Final Project Stat 716

Dane Isenberg, Alexis Ondongo, and Johanky Reyes

11/17/2018

## Introduction:

Marketing campaigns - that is campaigns to sell products via direct contact with customers, naturally want to optimize their targeting strategy. Of great interest are the characteristics of a customer that would indicate most clearly whether or not a customer would purchase the product. In the context of statistical language, we call these characteristics of the customers - the *predictor variables* - and whether or not the customer purchases the product - the binary *response variable*.

In this particular statistical analysis, we look at real data set collected from a phone marketing campaign conducted by a Portuguese retail bank to get potential clients to subscribe to a term deposit with them. The original data set, which was cleaned and parsed down for the analysis below, was gathered over multiple years from 2008 to 2013 and included 52,944 phone contacts[[1]](#footnote-1). The response variable, if the client subscribes to a term deposit (yes or no) was originally considered against 150 predictors but was reduced to 16 predictors by a feature selection process done independent of this paper. The 16 predictors that we consider in this analysis consider characteristics including standard demographic information (age,marital status etc.), socioeconomic indicators (education level, housing loan etc.), and type of contact (time of day, duration of call etc.).

As suggested above, the overall goal of our statistical analysis, is to predict as accurately as possible, based on 16 predictors, whether or not potential customers would commit to a term deposit. We do this by exploring several classification methods (that is methods interested in predicting whether or not a client takes a deposit), from logistic regression to random forests, on a pre-selected training data set of 31,647 observations and use cross-validation to assess their predictive success. Naturally, this bank and others, would like to optimize their marketing. If they have a vehicle that helps them predict clients with high accuracy based on several characteristics, they can target clients more effectively and with potentially few resources. In fact, this data set was collected during a period of economic recession, when banks were struggling and marketing efficiency was at a premium. To that end, this paper also examines some of the most significant predictors related to the response as they may provide an opportunity for banks to make greater sense of what characterizes their best potential client pool.

In today’s world, there has been a lot of focus on predatory modeling - to the point where even a mainstream book, **Weapons of Math Destruction** ,came out concerning the subject. While this may have not been an explicit goal of the original data collection, this analysis also gives banks and regulatory agencies a view on how banks could both gain advantage over other banks in their marketing practices or worse how banks, especially those that are now exclusively online, could exploit characteristics of potential customers to optimize their marketing. Think of for-profit-colleges and predatory pay-day loans.

We start the analysis by first describing the data set, then we discuss, fit, and evaluate classification models on the data, which includes a consideration of the relevant predictors, and finally we make some conclusions based on these results.

Note: We only provide code and its commentary if newly used.

## A Look at the Data:

Here is a snapshot of the data set. There are 31,647 rows - corresponding to each client called, and there are 17 columns, corresponding to the variables measured on each client (16 predictors, and 1 response variable).

The output provides a summary of the variables contained within the data frame, that is it describes the columns as representing a “factor” or an “integer” variable for example.

We see that martial is a categorical variable with 3 levels: divorced, married, and single and age is quantitative taking integral values.

str(Bank)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 31647 obs. of 17 variables:  
## $ age : int 39 52 40 48 46 33 49 34 36 28 ...  
## $ job : Factor w/ 12 levels "admin.","blue-collar",..: 8 5 7 5 10 2 2 2 10 2 ...  
## $ marital : Factor w/ 3 levels "divorced","married",..: 2 2 3 2 2 2 2 2 3 2 ...  
## $ education: Factor w/ 4 levels "primary","secondary",..: 2 3 4 1 3 2 2 2 2 2 ...  
## $ default : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ balance : int 0 1646 109 427 2207 2254 171 103 5436 373 ...  
## $ housing : Factor w/ 2 levels "no","yes": 2 1 2 1 2 2 1 2 2 2 ...  
## $ loan : Factor w/ 2 levels "no","yes": 1 2 1 1 1 1 1 2 1 1 ...  
## $ contact : Factor w/ 3 levels "cellular","telephone",..: 3 1 1 1 3 1 1 3 3 1 ...  
## $ day : int 20 9 4 17 16 17 4 8 20 14 ...  
## $ month : Factor w/ 12 levels "apr","aug","dec",..: 9 4 4 10 9 1 4 9 9 9 ...  
## $ duration : int 793 92 258 62 243 130 19 956 443 10 ...  
## $ campaign : int 2 3 1 1 3 1 3 2 1 9 ...  
## $ pdays : int -1 -1 -1 -1 -1 148 175 -1 -1 344 ...  
## $ previous : int 0 0 0 0 0 1 2 0 0 1 ...  
## $ poutcome : Factor w/ 4 levels "failure","other",..: 4 4 4 4 4 1 2 4 4 1 ...  
## $ y : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...

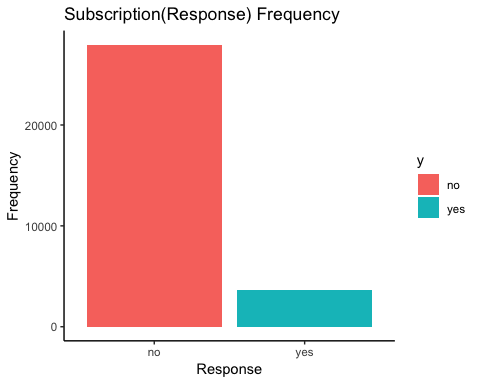
The variables for the Bank data set are listed out below. Included is a description of the variables and a basic graphical representation of their distributions.

### Response:

The response variable y is whether or not the client subscribed to a term deposit with the bank.

library(ggplot2)  
theme\_set(theme\_classic())  
freqy<-table(Bank$y)  
#proportion of y of each category  
prop.table(freqy)

##   
## no yes   
## 0.8835277 0.1164723

#frequency bar chart for subscription to term deposit   
ggplot(Bank) + geom\_bar(mapping = aes(x = y, fill = y)) + labs(title="Subscription(Response) Frequency", x="Response",y="Frequency")

This shows you that a large percentage of people contacted do not subscribe to a term deposit with this bank with a success rate of only about 11.65%. Generally speaking with such a large proportion of the response being no, an extremely rough model of simply always predicting no for your response is actually not the worst classifier.

### Predictors:

For each of the following predictors, we provide a frequency plot. Depending on whether the variable is taken as quantitative or categorical, we use a 5-number summary or proportion of each level of the factor respectively in the data. In order to visualize the variables relationship with the predictor, we provide either stacked bar charts accompanied by a cross-tabulation (categorical) or a jitter plot with brief commentary (quantitative).

