Final Project: Loan Approval

Data 621 Business Analytics and Data Mining

Aadi Kalloo, Nathan Lim, Asher Meyers, Daniel Smilowitz, Logan Thomson

Due July 21, 2016

Contents

Abstract

Keywords:

Introduction

Between 2009-2014, approximately 95 million mortgage loan applications were filed in the United States alone (Home Mortgage Disclosure Act, 2015). When further loan types, including education and personal loans, are considered, it becomes even clearer that the ability to accurately identify reliable borrowers is a high priority for banks and other financial institutions today. Likewise, consumers are significantly affected as well. Surveys and analyses performed by Lusardi & Scheresberg (2013) has shown that individuals of poor financial literacy, and subsequently poor credit history, are declined by banks for loans and mortgages, and turn to high-cost borrowing (i.e. payday loans). Thus, it becomes clear that the factors that determine which individuals are dependable borrowers is information important to both financial institutions and consumers alike.

The dataset of interest contains information about customers of a hypothetical bank. The data for this project was obtained from a data challenge hosted at AnalyticsVidhya The dataset has 614 rows (each representing a customer) and 13 predictor variables variables. Furthermore, there is 1 identification variable, and 1 response variables: `Loan_Status`, a binary categorical variable representing whether each customer has been approved for a loan given their credentials. The class of variables read in from the dataset is presented below:

	Class	Levels/Range
Married	numeric	0 - 1
Dependents	numeric	0 - 3
Education	numeric	0 - 1
Self_Employed	numeric	0 - 1
ApplicantIncome	integer	150 - 81000
CoapplicantIncome	numeric	0 - 41667
LoanAmount	integer	9 - 700
Loan_Amount_Term	integer	12 - 480
Credit_History	integer	0 - 1
Property_Area	factor	3
Loan_Status	numeric	0 - 1
Urban	numeric	0 - 1
SemiUrban	numeric	0 - 1
TotalIncome	numeric	1442 - 81000
LTI	numeric	3 - 900

Literature Review

Although predictors can vary by loan or income level, studies investigating loan approval predictors have found that factors such as Marital Status, Loan Duration, and Number of Dependents can serve as statistically significant predictors of loan approval status (Agbemava et al., 2016). However, as loan approval criteria can vary greatly between financial institutions there is a significant lack of published materials investigating the effects that various factors may play on loan approval status. Furthermore, the majority of financial institutions use proprietary algorithms and statistical methods in order to determine loan approvals.

It should be noted that ratio of training set cases to test set cases is 1.67:1. This naturally leads to markedly decreased algorithm training than the 4:1 ratio that is normally used in prediction tasks (Dobin & Simon, 2011).

Of significant interest to us were the concepts of additive logistic regression or "boosting" as outlined by Friedman et al. (2012). The original concept of boosting was introduced by Schapire (1990) and describes a method of increasing the performance of "weak" classifiers by combining many of them into a more powerful model (Friedman et al., 2012). The authors of this paper introduce the Adaboost algorithm, which is used and analyzed in detail for this project. The Adaboost algorithm is characterized by the use of weights on the target variable, increasing the weight on each iteration for misclassified values (Friedman et al., 2012).

Given that this project was centered around a data challenge, it was also of particular significance to us to research

data mining and machine learning methods that could potentially provide greater accuracy scores, despite being outside the scope of IS621 Data Mining. The primary method of prediction in this arena is the neural network. While the foundational ideas of artificial neural network date as far back as the 1940s (Warren & Pitts, 1943), the procedures used by the `neuralnet` package are based on Riedmiller and Braun (1993). As a brief overview, the `neuralnet` package calculates the following function as described by Gunther & Fritsch (2010):

$$o(x) = f(w_0 + \sum_{j=1}^{J} w_j \cdot f(w_{0j} + \sum_{i=1}^{n} w_{ij} x_i))$$

where w_0 is the intercept of the output neuron, w_{0j} is the intercept of the jth hidden neuron, and w_j is the synaptic weight. This mathematical model can be expressed visually, and is depicted in the following figure:
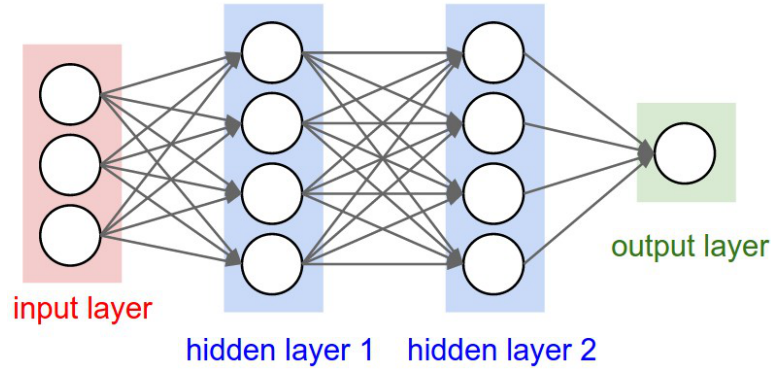
Methodology

In total, six models were created and used for prediction of the target variable. A tool was supplied by AnalyticsVidhya that reported accuracy (% of cases correctly predicted) upon submission of the test data predictions.

Given the variables in the data, two new features were created: `Total Income` and `LTI` (Loan to Income Ratio). The `Urban` and `SemiUrban` variables were created from the `Property_Type` variable, and the `Property_Type` varible was excluded from further analyses. As all other data were numeric, or binary factors converted to numeric, these data worked well for logistic regression.

Logistic regression is a very common method of classifying data (Thomas, 2000). In logistic regression, a linear regression is used where the dependent variable is a non-linear function of the probability of the event occuring. In this project, the purpose of the logistic regression method is to classify cases as either "Approved for loan" or "Denied". The general logistic regression model can be represented as:

$$\ln \frac{p_i}{1 - p_i} = \beta_0 + \beta_1 X_1 + ... + \beta_n X_n + \varepsilon$$

A few different logistic regression models were developed for this project. The first was a full model using all variables. This model made use of the `glm` function.

