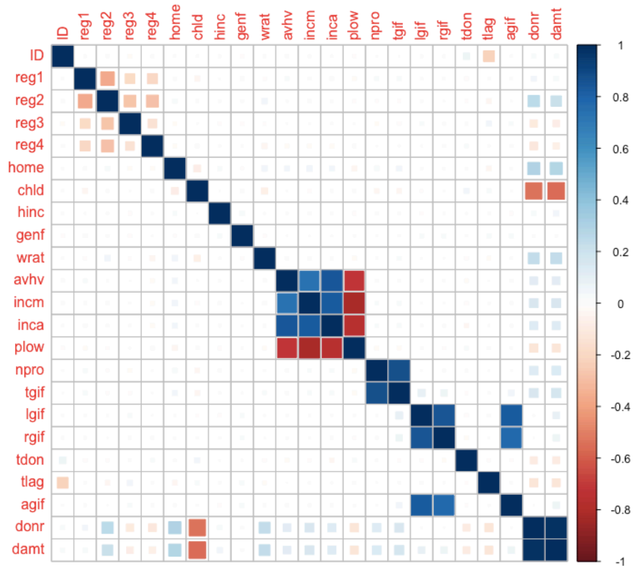


Figure 2: Correlation Matrix

var1	var2	corr
damt	donr	0.9817018
tgif	npro	0.8734276
rgif	lgif	0.8512241
inca	avhv	0.8484572
inca	incm	0.8296747
agif	lgif	0.8294224
plow	incm	-0.8120381
agif	rgif	0.7706645
plow	inca	-0.7510141
incm	avhv	0.7304313
plow	avhv	-0.7187952
damt	chld	-0.5531045
donr	chld	-0.5326077

Table 1: Significant Correlations

As **Figure 2** and **Table 1** reveal, certain variables have strong correlations. For instance, tgif (dollar amount of lifetime gifts to date) has a very strong positive correlation with npro (lifetime number of promotions received to date). Some variables, such as plow (percent categorized as "low income" in potential donor's neighborhood) and incm (median family income in potential donor's neighborhood in $ thousands), have a strong negative correlation. The correlations are something we bear in mind as we create our classification and prediction models.

Classification Models and Analysis

Model 1: Logistic Regression

Coefficient Estimates

Variable	Estimate	Std. Error	t value	Pr(>|t|)	Signif.
(Intercept)	149.920	2.976	50.382	< 2e-16	***
age	-66.758	68.946	-0.968	0.33364	
sex	-304.651	69.847	-4.362	1.74E-05	***
bmi	518.663	76.573	6.773	6.01E-11	***
map	388.111	72.755	5.335	1.81E-07	***
tc	-815.268	537.549	-1.517	0.13034	
ldl	387.604	439.162	0.883	0.37811	
hdl	162.903	269.117	0.605	0.54539	
tch	323.832	186.803	1.734	0.08396	.
ltg	673.62	206.888	3.256	0.00125	**
glu	94.219	79.59	1.184	0.23737	
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1					

Test Errors

Test MSE	Standard Error of Test MSE
3111.265	361.0908

Bruckner, Funk, Sheets, Zimmerman
Predict 422, Sec 56

In the coefficient estimates table above, we can see that only five of the ten predictors are statistically significant: sex, bmi, map, tch, and ltg. If I were to explore this data further, I might want to try simplifying the least squares regression model by fitting only the significant variables.

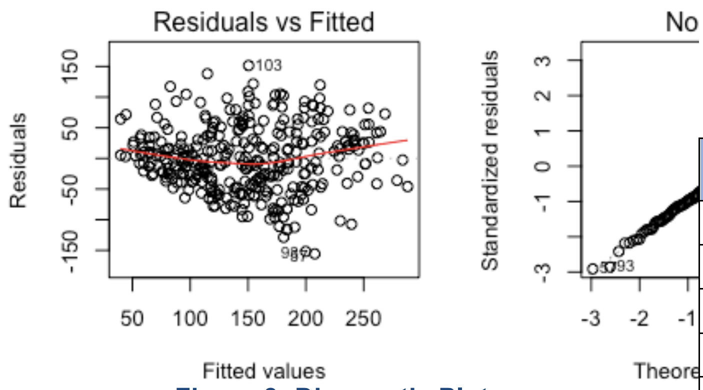


Figure 3: Diagnostic Plots

Variable	Estimate
(Intercept)	150.1166
sex	-306.042
bmi	538.8274
map	389.0673
tc	-379.0379
tch	332.6735

Test MSE	Standard Error of Test MSE
3095.483	369.7526

The Residuals vs Fitted plot in **Figure 1** reveals that the residuals for Model 1 appear randomly scattered about a mean of zero, and the Normal Q-Q plot shows that the residuals fall nearly in a straight line. Both plots suggest that the data do not violate the assumptions of normality.