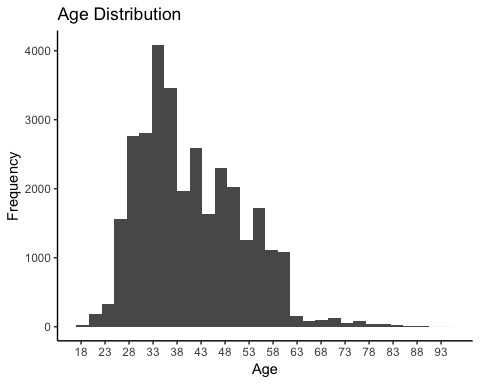
#### 1. Age

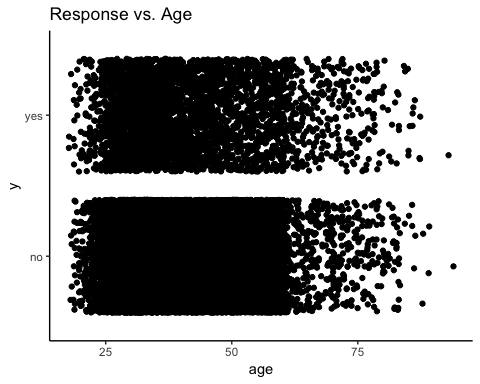
age = age (in years) of the client

#number summary of age   
summary(Bank$age)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 18.00 33.00 39.00 41.01 49.00 94.00

#histogram of age data   
ggplot(Bank, aes(x=age)) + geom\_histogram()+xlab("Age")+ylab("Frequency")+ggtitle("Age Distribution")+scale\_x\_continuous(breaks = round(seq(min(Bank$age), max(Bank$age), by = 5),1))



Jitter Plot Response vs. Age:

#jitter plot of y vs. age   
ggplot(Bank,aes(x=age,y=y))+geom\_jitter()+ggtitle("Response vs. Age")

The plot suggests that the youngest and oldest are committing to term deposits more readily as seen by the greater concentration of points in the yes response relative to the no response in these areas.

Indeed for those over 65 the proportion of yes is 40% vs. 11.65% in whole population.

#proportion yes given over 65  
sum(Bank$age>=65 & Bank$y=="yes")/sum(Bank$age>=65)

## [1] 0.4003527

For those under 20, proportion of yes is 31.4%.

#proportion yes given under 20  
sum(Bank$age<=20& Bank$y=="yes")/sum(Bank$age<=20)

## [1] 0.3142857

#### 2. Job

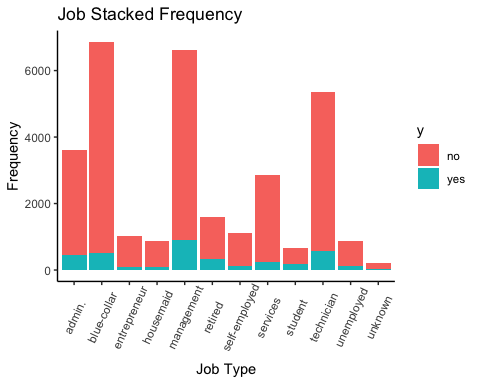
job= type of job of client

#proportion of data points for each level of factor

freqjob<-table(Bank$education)  
prop.table(freqjob)  
## admin. blue-collar entrepreneur housemaid management   
## 0.114355231 0.216766202 0.032609726 0.027585553 0.209087749   
## retired self-employed services student technician   
## 0.050304926 0.035295605 0.090245521 0.020507473 0.169305147   
## unemployed unknown   
## 0.027585553 0.006351313

#stacked bar chart

ggplot(Bank) + geom\_bar(mapping = aes(x = job, fill = y)) + labs(title="Job Stacked Frequency", x="Job Type",y="Frequency")

From this chart and the cross-tabulation below, we see that retired people and students are most likely to commit to a term deposit. This actually makes a lot of sense given what we observed about age in that most retired folks are above 65 and most students are younger (below 22) in age.

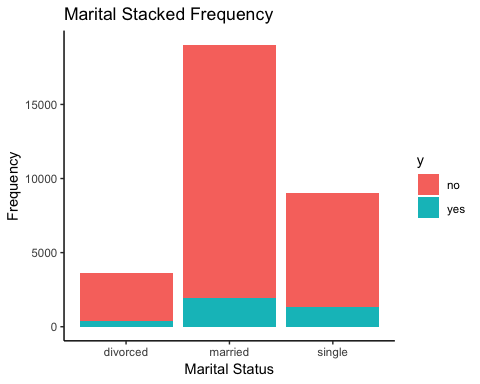
Cross-Tabulation Response vs. Job:

##   
## admin. blue-collar entrepreneur housemaid management retired  
## no 0.87510362 0.92434402 0.91472868 0.91179840 0.86277769 0.78454774  
## yes 0.12489638 0.07565598 0.08527132 0.08820160 0.13722231 0.21545226  
##   
## self-employed services student technician unemployed unknown  
## no 0.88809311 0.91246499 0.70416025 0.89305711 0.84421535 0.88557214  
## yes 0.11190689 0.08753501 0.29583975 0.10694289 0.15578465 0.11442786

#### 3. Martial

marital= martial status of client

divorced married single   
0.1142604 0.6013840 0.2843555

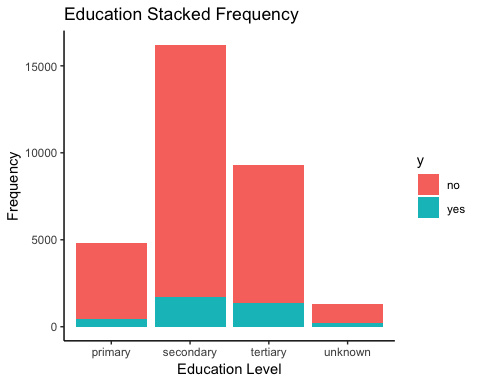
 We see that the type of marital status doesn’t figure so much into whether or not you commit to a term deposit. Perhaps, there is a slight uptick when someone is single.

Cross-Tabulation Response vs. Marital:   
## divorced married single  
## no 0.8896571 0.8975935 0.8513168  
## yes 0.1103429 0.1024065 0.1486832

#### 

#### 4. Education

education = education level of client  
## primary secondary tertiary unknown   
## 0.15129396 0.51240244 0.29411951 0.04218409

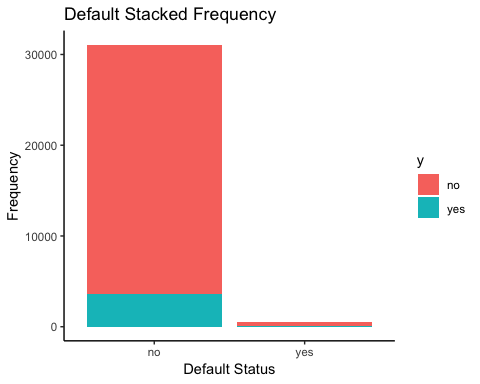


Generally speaking, we see that as education level increases the likelihood that you commit to a deposit also increases. That said, we can’t say how the unknown, which has a relative high percent of positive response (more than in the general data set), affects that assertion even if it constitutes a small percentage of the data.