The second model created utilized both the `glm` function and the `leaps` package in order to find the best combination/subset of variables that minimizes the Bayesian Information Criterion (BIC). The "best subsets" approach is a method of automated variable selection that seeks to find the best predictors of the target variable. The `regsubsets` function as part of the `leaps` package found the `Married`, `Credit_History` and `SemiUrban` variables to be the best subset of predictors for this data set.

Ridge regression methods were combined with logistic regression to create a "Logistic Ridge" model, the third of this project, using the `ridge` library. As this library is no longer actively in development, it was manually obtained and installed. Like Ridge Regression, Logistic Ridge Regression introduces a tuning parameter as a penalty to avoid overfitting (Cule & De Iorio, 2012). This is useful as the ridge regression method aims to reduce variance and

3

produce lower mean squared error for higher-dimensional data such as that used in the full model for these data. The ridge penalty used in the `ridge` package as outlined by Cule & De Iorio (2012) is given as:

$$k = \frac{p}{\hat{\beta}'\hat{\beta}}$$

The Adaboost algorithm was used for this project in two forms. The first was through the use of the `ada` package. For this second implementation, the effect of the number of iterations of the algorithm on the final accuracy of the test set was analyzed with iterations in the range of 50-525 with steps of 25. It is hypothesized that accuracy will gradually increase with iterations to a peak, then gradually decrease as overfitting starts to become a prominent issue. The second was a manually coded implementation based on the algorithmic outline provided by Friedman et al. (2000). A manual approach was used for the purposes of comparing the academically published to the packaged algorithm. Given that these algorithms work by combining basic models, it is believed that they will perform better than a single logistic regression model alone.

A simple neural network method was pursued. The neural network approach was performed using normalized data with a single hidden layer. The number of nodes to use in the hidden layer was determined using a 10-fold Cross Validation Approach. A plot of the mean CV error against the number of hidden nodes is shown below:

Determination of Number of Nodes for neuralnet
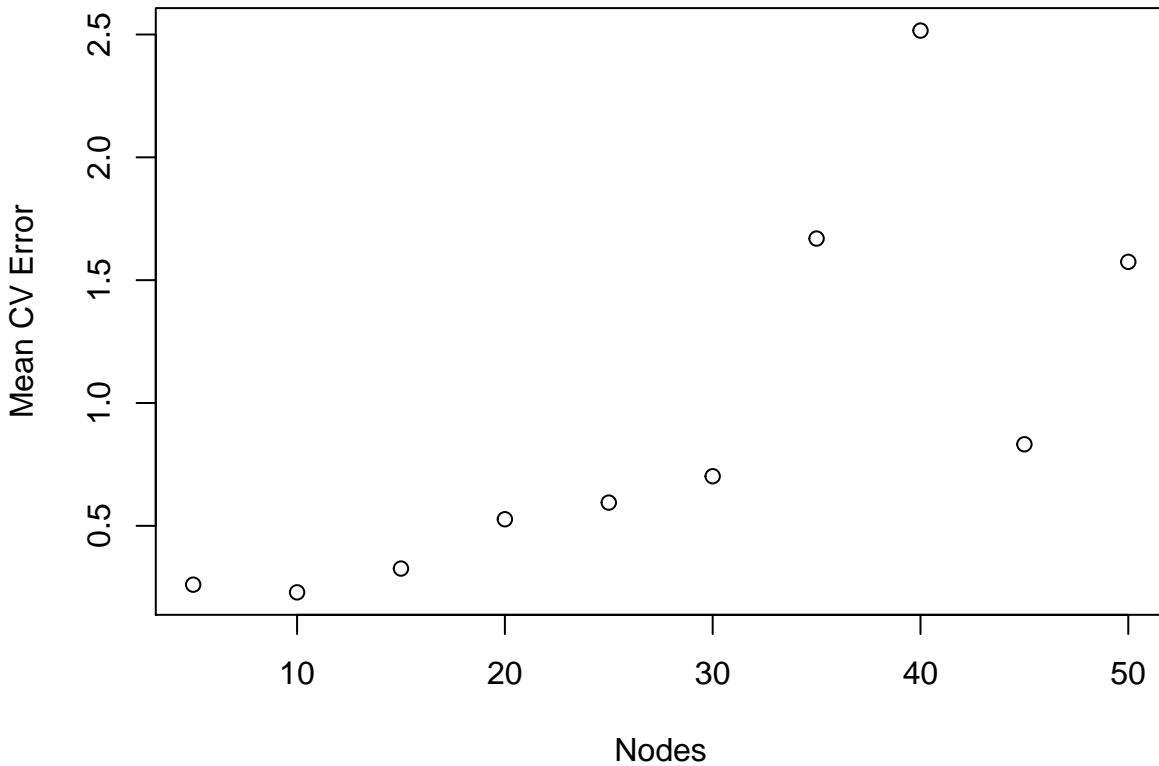
From the above data, a network of 10 nodes was chosen due to its minimal cross validation error and the illustration of the network used for prediction of the target variable is included here:
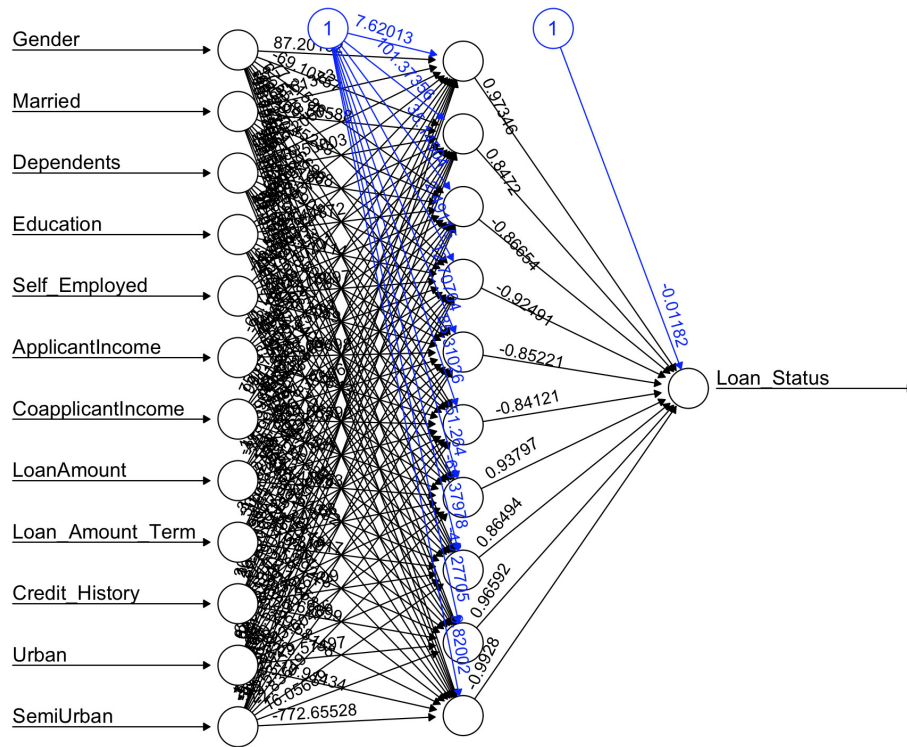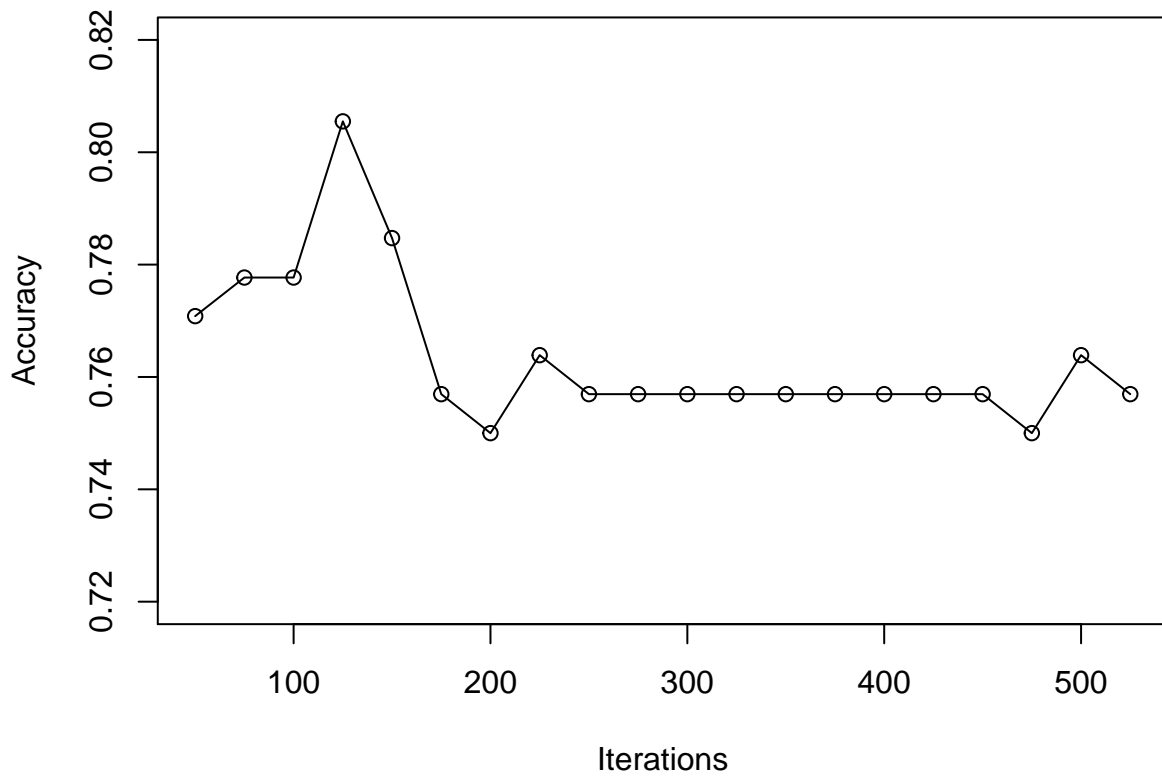
Results