Model 2: Best subset selection model using BIC to select the number of predictors

Coefficient Estimates *Test Errors*

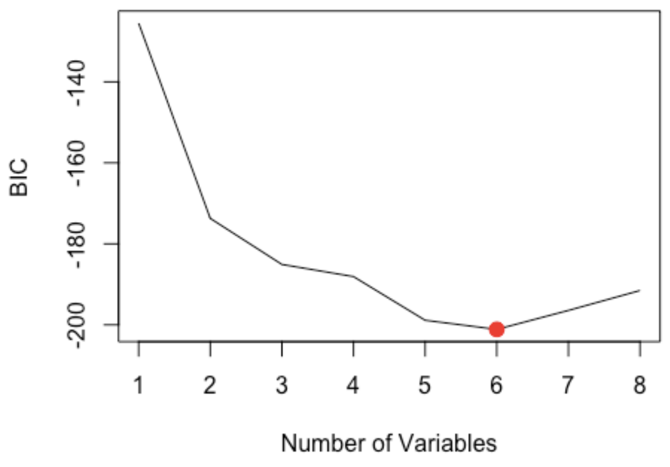


Figure 4: Best Subset Selection Plot

In order to determine the best subset selection model, I looked for the model with the smallest BIC value. **Figure 2** shows that the model with six variables has the smallest BIC value.

Model 3: Best subset selection model using 10-fold cross-validation to select the predictors

ltg	527.5658

For Model 3, I used 10-fold cross-validation to find the best subset selection model with the lowest mean cross-validation error. **Figure 3** shows that the model with six variables has the lowest mean cross-validation error.

As the tables show, the coefficient estimates, test MSE, and standard error of the test MSE are identical to the values in Model 2.

Coefficient Estimates *Test Errors*

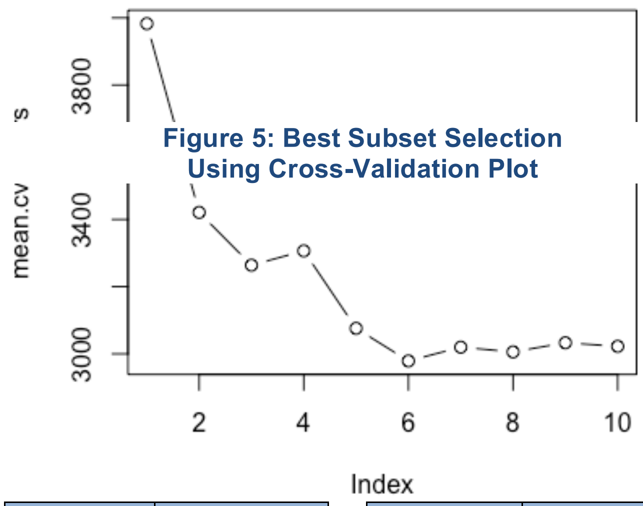


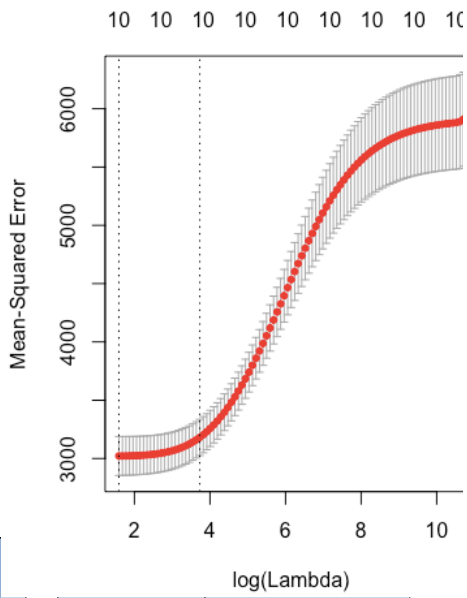
Figure 5: Best Subset Selection Using Cross-Validation Plot