Cross-Tabulation Response vs. Education:   
# primary secondary tertiary unknown  
## no 0.9122807 0.8947953 0.8524925 0.8599251  
## yes 0.0877193 0.1052047 0.1475075 0.1400749

#### 5. Default

default = whether or not client is in default on a loan  
## no yes   
## 0.98205201 0.01794799

 Note, columns represent default status. This suggests that it is significantly\* more likely for a client who is not in default to sign up for a term deposit with the bank.

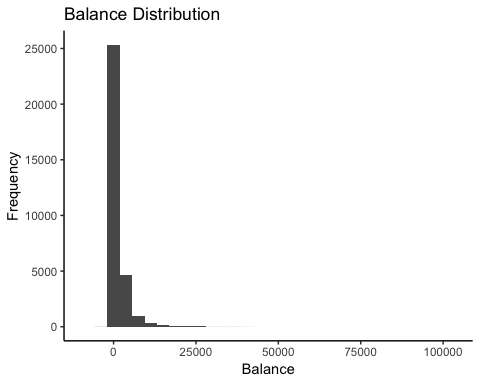
As n is large for each category, we might reasonable assume this\* but a test of significance (chi-squared) would be in order to make this a definitive claim.

Cross-Tabulation Response vs. Default:   
## no yes  
## no 0.8827182 0.9278169  
## yes 0.1172818 0.0721831

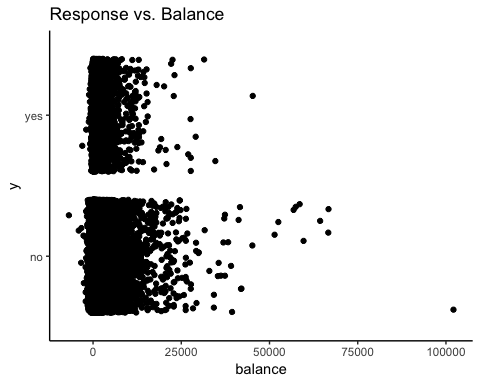
#### 6. Balance

balance = amount of money (dollars) client has in the bank

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -6847 72 450 1351 1420 102127



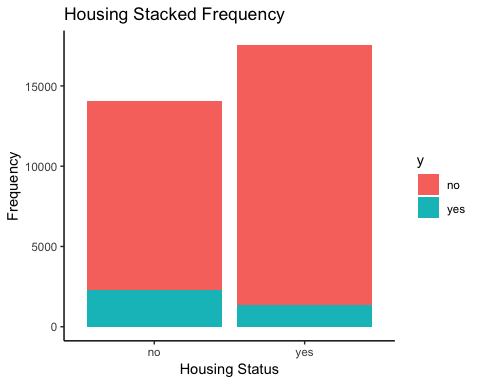
Jitter Plot Response vs. Balance:



This generally shows that that if you have somewhere between and you are most likely to sign up for a deposit with the bank. But, as more than 75% of folks have below , the largest quantity of yes responses come from those below that threshold.

#### 7. Housing

housing = whether or not client has a housing loan  
## no yes   
## 0.4448763 0.5551237



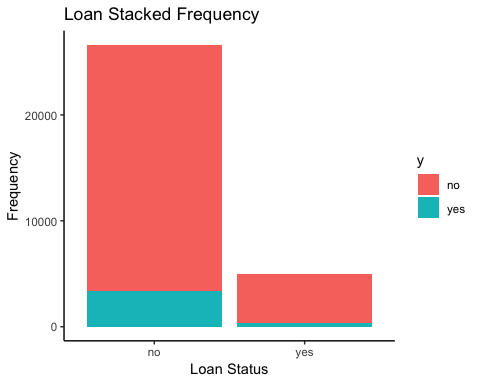
Note, columns of the cross-table represent housing loan status. The data suggest that it is significantly more likely for a client who does not have a housing loan to sign up for a term deposit with the bank.

Cross-Tabulation Response vs. Housing:  
## no yes  
## no 0.83557071 0.92196038  
## yes 0.16442929 0.07803962

#### 8. Loan

loan = whether or not client has a personal loan

##   
## no yes   
## 0.8414384 0.1585616

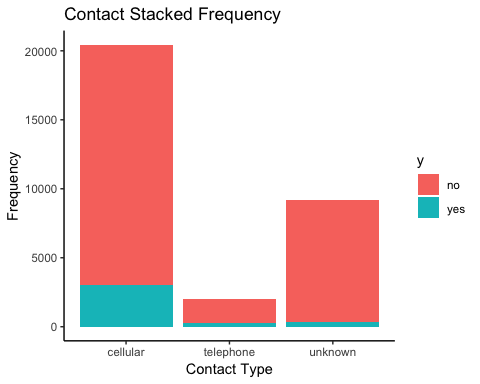


Note, columns of cross-table represent personal loan status. This suggests that it is significantly more likely for a client who does not have a loan to sign up for a term deposit with the bank.

Cross-Tabulation Response vs. Personal Loan:  
## no yes  
## no 0.87355890 0.93642886  
## yes 0.12644110 0.06357114

#### 9. Contact

contact = by what form of communication client was reached  
## cellular telephone unknown   
## 0.64600120 0.06430309 0.28969571



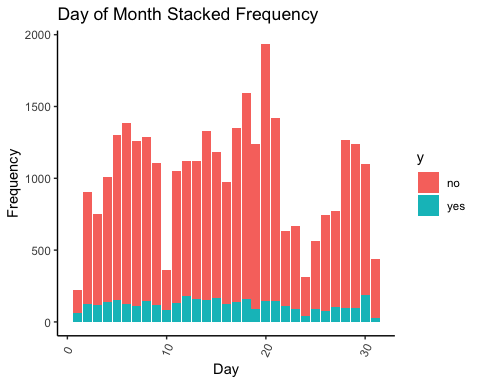
Cross-Tabulation Response vs. Contact:  
## cellular telephone unknown  
## no 0.85051849 0.86535627 0.96116928  
## yes 0.14948151 0.13464373 0.03883072

We see that for cellular vs. telephone the proportion of those who commit to a tem deposit is nearly the same. However, for the unknown category, there is a significantly lower proportion of those with a yes response. This really makes this variable a messy predictor because we don’t know what proportion of the unknown constitutes cell vs. landline. If we view unknown as a separate level of this factor variable apart from cell and landline, it’s presence becomes a strong indicator of a no response when in fact it is actually comprised of some combination of cell and landline contacts.

#### 10. Day

day = day of the month corresponding to last contact.