Accuracy

The results obtained are summarized as follows:

	Model	Accuracy
1	ada package	0.8055
2	Best Subsets	0.7778
3	Logistic Ridge	0.7708
5	Full Model, Poisson	0.7708
4	Full Model, Binomial	0.7638
6	Neural Network (neuralnet package)	0.7431
7	Manually implemented adaboost algorithm	0.7153

It can be seen that the best results were achieved through use of the `ada` package. The following figure illustrates the results of an analysis of test data accuracy against iterations of the `ada` function:

Effect of number of iteratons on test data accuracy

Optimal accuracy was attained at 125 iterations, with a steady decline and plateau thereafter. These results were in keeping with our hypothesis, however it was expected that the peak accuracy would occur at a higher iteration.

ROC Curves

ROC curves for the linear regression models were created and are displayed here.
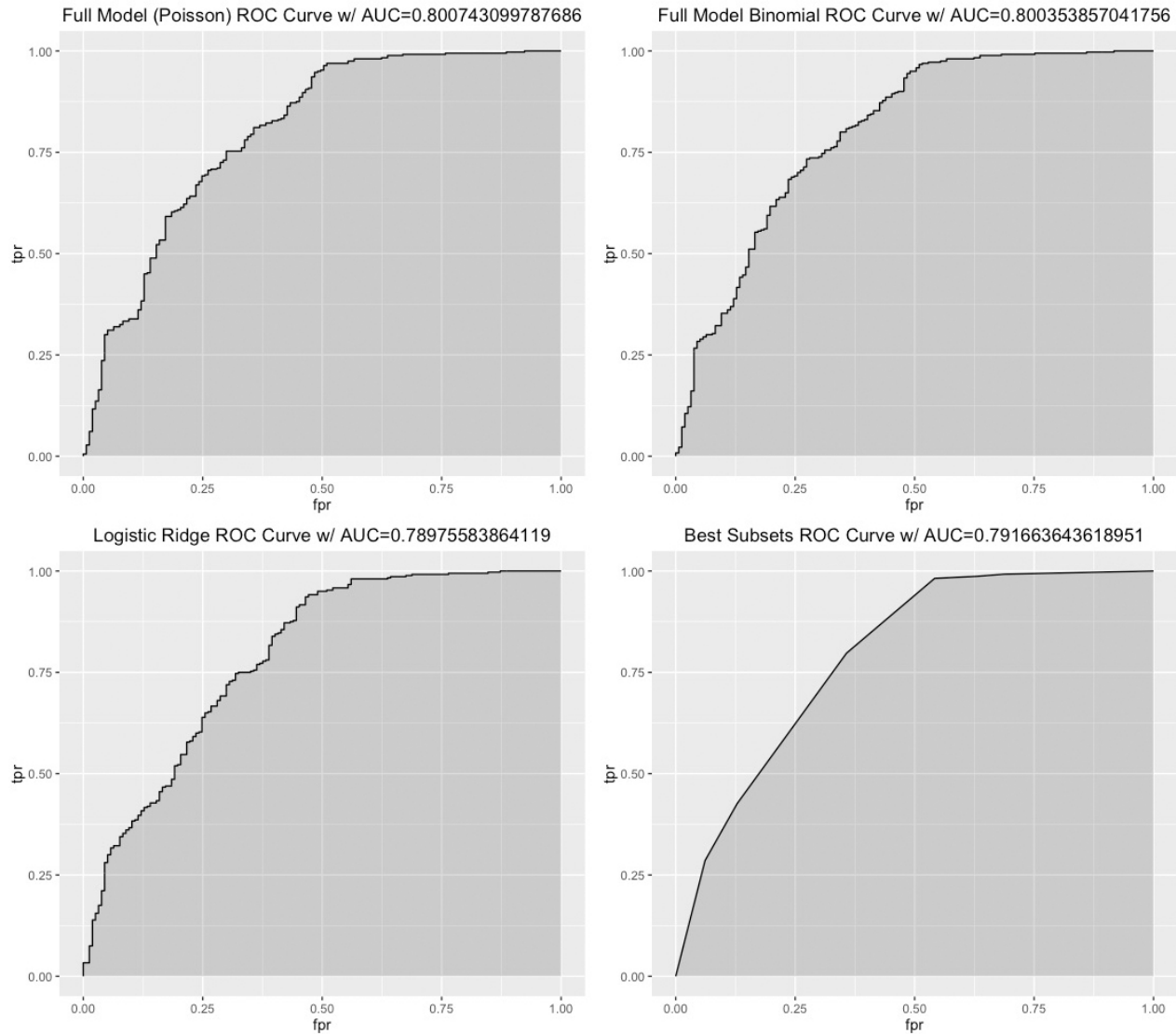
10-Fold Cross-Validation

The 10-Fold Cross-Validation approach was used again to determine which model produced the lowest error on unseen training data. This data is compared in the section below to the accuracy on test set data. A summary of the mean CV error for each model is shown here:

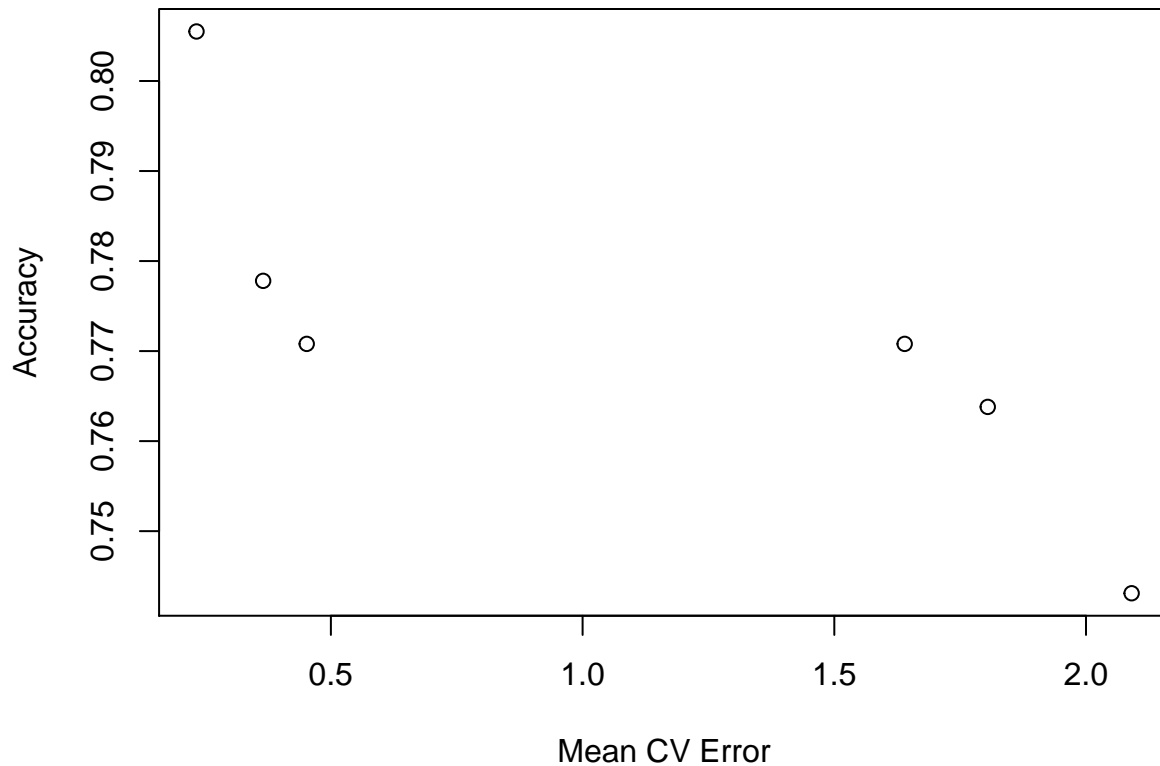| Model	Mean.CV.Error
Logistic Ridge Regression Model	0.2331
Neural Network	0.3654
Additive (Boosted) Logistic Regression Model	0.452
Best Subsets Logistic Regression Model	1.64
Full Logistic Regression Model	1.805
Full Regression Model, Poisson	2.091

Discussion

Many data mining methods were explored for this project. It was determined that the additive logistic regression algorithm (`ada` package) produced the best results, with the best subsets logistic regression coming in second. It is believed that the advantage of the `ada` package is due to its ability to combine a number of simpler models. Our

hypothesis regarding the manually implemented Adaboost algorithm was incorrect, as it resulted in lower accuracy than the best subsets regression model.