Variable	Estimate
(Intercept)	149.99068
age	-11.33162
sex	-156.91053
bmi	374.44939
map	264.89998
tc	-31.96990
ldl	-66.89724
hdl	-174.01202
tch	123.97204
ltg	307.68646
glu	134.48120

Test MSE	Standard Error of Test MSE
3070.870	350.5467

Model 4: Ridge regression model using 10-fold cross-validation to select the largest value of λ such that the cross-validation error is within 1 standard error of the minimum

Coefficient Estimates
Test Errors



Variable	Estimate
(Intercept)	150.1166
sex	-306.042
bmi	538.8274
map	389.0673
tc	-379.0379
tch	332.6735
ltg	527.5658

3095.483	369.7526

Figure 6: Cross-Validation Plot

In **Figure 4**, we can see the MSE is smallest when log(Lambda) is smallest. The first vertical line from the left of the plot signifies the smallest MSE, while the second vertical line represents one standard deviation from the minimum MSE. The number 10 at the top of the plot indicates that all ten predictor variables are present in the model regardless of the lambda value. I used lambda = 41.67209 to estimate the coefficients and calculate the test errors since this was the largest value of lambda where the error was still within one standard error of the minimum MSE.

Model 5: Lasso model using 10-fold cross-validation to select the largest value of λ such that the cross-validation error is within 1 standard error of the minimum

Coefficient Estimates

Variable	Estimate
(Intercept)	149.95298
age	.
sex	-119.62207
bmi	501.56473
map	270.92613
tc	.
ldl	.
hdl	-180.29436
tch	.
ltg	390.55001
glu	16.58881

Test Errors

Test MSE	Standard Error of Test MSE
2920.041	346.2248

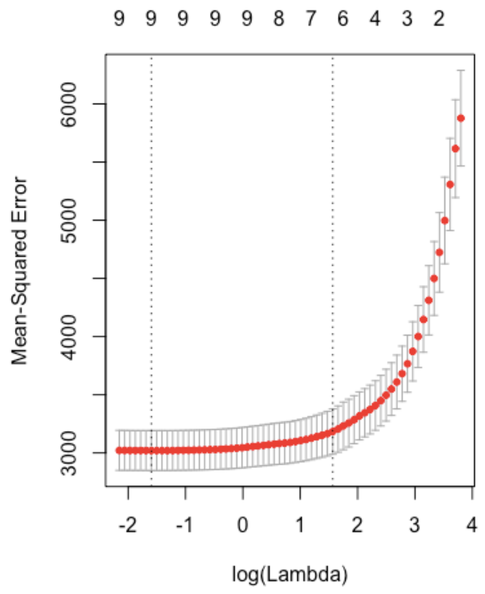


Figure 5: Cross-Validation Plot

Looking at the second vertical line from the left in **Figure 5**, we can see that the number of predictors in the model is six, as denoted by the number at the top of the plot. Although Models 2 and 3 also use only six predictors, the predictors for Model 5 are slightly different. All three models use the sex, bmi, map, and ltg predictors, but Model 5 uses hdl and glu instead of tc and tch as used in the other two models. To estimate the coefficients and calculate the test errors, I used lambda = 4.791278 since this was the largest value of lambda where the error was still within one standard error of the minimum MSE.

Results

Model	Test MSE	Standard Error of Test MSE
1	3111.265	361.0908
2	3095.483	369.7526
3	3095.483	369.7526
4	3070.870	350.5467
5	2920.041	346.2248

Model 1 (least squares regression containing all ten predictors) resulted in the worst test MSE, although its test MSE standard error was slightly better than the test MSE standard error for Models 2 and 3. As mentioned earlier, Model 2 (best subset selection using BIC) and Model 3 (best subset selection using 10-fold cross-validation) resulted in the same six-variable model with identical coefficient estimates and test errors. Model 4 (ridge regression using 10-fold cross-validation) performed slightly better than Models 1, 2, and 3. It, however, is a more complex model than Models 2 and 3 because it uses all ten predictors rather than just six. Model 5 (lasso using 10-fold cross-validation) resulted in the smallest test MSE and test MSE standard error of all the models. It contains only six predictors. Because of its small test errors and relative simplicity, Model 5 can be considered the best model.