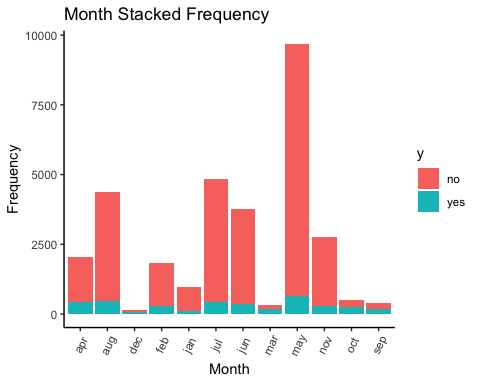
We provide just the chart due to the volume of categories.



It’s hard to discern any real pattern here. Of note is the fact that there are considerable fluctuations from day-to-day in the success rate. Indeed, there is a very high success rate on the first day of the month and a very low success rate on the last day of the month.

#### 11. Month

month = month corresponding to last contact  
## apr aug dec feb jan jul   
## 0.064682276 0.138370146 0.004802983 0.057951781 0.031092995 0.152526306   
## jun mar may nov oct sep   
## 0.119505798 0.010522324 0.305810977 0.086738079 0.015988877 0.012007457



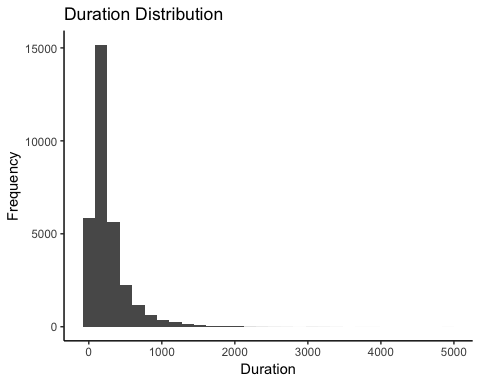
Cross-Tabulation Response vs. Month:  
## apr aug dec feb jan jul  
## no 0.79677577 0.89335465 0.50657895 0.83587786 0.89735772 0.90884607  
## yes 0.20322423 0.10664535 0.49342105 0.16412214 0.10264228 0.09115393  
##   
## jun mar may nov oct sep  
## no 0.90163934 0.49849850 0.93263071 0.89471767 0.53162055 0.55526316  
## yes 0.09836066 0.50150150 0.06736929 0.10528233 0.46837945 0.44473684

Certain months really appear to be an extremely strong predictor of success or failure. Indeed, if you’re contacted in September, October or March, the success rate is around 50%. But, if you’re contacted in June or July, your success rate is below 10%.

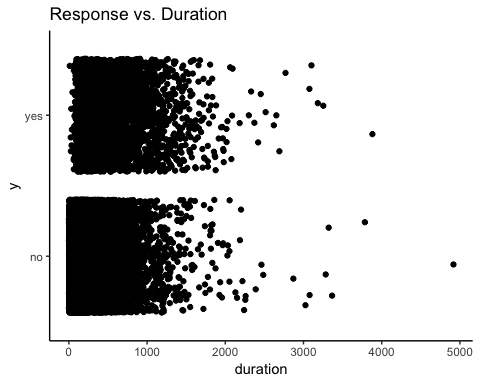
#### 12. Duration

duration = the amount of time spent (in seconds) in last contact with client

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 103.0 179.0 258.3 317.0 4918.0



Jitter Plot Response vs. Duration:

 This appears to be the strongest predictor yet in our sample. You can clearly see the concentration of points growing in the yes response as the duration of the contact increases. We can see this success rate increase considerable if you are below the first quartile vs. within the first quartile and median vs. above the third quartile of duration:

#proportion yes given under first quartile duration  
sum(Bank$duration<103.0 & Bank$y=="yes")/sum(Bank$duration<103.0)

## [1] 0.01039027

#proportion yes given over median duration  
sum(Bank$duration>=103.0 & Bank$duration<317.0 & Bank$y=="yes")/sum(Bank$duration>=103.0 & Bank$duration<317.0)

## [1] 0.08042268

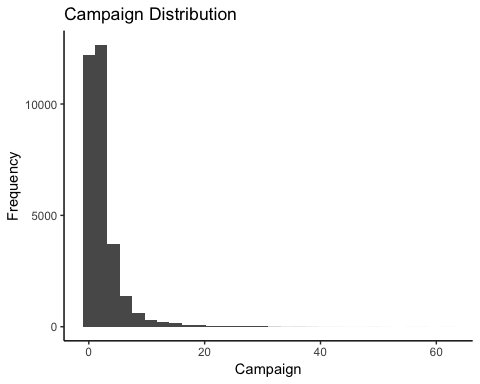
#proportion yes over third quarterile duration   
sum(Bank$duration>=317 & Bank$y=="yes")/sum(Bank$duration>=317)

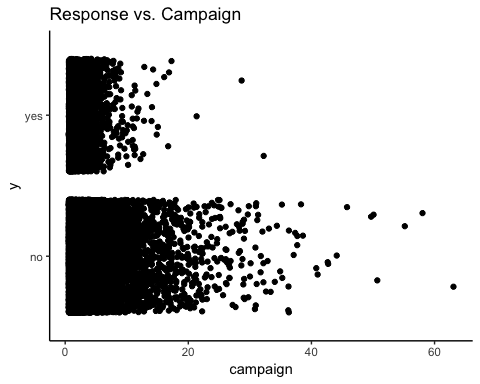
## [1] 0.2934222

#### 13. Campaign

campaign = number of contacts within campaign to an individual client (in an effort to get them to deposit in the bank)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 1.000 2.000 2.776 3.000 63.000



Jitter Plot Response vs. Campaign:

This plot shows most clearly that beyond 15 or so contacts in the campaign the success rate is quite small, that is excessive calling is not successful as can be seen by the first vs. second output in the commands below.

#proportion yes given over 15 contacts  
sum(Bank$campaign>15 & Bank$y=="yes")/sum(Bank$campaign>15)

## [1] 0.01861702

#proportion yes given under 15 contacts   
sum(Bank$campaign<=15 & Bank$y=="yes")/sum(Bank$campaign<=15)

## [1] 0.1176489

### 14. Pdays

pdays = number of days that pass after the previous calling attempt was made, -1 means no previous calling attempt was made.

Before we make a histogram of the days that pass after the previous calling attempt. We briefly look at success rate no attempt vs. attempt.