Here the relationship between the Cross Validation error on training data and accuracy on the test data is shown:

Relationship between Training Set CV Error and Test Set Accuracy

While this plot does indicate a general negative relationship between mean CV error and test data accuracy (i.e. as CV error increases, test accuracy decreases), the sample is too small to determine whether or not this data is linear or follows a higher order relationship.

The performance of the best subsets logistic regression model was surprising given its reliance on only three variables (`Married`, `Credit_History` and `SemiUrban`) for prediction of the target variable, `Loan_Status`. Despite this strong performance, the best subsets logistic regression model alone resulted in an overall leaderboard rank of 161.

At the time of writing this report, we are currently tied for 8th place out of a total of 237 participants. In subsequent trials and analyses, this project would benefit greatly from further experimentation with more advanced additive logistic regression models and with neural network architectures.

References

Agbemava, E., Nyarko, I. K., Adade, T. C., & Bediako, A. K. (2016). Logistic Regression Analysis Of Predictors Of Loan Defaults By Customers Of Non-Traditional Banks In Ghana. European Scientific Journal, 12(1).

Cule, E., & De Iorio, M. (2012). A semi-automatic method to guide the choice of ridge parameter in ridge regression. arXiv preprint arXiv:1205.0686.

Dobbin, K. K., & Simon, R. M. (2011). Optimally splitting cases for training and testing high dimensional classifiers. BMC medical genomics, 4(1), 1.

Günther, F., & Fritsch, S. (2010). neuralnet: Training of neural networks. The R journal, 2(1), 30-38.

"Home Mortgage Disclosure Act." Consumer Financial Protection Bureau: An Official Website of the United States Government. Consumer Financial Protection Bureau, n.d. Web. 17 July 2016. http://www.consumerfinance.gov/data-research/hmda/.

Lusardi, A., & Scheresberg, C. D. B. (2013). Financial literacy and high-cost borrowing in the United States (No. w18969). National Bureau of Economic Research.

Thomas, L. C. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. International journal of forecasting, 16(2), 149-172.

