Case 3: Predictive Modelling - Targeting Telemarketing

Dr. Avery Haviv
University of Rochester
MKT436 and MKT436R

Fall 2020

This case is based around a real world dataset about telemarketing calls made by a Portuguese bank. You can find more information about this dataset here: https://archive.ics.uci.edu/ml/datasets/bank+marketing

The bank is interested in a predictive model because it will allow them to call the right customers at the right times. From an analytics perspective, the primary distinguishing feature in this case is that solving a predictive problem is directly useful to the firm.

Basic Descriptive Analysis

As before, we start by looking at the columns of the dataset, the basic correlations, and a regression with all the variables of interest.

```
bankData = read.csv('D:/Dropbox/Teaching Lectures/Bank Telemarketing Case/Bank Case.csv',stringsAsFacto
#The str function lets you see a nice summary of the data.
str(bankData)
```

```
## 'data.frame':    41188 obs. of  12 variables:
##  $ age        : int  56 57 37 40 56 45 59 41 24 25 ...
##  $ job        : Factor w/ 12 levels "admin.","blue-collar",..: 4 8 8 1 8 8 1 2 10 8 ...
##  $ marital    : Factor w/ 4 levels "divorced","married",..: 2 2 2 2 2 2 2 2 3 3 ...
##  $ education  : Factor w/ 8 levels "basic.4y","basic.6y",..: 1 4 4 2 4 3 6 8 6 4 ...
##  $ default    : Factor w/ 3 levels "no","unknown",..: 1 2 1 1 1 2 1 2 1 1 ...
##  $ housing    : Factor w/ 3 levels "no","unknown",..: 1 1 3 1 1 1 1 1 3 3 ...
##  $ loan       : Factor w/ 3 levels "no","unknown",..: 1 1 1 1 3 1 1 1 1 1 ...
##  $ contact    : Factor w/ 2 levels "cellular","telephone": 2 2 2 2 2 2 2 2 2 2 ...
##  $ month      : Factor w/ 10 levels "apr","aug","dec",..: 7 7 7 7 7 7 7 7 7 7 ...
##  $ day_of_week: Factor w/ 5 levels "fri","mon","thu",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ duration   : int  261 149 226 151 307 198 139 217 380 50 ...
##  $ y          : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

The dependent variable we are interested in is y, which indicates whether the customer signed up for the new deposit account. Let's transform that into a Boolean variable.

```
bankData$y = bankData$y == 'yes'
```

'Duration' refers to how long the call was. The variable should be deleted from the analysis, as the firm does not know how long the call will be before they make it!

```
bankData$duration = NULL
```

Since this is a predictive problem, let's define the training and validation sets so we can test models.

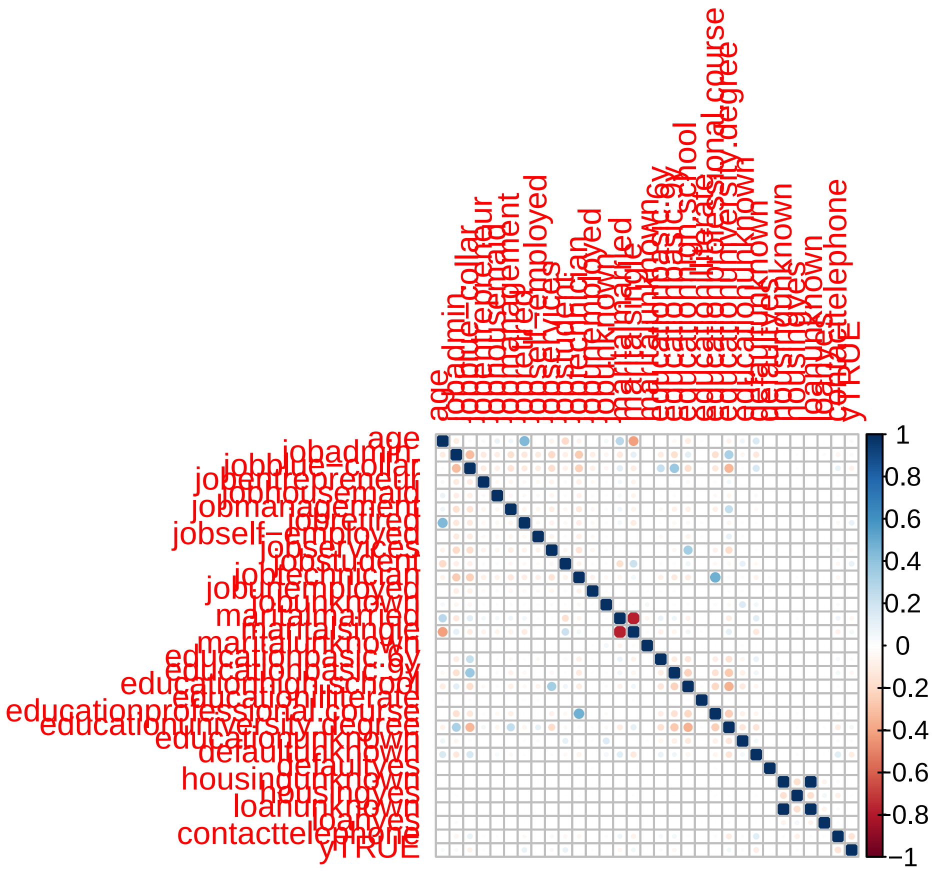
```
set.seed(1)
isTraining = runif(nrow(bankData))<.8
trainingData = subset(bankData,isTraining)
validationData = subset(bankData,!isTraining)
```

Let's look at a correlation plot. I am removing day_of_week and month to make the graph more readable.

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corrplot(cor(model.matrix(~.-day_of_week-month-1,data=bankData)))
```