#no attempt success rate   
sum(Bank$pdays==-1 & Bank$y=="yes")/sum(Bank$pdays==-1)

## [1] 0.09057187

#attempt success rate   
sum(Bank$pdays>-1 & Bank$y=="yes")/sum(Bank$pdays>-1)

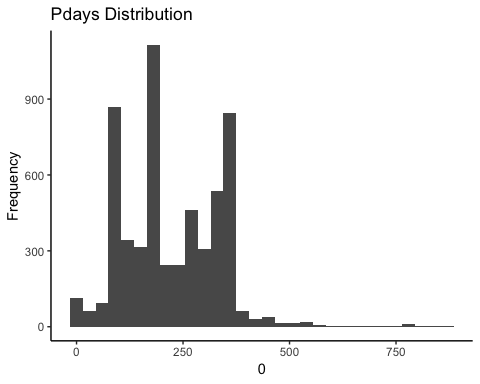
## [1] 0.2327033

As you can see, if there were no previous calling attempts made then the success rate is substantially lower than if there was.

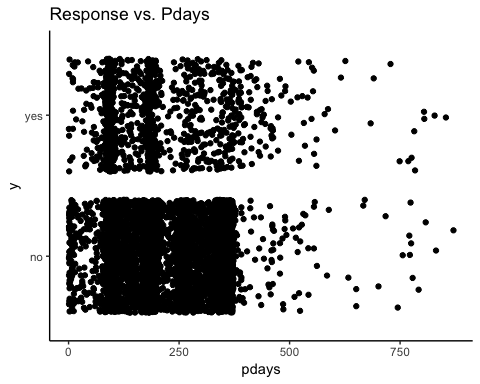
Looking at the histogram of success rate for different values of pdays (not =-1), we get:

#extracts all actual attempts  
Banka<- Bank[Bank$pdays>-1,]

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.0 131.0 195.0 224.3 327.0 871.0



Jitter Plot Response vs. Pdays (not =-1):

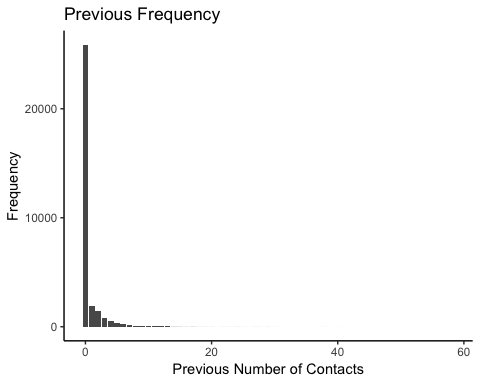


This plot doesn’t demonstrate a clear relationship between the number of days since the previous contact and the success rate. There appears to be some periods when the success rate is higher than others for example it’s quite lower at around 250-350 days and quite higher from 400-500 days.

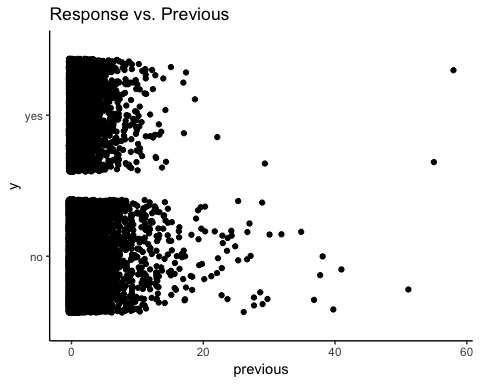
#### 15. Previous

previous = number of contacts made before the current calling campaign to a given client

Summary information for previous is not helpful as vast majority is 0.



Jitter Plot Response vs. Previous:



In this plot, we look at the response vs. previous number of contacts before the current campaign (previous) and we see a similar result as we saw with the response vs. previous number of contacts within campaign (campaign), but the threshold is a bit later. Indeed, as the number of contacts reaches beyond 22, the success rate declines. However, interestingly, that while beyond 40 previous contacts is certainly rare, the success rate is relatively high for that quantity of calls. Yes, they were clearly sick of being harassed.

#proportion yes given over 22 contacts previous  
sum(Bank$previous>22 & Bank$y=="yes")/sum(Bank$previous>22)

## [1] 0.08571429

#proportion yes given under 22 contacts previous  
sum(Bank$previous<=22 & Bank$y=="yes")/sum(Bank$previous<=22)

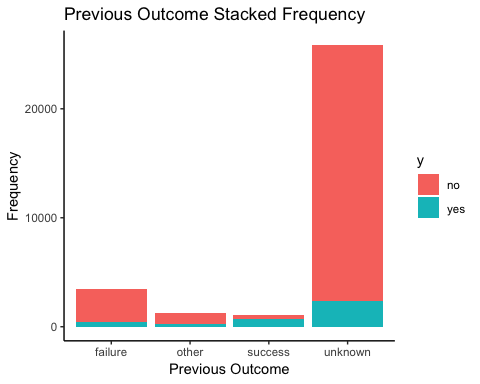
## [1] 0.1165064

#proportion yes given over 40 contacts previous  
sum(Bank$previous>40 & Bank$y=="yes")/sum(Bank$previous>40)

## [1] 0.5

#### 16. Poutcome

poutcome=outcome of previous marketing campaign  
## failure other success unknown   
## 0.10879388 0.04047777 0.03283092 0.81789743



Most significantly, this cross-tabulation demonstrates that if poutcome=success for a client, it is an extremely strong indicator that this new calling campaign will also be a success for that client.

Cross-Tabulation Response vs. Poutcome:   
## failure other success unknown  
## no 0.86988092 0.82825917 0.35322425 0.90936486  
## yes 0.13011908 0.17174083 0.64677575 0.09063514

We now explore possible models!

## Methods and Results:

Below are the packages necessary to produce our models and evaluate their fits. Note, we truncate some of the outputs of our models that are not necessary for understanding.

library(caret)

library(randomForest)

library(glmnet)

library(gbm)

library(rpart)

### Generalized Linear Models: Parametric Approach

#### Full Logistic Regression

We start with logistic regression, which is a generalized linear model (glm). We begin here because it is the simplest model to interpret and it uses all the predictors we already believe to be related to the response variable. Moreover, while not the explicit goal of this report, we can also use it for inference, that is determining whether the roles of subsets of predictors are significant relative to the response. So, if the glm model predicts well, we have an added bonus. We discuss prediction vs. utility in the concluding remarks.

We assess the fit of this model and others using 5-fold cross validation (CV). 5-fold cross validation, partitions the data set into 5 (disjoint) subsets, trains the model on 4 of those 5 and tests on the remaining one. It will repeat this for each distinct subset. While ideally we’d like to do 10-fold CV, the computing time for later models was prohibitive. We use accuracy (average accuracy over the 5 test subsets) throughout as our metric for evaluating prediction strength as it is extremely straightforward and relates directly to our goal.

With glm and it’s variants below, we note that we model to find the probability of success of client commitment, where above .5 is classified as a success (yes) and below .5 a failure (no).