Appendix – R Code

```r
library(knitr)
opts_chunk$set(echo=FALSE, warning=FALSE, message=FALSE, comment=NA, fig.align='center')
knitr::opts_chunk$set(error = TRUE)

library(stringr) #For string functions
library(glmnet) #For binary logistic regression
library(leaps) #For best subsets
library(pROC) #For ROC curve
library(car)
library(MASS)
library(ROCR)
library(ggplot2)
library(stringr)
library(dplyr)
library(reshape2)
library(vcd)
library(pander)
library(tidyr)
library(e1071)

train <- read.csv("https://github.com/dsmilo/DATA621/raw/master/Final-Project/loan_train.csv", string
test = read.csv("https://github.com/dsmilo/DATA621/raw/master/Final-Project/loan_test.csv", stringsAs
test2 = test

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
}

plotROC <- function(model, ndata, gtruth) {
  prob <- predict(model, newdata = ndata, type="response")
  pred <- prediction(prob, gtruth)
  perf <- performance(pred, measure = "tpr", x.measure = "fpr")
  auc <- performance(pred, measure = "auc")
  auc <- auc@y.values[[1]]
  roc.data <- data.frame(fpr=unlist(perf@x.values), tpr=unlist(perf@y.values), model="GLM")
  ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) + geom_ribbon(alpha=0.2) + geom_line(aes(y=tpr)) + g
}

fillwithmedian <- function(x) {
  median_val = median(x, na.rm = TRUE)
  x[is.na(x)] = median_val
  return(x)
}

train3 = train
test2 = test
```

```r
train3$Gender = ifelse(train$Gender == "Male", 1, 0)
train3$Married = ifelse(train$Married == "Yes", 1, 0)
train3$Dependents[train3$Dependents == "3+" ] = 3
train3$Dependents = as.numeric(train3$Dependents)
train3$Education = ifelse(train$Education == "Graduate", 1, 0)
train3$Self_Employed = ifelse(train$Self_Employed == "Yes", 1, 0)

train3$Urban = ifelse(train$Property_Area == "Urban", 1, 0)
train3$SemiUrban = ifelse(train$Property_Area == "Semiurban", 1, 0)
train3$TotalIncome = train3$ApplicantIncome + train3$CoapplicantIncome
train3$LTI = (train3$LoanAmount*1000)/train3$ApplicantIncome #Loan to income ratio

train3$Loan_Status = ifelse(train$Loan_Status == "Y", 1, 0)


test2$Gender = ifelse(test$Gender == "Male", 1, 0)
test2$Married = ifelse(test$Married == "Yes", 1, 0)
test2$Dependents[test2$Dependents == "3+" ] = 3
test2$Dependents = as.numeric(test2$Dependents)
test2$Education = ifelse(test$Education == "Graduate", 1, 0)
test2$Self_Employed = ifelse(test$Self_Employed == "Yes", 1, 0)

test2$Urban = ifelse(test$Property_Area == "Urban", 1, 0)
test2$SemiUrban = ifelse(test$Property_Area == "Semiurban", 1, 0)
test2$TotalIncome = test2$ApplicantIncome + test2$CoapplicantIncome
test2$LTI = (test2$LoanAmount*1000)/test2$ApplicantIncome #Loan to income ratio

train4 = train3[complete.cases(train3),]

train = train3[, -1]
train2 = train4
#remove(train4)
#remove(train3)

test = test2
#remove(test2)



##################################
#Full Model
glmflagfull <- glm(Loan_Status~.-Property_Area, data = train, family = poisson, control = list(maxit
summary(glmflagfull)
vif(glmflagfull) #Check for collinearity, VIF > 10
predglmflagfull <- data.frame("class" = train$Loan_Status, "logit" = predict(glmflagfull, train))
#pROC::roc(class ~ logit, data = predglmflagfull, auc = TRUE, plot = TRUE, smooth =  TRUE)

plotROC(model = glmflagfull, ndata = train, gtruth = train$Loan_Status, name = "Full Model (Poisson)"

a = predict(glmflagfull, test, type = "response")
testdf = data.frame(Loan_ID = test$Loan_ID, Loan_Status = a)
testdf$Loan_Status[is.na(testdf$Loan_Status)] = median(testdf$Loan_Status, na.rm = TRUE)
```

```r
testdf$Loan_Status = ifelse(testdf$Loan_Status > 0.5, "Y", "N")

write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()), ".csv"))
#SCore: 0.770833333333
###################

############ Best Subsets
regfit.full <- regsubsets(Loan_Status~.-Property_Area -Loan_ID, data = train3, nvmax = 17)
summary(regfit.full)
par(mar=c(1,1,1,1))
par(mfrow = c(1, 2))
plot(regfit.full, scale = "bic", main = "Predictor Variables vs. BIC")
reg.summary <- summary(regfit.full)
reg.summary$bic
plot(reg.summary$bic, xlab = "Number of Predictors", ylab = "BIC", type = "l", main = "Best Subset Se
minbic <- which.min(reg.summary$bic)
points(minbic, reg.summary$bic[minbic], col = "brown", cex = 2, pch = 20)
coef(regfit.full, minbic)
var_names = names(coef(regfit.full, minbic))[2:length(names(coef(regfit.full, minbic)))]
length(var_names)

Model_toEval = paste0("glm(Loan_Status ~ ", paste(var_names, collapse = " + "), ", data = train, fami

bestsubset21 = eval(parse(text = Model_toEval))

summary(bestsubset21)

plotROC(model = bestsubset21, ndata = train, gtruth = train$Loan_Status, name = "Best Subsets")