Some obvious correlations are apparent in the correlation plot: Age is negatively correlated with marriage, but positively correlated with income. This gives some confidence in the integrity dataset.

```
summary(lm(y~.,data=bankData))
```

```
##
## Call:
## lm(formula = y ~ ., data = bankData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.64287 -0.12109 -0.08726 -0.02863  1.05723
##
## Coefficients: (1 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.674e-01  1.350e-02  12.393  < 2e-16 ***
## age                          8.316e-04  1.835e-04   4.532 5.86e-06 ***
## jobblue-collar              -2.293e-02  5.493e-03  -4.174 3.00e-05 ***
## jobentrepreneur             -2.282e-02  8.523e-03  -2.677 0.007423 **
## jobhousemaid                -6.812e-03  1.015e-02  -0.671 0.501998
## jobmanagement                -1.595e-02  6.417e-03  -2.486 0.012931 *
## jobretired                    7.291e-02  8.972e-03   8.127 4.53e-16 ***
## jobself-employed             -1.762e-02  8.560e-03  -2.058 0.039599 *
## jobservices                  -1.997e-02  5.974e-03  -3.343 0.000828 ***
## jobstudent                    1.293e-01  1.109e-02  11.663  < 2e-16 ***
## jobtechnician                -1.425e-02  5.290e-03  -2.694 0.007061 **
## jobunemployed                 1.592e-02  9.987e-03   1.594 0.110963
## jobunknown                   -2.688e-05  1.716e-02  -0.002 0.998750
## maritalmarried                9.484e-03  4.862e-03   1.951 0.051120 .
## maritalsingle                 2.574e-02  5.590e-03   4.604 4.15e-06 ***
## maritalunknown                2.932e-02  3.386e-02   0.866 0.386562
## educationbasic.6y             4.434e-03  7.923e-03   0.560 0.575743
## educationbasic.9y            -6.900e-03  6.266e-03  -1.101 0.270826
## educationhigh.school         -1.503e-03  6.483e-03  -0.232 0.816712
## educationilliterate           1.210e-01  7.079e-02   1.710 0.087313 .
## educationprofessional.course  2.240e-03  7.290e-03   0.307 0.758674
## educationuniversity.degree    1.327e-02  6.615e-03   2.006 0.044862 *
## educationunknown              2.065e-02  8.901e-03   2.320 0.020344 *
## defaultunknown               -4.188e-02  3.864e-03 -10.841  < 2e-16 ***
## defaultyes                   -1.171e-01  1.730e-01  -0.676 0.498738
## housingunknown               -4.191e-03  9.789e-03  -0.428 0.668548
## housingyes                   -1.084e-03  3.013e-03  -0.360 0.718949
## loanunknown                          NA         NA      NA       NA
## loanyes                      -4.002e-03  4.131e-03  -0.969 0.332589
## contacttelephone             -9.006e-02  3.904e-03 -23.069  < 2e-16 ***
## monthaug                     -1.036e-01  7.090e-03 -14.616  < 2e-16 ***
## monthdec                      2.578e-01  2.304e-02  11.189  < 2e-16 ***
## monthjul                     -9.801e-02  6.895e-03 -14.214  < 2e-16 ***
## monthjun                     -1.868e-02  7.776e-03  -2.403 0.016269 *
## monthmar                      2.806e-01  1.414e-02  19.836  < 2e-16 ***
## monthmay                     -7.989e-02  6.752e-03 -11.833  < 2e-16 ***
## monthnov                     -9.972e-02  7.536e-03 -13.232  < 2e-16 ***
## monthoct                      2.166e-01  1.270e-02  17.055  < 2e-16 ***
## monthsep                      2.191e-01  1.392e-02  15.740  < 2e-16 ***
## day_of_weekmon               -1.351e-02  4.702e-03  -2.874 0.004058 **
## day_of_weekthu                8.190e-03  4.694e-03   1.745 0.081056 .
## day_of_weektue                1.090e-02  4.768e-03   2.286 0.022255 *
## day_of_weekwed                1.404e-02  4.759e-03   2.949 0.003188 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2995 on 41146 degrees of freedom
## Multiple R-squared:  0.1034, Adjusted R-squared:  0.1025
## F-statistic: 115.7 on 41 and 41146 DF,  p-value: < 2.2e-16
```

```
anova(lm(y~.,data=bankData))
```

```
## Analysis of Variance Table
##
## Response: y
##                Df Sum Sq Mean Sq  F value    Pr(>F)
## age             1    3.8   3.805  42.4057 7.503e-11 ***
## job            11   92.8   8.437  94.0328 < 2.2e-16 ***
## marital         3    8.1   2.687  29.9526 < 2.2e-16 ***
## education       7    5.8   0.823   9.1741 2.205e-11 ***
## default         2   31.2  15.583 173.6802 < 2.2e-16 ***
## housing         2    0.3   0.144   1.6023    0.2014
## loan            1    0.2   0.157   1.7461    0.1864
## contact         1   57.7  57.687 642.9512 < 2.2e-16 ***
## month           9  221.8  24.643 274.6612 < 2.2e-16 ***
## day_of_week     4    4.1   1.014  11.3004 3.651e-09 ***
## Residuals   41146 3691.7   0.090
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It turns out that seasonality is potentially very important here! Furthermore, retired, single people are more likely to say yes. Meanwhile housing and loan seem unimportant. Let's obtain a simpler model that we can interpret using leaps and regsubsets.