#establishes a 5-fold cross-validation method for this model and all future   
train.control<-trainControl(method="cv",number=5)  
set.seed(5)  
#computes 5-fold cv on glm fit  
cv.glm.fit<-train(y~.,data=Bank,method='glm',family='binomial', metric='Accuracy',trControl=train.control)  
#prints accuracy of fit  
cv.glm.fit$results$Accuracy

## [1] 0.9013494

We get an accuracy of .9013494 using the full glm model.

Of course, we know that the glm model using the full set of predictors as opposed to a proper subset of them suffers from the highest variance as it requires the most estimated parameters. As such, we make attempts at reducing variance (avoiding overfitting) via variable subset selection and penalized regression.

#### Dimension Reduction: Backward-selection

We reduce the number of parameters on the full model via stepwise AIC. In this form of backward selection, you start with the full model and reduce the model sequentially to minimize AIC, which is a statistic of goodness of fit that penalizes for the number of predictors you use.

set.seed(5)  
#cv error on step AIC model, the variables included in the model are listed in command  
cv.glmfitopt<-train(y~job + marital + education + balance +   
 housing + loan + contact + day + month + duration + campaign +   
 previous + poutcome,data=Bank,method='glm',family='binomial',metric='Accuracy',trControl=train.control)  
cv.glmfitopt$results$Accuracy

## [1] 0.9013494

This step AIC model is simpler but the prediction accuracy is identical to that of the original full model.

#### Dimension Reduction: Elastic Net Shrinkage

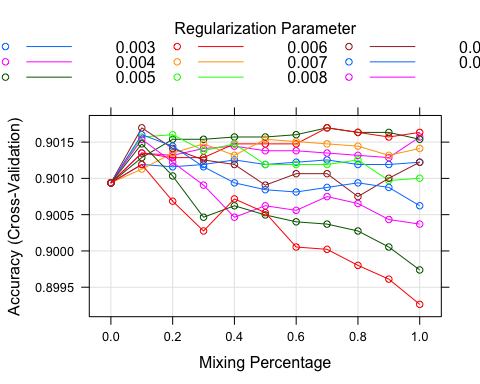
We do penalized regression (or regularization) via elastic net, picking the best model for tuning parameters and , where when = 0 we get RIDGE and we get LASS0 and is the weight of your penalty for estimating too many parameters - thus allowing for both variable selection (LASSO) and handling of correlation among predictors (RIDGE). Often times, stepwise AIC handles correlated predictors poorly as well.

We provide a plot to show what’s going on as we tune over these parameters.

set.seed(5)  
#sets tuning parameters for lambda and alpha as dicussed above  
lambda\_grid <- seq(0, .01, 0.001)  
alpha\_grid <- seq(0, 1, 0.1)  
full\_grid<-expand.grid(.alpha = alpha\_grid, .lambda = lambda\_grid)  
#5-fold cv for elastic net  
cv.penr<-train(y ~., Bank, method = "glmnet",tuneGrid = full\_grid, trControl = train.control)  
#optimal tuning parameters  
cv.penr

## glmnet   
##   
## 31647 samples  
## 16 predictor  
## 2 classes: 'no', 'yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 25317, 25318, 25318, 25317, 25318   
## Resampling results across tuning parameters:  
  
## 0.6 0.010 0.9000539 0.3501021  
## 0.7 0.000 0.9012546 0.3957851  
## 0.7 0.001 0.9013493 0.3930289  
## 0.7 0.002 0.9016969 0.3906781  
## 0.7 0.003 0.9016970 0.3867916  
  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were alpha = 0.7 and lambda = 0.003.

#plots cv accuracy as parameters adjust   
plot(cv.penr)



Note, we’ve cropped the output of the model for convenience. The final accuracy is 0.9016970 for these optimal tuning parameters (alpha = 0.7 and lambda = 0.003), which is a slight improvement over our previous two glm models. The size of alpha suggests that this fit is overall closer to LASSO.

#### Reduction of Variables Based on Data Exploration:

Circling back to our data exploration, we saw the whether or not a previous campaign was a success and the duration of the contact were directly related to response. Fitting a model to those two predictors, we get:

set.seed(5)  
#cv for train vs duration and poutcome  
train(y~duration+poutcome,data=Bank,method='glm',family='binomial',metric='Accuracy',trControl=train.control)

## Generalized Linear Model   
##   
## 31647 samples  
## 2 predictor  
## 2 classes: 'no', 'yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 25317, 25318, 25318, 25317, 25318   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.9000539 0.3704389

We get about .9 accuracy with this very simple model and it’s quite easy to interpret.

We noted earlier that success or not for poutcome and attempt or no attempt for pdays were both significant. We converted these to binary predictors, and got:

set.seed(5)  
#makes pdays and pout binary  
pdays1<-ifelse(Bank$pdays==-1,0,1)  
pout1<-ifelse(Bank$poutcome=="success",1,0)  
#attaches new binary variables to original data frame  
Bankc<-mutate(Bank,pdays1,pout1)  
#runs regression and provides 5 fold cv  
train(y~.-pdays-poutcome,data=Bankc,method='glm',family='binomial', metric='Accuracy',trControl=train.control)

## Generalized Linear Model   
##   
## 31647 samples  
## 18 predictor  
## 2 classes: 'no', 'yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 25317, 25318, 25318, 25317, 25318   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.9016022 0.3995609

We get a .9016022 accuracy!

### Non-Parametic Methods:

As the underlying data may not conform to a generalized linear model, we turn to non-parametric models, that is models that don’t assume that the data can be fit according to a family of functions.

#### K-Nearest Neighbors(KNN)

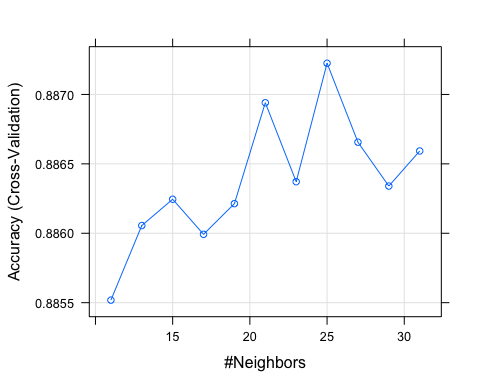
We start with KNN as it’s a method that is generally effective and is quite intuitive. Indeed, we identify the k-closest points (via euclidean distance) in our training data to the data point we wish to predict on. We predict yes if there is a higher percentage of these k-closest points that are yes and otherwise we predict no.