a = predict(bestsubset21, test, type = "response")
testdf = data.frame(Loan_ID = test$Loan_ID, Loan_Status = a)
testdf$Loan_Status[is.na(testdf$Loan_Status)] = median(testdf$Loan_Status, na.rm = TRUE)
testdf$Loan_Status = ifelse(testdf$Loan_Status > 0.5, "Y", "N")
#Score = 0.777777777778
####################################

############ Logistic Ridge
library(ridge) #had to install ridge package manually
train$Loan_Status = as.numeric(as.character(train$Loan_Status))
train_lr <- data.frame(lapply(train[, -11], fillwithmedian))
train_lr$Property_Area = train$Property_Area

lr2 = logisticRidge(Loan_Status ~ . -Property_Area, data = train_lr)




plotROC(model = lr2, ndata = train, gtruth = train$Loan_Status, name = "Logistic Ridge")

a = predict(lr2, test2, type = "response")
testdf = data.frame(Loan_ID = test$Loan_ID, Loan_Status = a)
testdf$Loan_Status[is.na(testdf$Loan_Status)] = median(testdf$Loan_Status, na.rm = TRUE)
```

```r
testdf$Loan_Status = ifelse(testdf$Loan_Status > 0.5, "Y", "N")
write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()), ".csv"))

#Score: 0.770833333333


############################

#############Full model binomial
glmflagfull <- glm(Loan_Status~.-Property_Area, data = train, family = binomial, control = list(maxit
summary(glmflagfull)
vif(glmflagfull) #Check for collinearity, VIF > 10
predglmflagfull <- data.frame("class" = train$Loan_Status, "logit" = predict(glmflagfull, train))
#pROC::roc(class ~ logit, data = predglmflagfull, auc = TRUE, plot = TRUE, smooth =  TRUE)

plotROC(model = glmflagfull, ndata = train, gtruth = train$Loan_Status, name = "Full Model Binomial")

a = predict(glmflagfull, test, type = "response")
testdf = data.frame(Loan_ID = test$Loan_ID, Loan_Status = a)
testdf$Loan_Status[is.na(testdf$Loan_Status)] = median(testdf$Loan_Status, na.rm = TRUE)
testdf$Loan_Status = ifelse(testdf$Loan_Status > 0.5, "Y", "N")
write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()), ".csv"))
#Score: 0.763888888889
############################

##########Determination of nodes for neuralnet
k = 10
set.seed(1306)
folds = sample(1:k, nrow(train), replace = TRUE)
#folds_amt = sample(1:k, nrow(train_amt), replace = TRUE)
cv.errors1 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors2 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors3 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors4 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors5 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors6 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors7 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors8 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))

train_2 <- data.frame(lapply(train[, -11], fillwithmedian))
train_2$Property_Area = train$Property_Area
scaled = data.frame(lapply(scaled, fillwithmedian))

#train_flag$target = as.numeric(as.character(train_flag$target))
num_layers = 0

f1 <- as.formula(paste("Loan_Status ~", paste(n[!n %in% "Loan_Status"], collapse = " + ")))
for (j in 1:k) {

  num_layers = num_layers + 5
  print(num_layers)
#
#   model1 <- glm(Loan_Status~.-Property_Area, data = train_2[folds != j, ], family = poisson, contro
```

```r
#   model2 <- glm(formula = Loan_Status ~ Married + Credit_History + SemiUrban, family = binomial(lir
#   model3 <- logisticRidge(Loan_Status ~ . -Property_Area, data = train_2[folds != j, ])
#   model4 <- glm(Loan_Status~.-Property_Area, data = train_2[folds != j, ], family = binomial, contr
  dat = scaled[folds != j, ]
  dat =dat[complete.cases(dat),]
  model5 <- neuralnet(f1, data=dat, hidden=c(num_layers), linear.output=TRUE, stepmax = 10000000)
  #model6 <- ada(formula = Loan_Status ~ ., data = train_x1[folds != j, ], iter = 125, loss = "logist
  #model7 <- glm(TARGET_FLAG ~ KIDSDRIV + INCOME + PARENT1 + HOME_VAL + MSTATUS + TRAVTIME + CAR_USE
  #model8 = glm(TARGET_FLAG ~ KIDSDRIV + INCOME + PARENT1 + HOME_VAL + MSTATUS + TRAVTIME + CAR_USE


  #best.fit = regsubsets(y ~ ., data = train_2_df[folds != j, ], nvmax = 10)
  for (i in 1:10) {
    print(i)
    f = train_2[folds == j, ]
    f = f[complete.cases(f),]

    # pred1 = predict(model1, f, id = i)
    # cv.errors1[j, i] = mean((train_2$Loan_Status[folds == j] - pred1) ^ 2, na.rm = TRUE)
    #
    # pred2 = predict(model2, f, id = i)
    # cv.errors2[j, i] = mean((train_2$Loan_Status[folds == j] - pred2) ^ 2, na.rm = TRUE)
    #
    # pred3 = predict(model3, f, id = i)
    # cv.errors3[j, i] = mean((train_2$Loan_Status[folds == j] - pred3) ^ 2, na.rm = TRUE)
    #
    # pred4 = predict(model4, f, id = i)