Conclusion

After fitting various machine learning modeling techniques, including least squares regression, best subset selection using BIC, best subset selection using 10-fold cross validation, ridge regression using 10-fold cross validation, and lasso using 10-fold cross validation, I discovered that the lasso model using 10-fold cross validation performed best in predicting the progression of diabetes one year after baseline. This model contained the predictor variables sex, bmi, map, hdl, ltg, and glu. Before deploying the model, I recommend further model testing and consulting a diabetes expert to determine if the predictors included in the model make medical sense. If the model performs well in additional tests and receives the approval of a medical expert, it should be safe to deploy the model on new data.

Appendix A: R Code

```
# Individual Project 1
# Andrea Bruckner
# Predict 422, Sec 56

# Load the diabetes data
install.packages("lars")
library(lars)
data(diabetes)
data.all <- data.frame(cbind(diabetes$x, y = diabetes$y))

# Get overview of data
summary(data.all) # no missing data
str(data.all) # all numeric
head(data.all)
class(data.all) # data.frame
names(data.all)
nrow(data.all) # 442 rows
ncol(data.all) # 11 variables
```

```
# Partition the patients into two groups: training (75%) and test (25%)
n <- dim(data.all)[1] # sample size = 442
set.seed(1306) # set random number generator seed to enable

# repeatability of results
test <- sample(n, round(n/4)) # randomly sample 25% test
data.train <- data.all[-test,]
data.test <- data.all[test,]

# (Use these starting in Models 3-5)
x <- model.matrix(y ~ ., data = data.all)[,-1] # define predictor matrix
# excl intercept col of 1s
x.train <- x[-test,] # define training predictor matrix
x.test <- x[test,] # define test predictor matrix
y <- data.all$y # define response variable
y.train <- y[-test] # define training response variable
y.test <- y[test] # define test response variable
n.train <- dim(data.train)[1] # training sample size = 332
n.test <- dim(data.test)[1] # test sample size = 110


# 1.) Least squares regression model using all ten predictors (R function lm).

lm.fit <- lm(y~., data = data.train)
summary(lm.fit)
par(mfrow = c(2, 2))
plot(lm.fit)

# Coefficient Estimates
coef(lm.fit)

# Calculate MSE == 3111.265
mean((data.test$y - predict(lm.fit, data.test))^2) # 3111.265

lm.pred <- predict(lm.fit, data.test)
mean((data.test$y - lm.pred)^2)

# SE of MSE == 361.0908
sd((lm.pred - data.test$y)^2)/sqrt(length((lm.pred - data.test$y)^2)) # 361.0908


# 2.) Apply best subset selection using BIC to select the number of predictors (R function
regsubsets in package leaps).
# applying best subset using either 8 or 10 variables results in the same best model

library(leaps)
fit.bestsub <- regsubsets(y ~ ., data = data.train, nvmax=10) # fit the model
fit.bestsub.sum <- summary(fit.bestsub)
names(fit.bestsub.sum)
fit.bestsub.sum$bic # -201.1269 is lowest BIC, M6
```

```
summary(fit.bestsub)
plot(fit.bestsub)

which.min(fit.bestsub.sum$bic) # 6

# resets plots window
par(mfrow = c(1, 1))

plot(fit.bestsub.sum$bic,xlab="Number of Variables",ylab="BIC",type="l")
points (6, fit.bestsub.sum$bic [6], col = "red",cex = 2, pch = 20)

# Coefficient Estimates
coef(fit.bestsub,6) # plot bottoms out at 6/has smallest BIC at 6 variables

# Calculate MSE == 3095.483
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
}

bestsub.pred <- predict(fit.bestsub,data.test,id=6)
mean((data.test$y-bestsub.pred)^2) # 3095.483