```
library('leaps')
basicSubset = regsubsets(y~.,data=bankData)
basicSummary = summary(basicSubset)

bestAIC = which.min(basicSummary$cp)
bestBIC = which.min(basicSummary$bic)
```

```
coef(basicSubset,bestBIC)
```

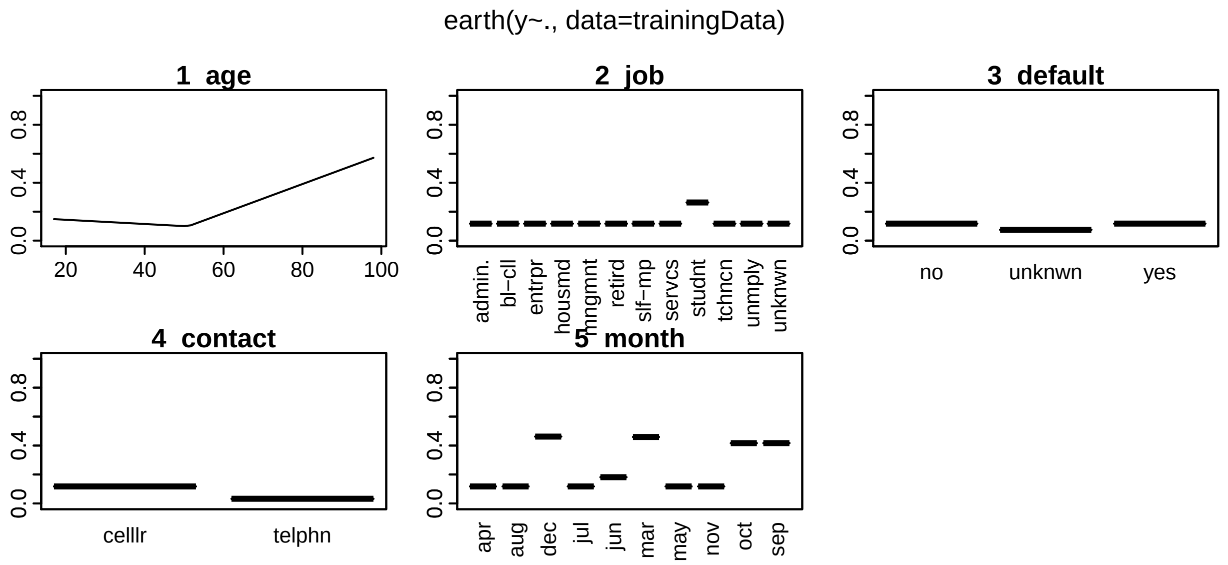
```
##    (Intercept)     jobretired     jobstudent defaultunknown       monthaug
##     0.15193255     0.12228655     0.17390554    -0.05937921    -0.04036319
##       monthjul       monthmar       monthmay       monthsep day_of_weekmon
##    -0.05030858     0.33279978    -0.07517011     0.26798870    -0.01867308
```

We learn that retired individuals and students are more likely to say yes, and that March and September are the best times to call.

Using MARS gives a clear plot of any non-linear relationships in the data.