    # cv.errors4[j, i] = mean((train_2$Loan_Status[folds == j] - pred4) ^ 2, na.rm = TRUE)
    #
    dat = scaled[folds == j, ]
    #dat = dat[complete.cases(dat),]
    dat2 = dat[complete.cases(dat), -11]
    pred5 = neuralnet::compute(model5, dat2)
    netresult = pred5$net.result[, 1]
    sc = scaled$Loan_Status[folds == j]
    sc = sc[complete.cases(sc)]
    cv.errors5[j, i] = mean((sc - netresult) ^ 2, na.rm = TRUE)

    # pred6 = predict(model6, train_x1[folds == j, ], id = i, type = "probs")
    # cv.errors6[j, i] = mean((train_2$Loan_Status[folds == j] - pred6) ^ 2, na.rm = TRUE)
    #
    # pred7 = predict(model7, train_2_flag[folds == j, ], id = i)
    # cv.errors7[j, i] = mean((train_2_flag$TARGET_FLAG[folds == j] - pred7) ^ 2, na.rm = TRUE)
    #
    # pred8 = predict(model8, train_2_flag[folds == j, ], id = i)
    # cv.errors8[j, i] = mean((train_2_flag$TARGET_FLAG[folds == j] - pred8) ^ 2, na.rm = TRUE)
  }

}

cv_means = rowMeans(cv.errors5)
node_counts = seq(5, 50, 5)
```

```r
nn_df = data.frame(Nodes = node_counts, CV_Error = cv_means)
names(nn_df) = c("Nodes", "Mean CV Error")
write.csv(nn_df, "~/Downloads/nn_df.csv", row.names = FALSE)




#########################


###########Neural network
train4 = train3
train4 = train4[complete.cases(train4),]
train4 = train4[, -c(1,12)]
maxs <- apply(train4, 2, max)
mins <- apply(train4, 2, min)
scaled <- as.data.frame(scale(train4, center = mins, scale = maxs - mins))
n <- names(scaled)
f <- as.formula(paste("Loan_Status ~", paste(n[!n %in% "Loan_Status"], collapse = " + ")))
nn = neuralnet(f, data=scaled, hidden=c(10), linear.output=TRUE, stepmax = 1000000)
plot(nn)


# pr.nn <- compute(nn, scaled[,-11])
# pr.nn_ <- pr.nn$net.result*(max(train4$Loan_Status)-min(train4$Loan_Status))+min(train4$Loan_Status
# #test.r <- (test_$medv)*(max(data$medv)-min(data$medv))+min(data$medv)
# pr_nn = ifelse(pr.nn_ > 0.5, 1, 0)

max2 <- function(x) {
  return(max(x, na.rm = TRUE))
}

min2 <- function(x) {
  return(min(x, na.rm = TRUE))
}

test <- data.frame(lapply(test, fillwithmedian))
test = test2
test = test[, -c(1, 12)]
test <- data.frame(lapply(test, fillwithmedian))


maxs.t <- apply(test, 2, max2)
mins.t <- apply(test, 2, min2)
scaled.t <- as.data.frame(scale(test, center = mins.t, scale = maxs.t - mins.t))
scaled.t = as.data.frame(lapply(scaled.t, fillwithmedian))
pr.test <- neuralnet::compute(nn, scaled.t)
pr.t_ <- pr.test$net.result*(max(train4$Loan_Status)-min(train4$Loan_Status))+min(train4$Loan_Status)
#test.r <- (test_$medv)*(max(data$medv)-min(data$medv))+min(data$medv)
pr_nn.t = ifelse(pr.t_ > 0.5, "Y", "N")
#pr_nn.t = fillwithmedian(pr_nn.t)

test1 = read.csv("https://github.com/dsmilo/DATA621/raw/master/Final-Project/loan_test.csv", stringsA
```

```r
df = data.frame(Loan_ID = test1$Loan_ID, Loan_Status = as.data.frame(as.vector(pr_nn.t)))
names(df) = c("Loan_ID", "Loan_Status")
sum(is.na(df[, 2])) #hopefully 0
write.csv(df, paste0("nn-", as.numeric(Sys.time()), ".csv"), row.names = FALSE)
#Score: 0.743055555556
#####################################

############### ada package analysis
model1 = list()


for (i in 1:20) {
  iters = 25 + 25*i
  model1[[i]] = ada(formula = Loan_Status ~ ., data = train_x1, iter = iters, loss = "logistic")
  results = as.numeric(as.character(predict(model1[[i]], test)))
  results = ifelse(results == 1, "Y", "N")
  testdf = data.frame(Loan_ID = test2$Loan_ID, Loan_Status = results)
  write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()), " - ", iters, ".csv"), row.names
}

ada_results = read.csv("https://raw.githubusercontent.com/aadikalloo/AadiMSDA/master/IS621-Data-Minin
names(ada_results) = c("Iterations", "Accuracy")
###################

############### Adaboost manual implementation
wts = list()

models = list()
alpha = list()
error_rate = list()
final = 0
predictions = list()
total = 12



train_x <- data.frame(lapply(train[-c(1,2), -11], fillwithmedian))
test_x = data.frame(lapply(test2[, -c(1, 12)], fillwithmedian))

k = total
set.seed(1306)
folds = sample(1:k, nrow(train_x), replace = TRUE)
folds = rep(1:12, nrow(train_x)/12)

wts[[1]] = rep(1/length(train_x1[,1]), length(train_x1[,1]))
Y_hat_run = list()
Y_hat_run[[1]] = rep(1, length(train_x1[,1]))

for (i in 1:total) {
  #train_x1 = train_x[folds != i,]
```

```
  train_x1 = train_x

  models[[i]] = glm(Loan_Status ~ ., data = train_x1, family = binomial, weights = wts[[i]])
  Y_hat = predict(models[[i]], train_x1, type = "response")
  Y_hat_run = Y_hat_run[[i]]*Y_hat
  Y_hat = ifelse(Y_hat > 0.5, 1, 0)
  error_rate[[i]] = sum(wts[[i]]*(train_x1$Loan_Status != Y_hat))/length(Y_hat)