# I think I can also use y.test instead of data.test$y

# SE of MSE == 369.7526
sd((bestsub.pred - data.test$y)^2)/sqrt(length((bestsub.pred - data.test$y)^2)) # 369.7526
sd((bestsub.pred - data.test$y)^2)/sqrt(n.test) # 369.7526


# 3.) Apply best subset selection using 10-fold cross-validation to select the number of
predictors (R function regsubsets in package leaps). [Use a random number seed of 1306
before entering the command: folds <- sample(1:k, nrow(data.train), replace = TRUE).]

k=10
set.seed(1306)
folds <- sample(1:k, nrow(data.train), replace = TRUE)
cv.errors<-matrix (NA,k,10, dimnames=list(NULL, paste(1:10)))

for(j in 1:k){
  cv10.fit<-regsubsets(y ~ ., data = data.train[folds !=j,],
                       nvmax =10)
  for(i in 1:10) {
    pred=predict(cv10.fit, data.train[folds==j,], id=i)
    cv.errors[j,i]=mean((data.train$y[folds==j]-pred)^2)
  }
}

mean.cv.errors=apply(cv.errors, 2, mean)
mean.cv.errors # M6 has smallest error

par(mfrow =c(1,1))
plot(mean.cv.errors, type='b') # 6 variable model is best

cv10.best <- regsubsets(y ~ ., data = data.train, nvmax=10)

# Coefficient Estimates
coef(cv10.best, 6)
#coef(cv10.best, 10)

# Calculate MSE == 3095.483 --- same as # 2 MSE
cv10.pred <- predict(cv10.best, data.test, id=6)
mean((cv10.pred-y.test)^2)
mean((cv10.pred-data.test$y)^2)

# SE of MSE == 369.7526 --- same as # 2 MSE
sd((cv10.pred-data.test$y)^2)/sqrt(n.test) # 369.7526


# 4.) Ridge regression modeling using 10-fold cross-validation to select the largest value of λ
such that the cross-validation error is within 1 standard error of the minimum (R functions
glmnet and cv.glmnet in package glmnet). [Use a random number seed of 1306 immediately
before entering the command: cv.out <- cv.glmnet(x.train, y.train, alpha = 0).]
# pages 266-269

grid = 10^seq(10,-2, length = 100)
ridge.fit0 = glmnet(x.train, y.train, alpha = 0, lambda = grid)
plot(ridge.fit0,xvar="lambda",label=T)
# is grid even necessary? -- not when you use cv.glmnet

library (glmnet)
k=10
set.seed(1306)
cv.out <- cv.glmnet(x.train, y.train, alpha = 0)
names(cv.out)
plot(cv.out)
#plot(cv.out, sign.lambda = -1)
# example of how to interpret this plot: http://gerardnico.com/wiki/r/ridge_lasso
#plot(cv.out$glmnet.fit) # not sure what to make of this...
names(cv.out)

largelam <- cv.out$lambda.1se
largelam # lambda = 41.67209

ridge.fit <- glmnet(x.train,y.train,alpha=0,lambda=41.67209)
#plot(ridge.fit, xvar="lambda", label=T)
# the plot does not look right...

# Coefficient Estimates
coef(ridge.fit)
```

```
ridge.pred <- predict(ridge.fit,newx = x.test)

# Calculate MSE == 3070.87
mean((ridge.pred-y.test)^2) # 3070.87


# SE of MSE == 350.5467
sd((ridge.pred-y.test)^2)/sqrt(n.test) # 350.5467
```

# 5.) Lasso model using 10-fold cross-validation to select the largest value of λ such that the cross-validation error is within 1 standard error of the minimum (R functions glmnet and cv.glmnet in package glmnet). [Use a random number seed of 1306 immediately before entering the command: cv.out <- cv.glmnet(x.train, y.train, alpha = 1).]

```
library (glmnet)
k=10
set.seed(1306)
cv.out <- cv.glmnet(x.train, y.train, alpha = 1)
plot(cv.out)
names(cv.out)

largelam.lasso <- cv.out$lambda.1se
largelam.lasso # 4.791278

lasso.fit <- glmnet(x.train,y.train,alpha=1,lambda=4.791278)
#plot(lasso.fit) # doesn't look right...

# Coefficient Estimates
coef(lasso.fit)

# Calculate MSE == 2920.041
lasso.pred <- predict(lasso.fit,newx=x.test)
mean((lasso.pred-y.test)^2) # 2920.041

# SE of MSE == 346.2248
sd((lasso.pred-y.test)^2)/sqrt(n.test) # 346.2248
```