We tried KNN for different values of k, in our initial test run, we saw improvements up until k=21 so we looked for values from 11-31. The reason we don’t run a wider range is due to extreme computational inefficiency.

set.seed(5)  
#cv for KNN over odds tuning parameters k=11-31  
cv.knn<-train(y~.,data=Bank,method="knn",tuneGrid= expand.grid(k = c(11,13,15,17,19,21,23,25,27,29,31)),metric= "Accuracy",trControl=train.control)  
cv.knn

## k-Nearest Neighbors   
##   
## 31647 samples  
## 16 predictor  
## 2 classes: 'no', 'yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 25317, 25318, 25318, 25317, 25318   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 11 0.8855184 0.2671492  
## 13 0.8860555 0.2623334  
## 15 0.8862451 0.2588653  
## 17 0.8859923 0.2491578  
## 19 0.8862135 0.2481226  
## 21 0.8869403 0.2460678  
## 23 0.8863715 0.2336193  
## 25 0.8872246 0.2369991  
## 27 0.8866559 0.2247874  
## 29 0.8863399 0.2155249  
## 31 0.8865927 0.2151590  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 25.

#plot of KNN accuracy as K grows   
plot(cv.knn)

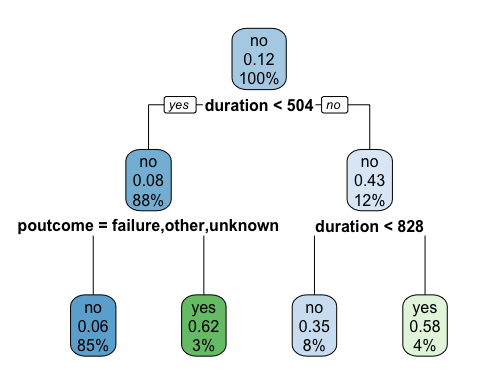


The KNN with K=25, gives us the best accuracy of .8872246, which does not perform better than our generalized linear models above.

#### Random Forests:

We move onto a more sophisticated model called a random forest, which relies on building classification *trees* from our data. Trees are loosely a set of paths of conditions that result in a yes or no response based on highest probabilities in the classifying step (terminal node of the path). For a forest, we then repeatedly resample from the data to build any number of trees and make a choice based on aggregating the classifications. We’ve built one small tree to demonstrate what a tree on this data looks like.

set.seed(5)  
library(rpart.plot)  
exampletree<-rpart(y~.,data = Bank, maxdepth=2)  
rpart.plot(exampletree)



For example, if you take the path duration less than 504 and poutcome is success, the model would predict yes as seen by the terminal node (with probability 62% -proportion of success+duration<504 that are yes out of total success+duration<504). Note that duration and poutcome are used in this very basic tree!

Indeed, we simulated this with the 28th data point which has the above conditions:

predict(exampletree, newdata=Bank[28,])

## no yes  
## 1 0.3802497 0.6197503

predict(exampletree, newdata=Bank[28,],type="class")

## 1   
## yes   
## Levels: no yes

Now, we could make more complex trees with a different node depth and use that alone to predict response. Nodes are where decisions/splits are made, the node depth above is 2, represented by circles in the diagram. But, too complex of trees risk overfitting the data and would require pruning, resampling, or boosting methods. We’ve opted first for resampling (random forest), as described above, and later we do boosting.

There are two tuning parameters with random forest that we consider. The first is the number of tree that are built from resamples ntree. The second is the number of randomly selected predictors out of the entire set of predictors to consider that you’re making at each node split of the tree, denoted mtry. Generally, for ntree large, the predictions level out and accuracy (or error) stabilizes. For smaller number of variables mtry to consider at splits, we get less correlated trees (but with slightly higher bias) and a reduction in variance in our random forest.

We will use OOB (out-of-bag error) to evaluate our fits, as the data set is so large that the program crashes when trying to cross-validate over these tuning parameters.

We set ntree=2000 (which is large enough so it doesn’t really need tuning) and play only with tuning parameter mtry, the default for the randomForest package is 4. We thus try a range of 3-5.

set.seed(5)  
#random forest with m=5 predictors  
bag.fit<-randomForest(y~.,data=Bank,ntree=2000,mtry=5)  
bag.fit

##   
## Call:  
## randomForest(formula = y ~ ., data = Bank, ntree = 2000, mtry = 5)   
## Type of random forest: classification  
## Number of trees: 2000  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 9.19%  
## Confusion matrix:  
## no yes class.error  
## no 26925 1036 0.03705161  
## yes 1873 1813 0.50813890

err5<-bag.fit$err.rate  
  
#default random forest with m=4 predictors (default)  
rf.fitd<-randomForest(y~.,data=Bank,ntree=2000)  
rf.fitd

##   
## Call:  
## randomForest(formula = y ~ ., data = Bank, ntree = 2000)   
## Type of random forest: classification  
## Number of trees: 2000  
## No. of variables tried at each split: 4  
##   
## OOB estimate of error rate: 9.16%  
## Confusion matrix:  
## no yes class.error  
## no 26979 982 0.03512035  
## yes 1916 1770 0.51980467

err4<-rf.fitd$err.rate  
  
#random forest with m=4 predictors   
rf.fit3<-randomForest(y~.,data=Bank,ntree=2000,mtry=3)  
rf.fit3

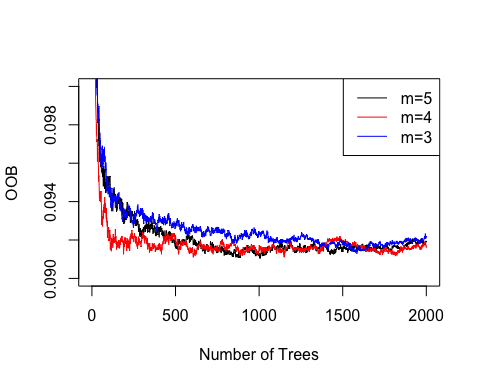
##   
## Call:  
## randomForest(formula = y ~ ., data = Bank, ntree = 2000, mtry = 3)   
## Type of random forest: classification  
## Number of trees: 2000  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 9.21%  
## Confusion matrix:  
## no yes class.error  
## no 27129 832 0.02975573  
## yes 2084 1602 0.56538253

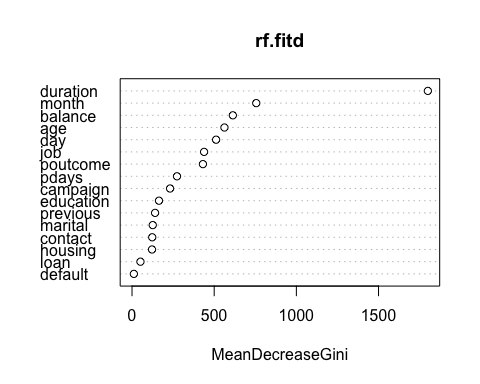
err3<-rf.fit3$err.rate

The optimal model, with the default 4 variable per split has an OOB error rate of .0916, that is an accuracy of (1-.0916)=.9084, which is the highest so far.

We provide a plot to show that the test accuracy of the 4 predictor split model beats out the other two for ntree large.