  alpha[[i]] = (1/2)*log((1 - error_rate[[i]])/error_rate[[i]])
  loan_status = ifelse(train_x1$Loan_Status > 0, 1, -1)
  Y_hat = ifelse(Y_hat == 1, 1, -1)
  wts[[i + 1]] = wts[[i]] * exp(-alpha[[i]]*loan_status*Y_hat)
  wts[[i + 1]] = wts[[i + 1]]/sum(wts[[i + 1]])
}


for (i in 1:total) {
  predictions[[i]] = predict(models[[i]], test_x, type = "response")
  predictions[[i]] = alpha[[i]]*predictions[[i]]

  #predictions[[i]] = ifelse(predictions[[i]] > 0.5*alpha[[i]], 1, -1)
  #predictions[[i]] = ifelse(predictions[[i]] > 0.5, 1, -1)

  #final = final + predictions
}

toEval = "predictions[[1]]"

for (i in 2:4) {
  toEval = paste0(toEval, " * predictions[[", i, "]]")

}

prop = table(train$Loan_Status)[[1]]/table(train$Loan_Status)[[2]]
new = Y_hat_run[order(Y_hat_run)]
pred2 = ifelse((rank(Y_hat_run))>prop*length(test[,1]),1,0)

results = eval(parse(text = toEval))
results = ifelse(results < error_rate[[1]], "N", "Y")
#results = ifelse(results < 0, 0, 1)
#results = ifelse(results < 0, "N", "Y")


#results = ifelse(pred2 == 1, "Y", "N")
testdf = data.frame(Loan_ID = test2$Loan_ID, Loan_Status = results)

write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()), ".csv"), row.names = FALSE)
#Score: 0.715277777778
###############

#####10 fold Cross Validation

k = 10
```

```r
set.seed(1306)
folds = sample(1:k, nrow(train), replace = TRUE)
cv.errors1 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors2 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors3 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors4 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors5 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors6 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors7 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors8 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))

train_2 <- data.frame(lapply(train[, -11], fillwithmedian))
train_2$Property_Area = train$Property_Area
scaled = data.frame(lapply(scaled, fillwithmedian))

f1 <- as.formula(paste("Loan_Status ~", paste(n[!n %in% "Loan_Status"], collapse = " + ")))
for (j in 1:k) {
  print(j)

  model1 <- glm(Loan_Status~.-Property_Area, data = train_2[folds != j, ], family = poisson, control
  model2 <- glm(formula = Loan_Status ~ Married + Credit_History + SemiUrban, family = binomial(link
  model3 <- logisticRidge(Loan_Status ~ . -Property_Area, data = train_2[folds != j, ])
  model4 <- glm(Loan_Status~.-Property_Area, data = train_2[folds != j, ], family = binomial, control
  dat = scaled[folds != j, ]
  dat =dat[complete.cases(dat),]
  model5 <- neuralnet(f1, data=dat, hidden=c(5, 3), linear.output=TRUE, stepmax = 10000000)
  model6 <- ada(formula = Loan_Status ~ ., data = train_x1[folds != j, ], iter = 125, loss = "logisti

  #best.fit = regsubsets(y ~ ., data = train_2_df[folds != j, ], nvmax = 10)
  for (i in 1:10) {
    print(i)
    f = train_2[folds == j, ]
    f = f[complete.cases(f),]

    pred1 = predict(model1, f, id = i)
    cv.errors1[j, i] = mean((train_2$Loan_Status[folds == j] - pred1) ^ 2, na.rm = TRUE)

    pred2 = predict(model2, f, id = i)
    cv.errors2[j, i] = mean((train_2$Loan_Status[folds == j] - pred2) ^ 2, na.rm = TRUE)

    pred3 = predict(model3, f, id = i)
    cv.errors3[j, i] = mean((train_2$Loan_Status[folds == j] - pred3) ^ 2, na.rm = TRUE)

    pred4 = predict(model4, f, id = i)
    cv.errors4[j, i] = mean((train_2$Loan_Status[folds == j] - pred4) ^ 2, na.rm = TRUE)

    dat = scaled[folds == j, ]
    dat = dat[complete.cases(dat), -11]
    pred5 = neuralnet::compute(model5, dat2)
    netresult = pred5$net.result[, 1]
    sc = scaled$Loan_Status[folds == j]
    sc = sc[complete.cases(sc)]
```

```
      cv.errors5[j, i] = mean((sc - netresult) ^ 2, na.rm = TRUE)

      pred6 = predict(model6, train_x1[folds == j, ], id = i, type = "probs")
      cv.errors6[j, i] = mean((train_2$Loan_Status[folds == j] - pred6) ^ 2, na.rm = TRUE)

   }

}



mean.cv.errors1 <- apply(cv.errors1, 2, mean)
mean.cv.errors2 <- apply(cv.errors2, 2, mean)
mean.cv.errors3 <- apply(cv.errors3, 2, mean)
mean.cv.errors4 <- apply(cv.errors4, 2, mean)
mean.cv.errors5 <- apply(cv.errors5, 2, mean)
mean.cv.errors6 <- apply(cv.errors6, 2, mean)


all.cv.error = data.frame(
  mean(mean.cv.errors1),
  mean(mean.cv.errors2),
  mean(mean.cv.errors3),
  mean(mean.cv.errors4),
  mean(mean.cv.errors5),
  mean(mean.cv.errors6)
#  mean(mean.cv.errors7),
#  mean(mean.cv.errors8)
)
names(all.cv.error) = c("Full Regression Model, Poisson", "Best Subsets Logistic Regression Model", '
all.cv.error = t(all.cv.error)
names(all.cv.error) = c("Model", "Mean CV Error")
```