#plots err5 vs. number of trees as line graph  
plot(1:2000,err5[,1],type="l",ylim=c(.09,.1),xlab="Number of Trees",ylab="OOB")  
#plots err4 vs. number of trees as line graph  
lines(1:2000,err4[,1],col="red",type="l")  
#plots err3 vs. number of trees as line graph  
lines(1:2000,err3[,1], col="blue",type="l")  
#provides legend for graph  
legend("topright",c("m=5","m=4","m=3"),col=c("black","red","blue"),lty=1)



We plot a variable importance graph, where predictor importance is measured by the total amount that the Gini Index has decreased over splits of each predictor averaged over all trees. The Gini Index is larger when the proportions of yes’s and no’s for the response are near 50% in the terminal nodes - which is bad for prediction. Note, the product of two values between 0 and 1 is highest when both values are .5. We comment on this plot in the conclusion.

#variable importance graph  
varImpPlot(rf.fitd)

#### Boosting Trees:

Random forests are based on the premise that variance is reduced as you resample and average predictions over multiple relatively complex trees. However, these relatively complex trees themselves may have either high variance vis-a-vis less complex trees or high bias vis-a-vis more complex trees - thus may compound these issues as you iterate over many trees. We approach the bias-variance trade-off a bit differently when boosting - instead we fit a small tree with high bias and low variance to begin with and then build on it sequentially, both reducing bias and variance, by adding small trees to that initial model.

The tuning parameters for the boosting approach are the following:

-n.trees=number of trees used sequentially, which we don’t want too large as to overfit but too small as to not decrease variance enough. That said, we want these to be bigger if our shrinkage is small.

-interaction.depth=number of splits in each tree (tree complexity), which we want to keep relatively minimal

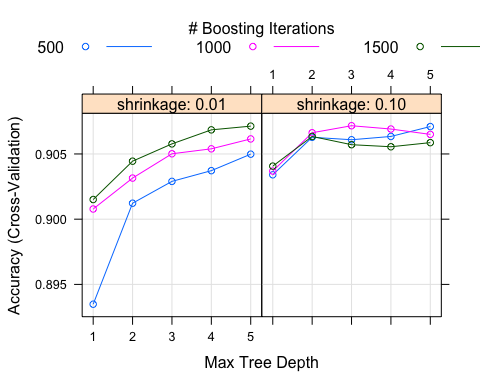
-shrinkage = learning rate, which we wish to keep small and force slower learning

set.seed(5)  
#sets tuning parameters, n.minobsinnode = minimun number of observations in terminal nodes set to default  
boost.grid<-expand.grid(n.trees=c(500,1000,1500),interaction.depth=(1:5),shrinkage=c(.01,.1),n.minobsinnode=10)  
#runs 5-fold CV for over all tuning parameters for boosting trees   
boost.tr<-train(y~.,data=Bank,method="gbm",trControl=train.control,metric="Accuracy",tuneGrid=boost.grid

boost.tr

## Stochastic Gradient Boosting   
##   
## 31647 samples  
## 16 predictor  
## 2 classes: 'no', 'yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 25317, 25318, 25318, 25317, 25318   
## Resampling results across tuning parameters:  
## 0.10 2 1500 0.9063419 0.4698372  
## 0.10 3 500 0.9060892 0.4665417  
## 0.10 3 1000 0.9071635 0.4813180  
## 0.10 3 1500 0.9057099 0.4773371  
## 0.10 4 500 0.9063420 0.4731140  
## 0.10 4 1000 0.9069107 0.4841560  
## 0.10 4 1500 0.9055519 0.4814884  
## 0.10 5 500 0.9071003 0.4801018  
## 0.10 5 1000 0.9064999 0.4837311  
## 0.10 5 1500 0.9058679 0.4874892  
##   
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were n.trees = 1000,  
## interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.

#plots boost accuracy as parameters adjust   
plot(boost.tr)



We observe a highest accuracy of .9072, where shrinkage=.1, interaction.depth=3, and n.trees=1000.

## Conclusion:

The model that gave us the highest prediction accuracy overall, .9084, was the random forest model with tuning parameters: 2000 resampled trees, ntree=2000, and 4 randomly selected variables to consider at each split in a tree, mtry=4. Therefore, for prediction purposes, we would recommend this model (where ntree can be made larger if possible). What’s nice about this model, moreover, is that we can assess the importance of each variable in building the trees and reducing error as seen by the Gini Index variable importance plot. Indeed, we saw that duration and month were among the most powerful predictors in our optimal random forest model. When we were initially exploring the data, we saw that duration and month appeared to be significant predictors right off the bat. Note that age is the most powerful demographic predictor according to the variable importance plot. We saw in the exploratory data analysis that particularly old or young clients were more susceptible to commit to a term deposit - this may encourage predatory activities in the future as we discussed in the introduction.

That said, if we were to pitch a model or explain a model to a company, we might prefer to use the stepwise reduced glm model or the model where the two variables poutcome and pdays were made binary. These models are easy to build and great for inference (that is to measure the effect of individual subsets of predictors on the response). Indeed, the reduced model has the added benefit of having fewer predictors. For example, when looking at what predictors are significant via their p-value in this reduced model, we see once again that duration, poutcome=succees, and certain months play a huge part. We could do a more robust inferential analysis if we wish. In fact, using just the duration and poutcome variables in our glm alone was enough to make accuracy 90%.

There were three major limitations to our analysis. First, the “unknown” level in certain predictor variables invalidates the accuracy of our models to some extent, in particular it might limit or unduly increase the effect of the other levels associated with that variable. For example, in the poutcome variable, success in the previous campaign was a huge indicator of a yes response, but 82% of respondents were unknown in this variable and the success rate for these folks was significantly lower. We could be over-measuring the effect of success for poutcome in our models if there is a decent proportion of unknown folks that had a yes response in a previous campaign. The second limitation was in optimizing tuning parameters for our models. Simply put, it takes too much computing power to cross-validate over the scope of tuning parameters necessary, particularly for the non-parametric models. For example, for boosting, we were not able to tune over enough trees for slower learning algorithms (shrinkage small) and it was even worse for the support vector machine for even just a linear subspace - we had to leave it out of our list of models because of gross inefficiency. Moreover, our knowledge of what range of tuning parameters to consider in what circumstances is not so refined, so further hands-on practice with statistical learning is definitely required. The last limitation is that we had to use OOB error rate (1-error = accuracy) instead of that obtained from 5-fold CV for our random forest model as our computer running R could not complete the later task in a reasonable time-frame. Indeed, while this model had the highest approximate prediction accuracy, the OOB measure makes the comparison of this model relative to the others a bit less robust.

All in all, a 90.84% accuracy is pretty great, and we can be sure that our final random forest model will do quite well with prediction. And if inference is of primary importance, we could always use our backward-selected glm and get a solid prediction accuracy as well.

1. S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014 [↑](#footnote-ref-1)