```
library(earth)
earth1 = earth(y~.,data=trainingData)
plotmo(earth1)
```

```
##  plotmo grid:    age    job marital         education default housing loan
##                   38 admin. married university.degree      no     yes   no
##   contact month day_of_week
##  cellular   may         thu
```



We see that age has a non-linear relationship with acceptance, that December and March are the best times to call, and that retired individuals are more likely to say yes.

Predictive Modeling and Tuning

A predictive model tells you who is likely to be receptive to the calls from the bank. The bank can use this information to target their calls more effectively. We will do some basic tuning of the regression model, and MARS (I also tried nnet, but found the performance to be lacking)

```
#Test and Evaluate linear models.
lm1 = lm(y~age+factor(month),data=trainingData)
lm2 = lm(y~poly(age,3)+factor(month),data=trainingData)
lm3 = lm(y~.,data=trainingData)
lm4 = lm(y~.^2,data=trainingData)

mean((predict(lm1,validationData) - validationData$y)^2)
```

```
## [1] 0.09076575
```

```
mean((predict(lm2,validationData) - validationData$y)^2)
```

```
## [1] 0.08963436
```

```
mean((predict(lm3,validationData) - validationData$y)^2)
```

```
## [1] 0.0881642
```

```r
mean((predict(lm4,validationData) - validationData$y)^2)
```

```
## [1] 0.08765032
```

From this we learn that the full set of interactions seems to result in the best predictions. Let's use anova to learn what variable might be unimportant.

```r
anova(lm3)
```

```
## Analysis of Variance Table
##
## Response: y
##                Df  Sum Sq Mean Sq  F value    Pr(>F)    
## age             1    3.05   3.054  33.8763 5.927e-09 ***
## job            11   75.84   6.895  76.4909 < 2.2e-16 ***
## marital         3    6.95   2.318  25.7178 < 2.2e-16 ***
## education       7    5.31   0.758   8.4102 2.591e-10 ***
## default         2   25.03  12.515 138.8432 < 2.2e-16 ***
## housing         2    0.15   0.077   0.8554    0.4251    
## loan            1    0.22   0.215   2.3889    0.1222    
## contact         1   43.25  43.251 479.8297 < 2.2e-16 ***
## month           9  181.66  20.184 223.9236 < 2.2e-16 ***
## day_of_week     4    3.43   0.859   9.5256 1.078e-07 ***
## Residuals   32924 2967.71   0.090                       
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Housing/Loan seems unimportant. Lets delete those.

```r
lm5 = lm(y~.^2,data=trainingData[,-c(6,7)])
mean((predict(lm5,validationData) - validationData$y)^2)
```

```
## [1] 0.08724407
```

```r
anova(lm5)
```

```
## Analysis of Variance Table
##
## Response: y
##                           Df  Sum Sq Mean Sq  F value    Pr(>F)    
## age                        1    3.05   3.054  35.5458 2.517e-09 ***
## job                       11   75.84   6.895  80.2605 < 2.2e-16 ***
## marital                    3    6.95   2.318  26.9853 < 2.2e-16 ***
## education                  7    5.31   0.758   8.8247 6.843e-11 ***
## default                    2   25.03  12.515 145.6856 < 2.2e-16 ***
## contact                    1   43.32  43.316 504.2365 < 2.2e-16 ***
## month                      9  181.71  20.190 235.0240 < 2.2e-16 ***
## day_of_week                4    3.45   0.862  10.0293 4.139e-08 ***
## age:job                   11   13.27   1.207  14.0453 < 2.2e-16 ***
## age:marital                3    2.58   0.859   9.9979 1.393e-06 ***
## age:education              7    2.85   0.406   4.7314 2.530e-05 ***
## age:default                2    0.35   0.175   2.0412    0.1299    
## age:contact                1    0.02   0.017   0.1975    0.6568    
## age:month                  9   13.09   1.454  16.9289 < 2.2e-16 ***
## age:day_of_week            4    0.15   0.037   0.4309    0.7864    
## job:marital               33    3.08   0.093   1.0871    0.3353    
## job:education             71    6.57   0.093   1.0774    0.3070    
## job:default              11    4.32   0.393    4.5737 5.577e-07 ***
## job:contact              11    4.37   0.398    4.6296 4.322e-07 ***
## job:month                99   33.82   0.342    3.9770 < 2.2e-16 ***
## job:day_of_week          44    4.07   0.092    1.0767    0.3368
## marital:education        20    0.99   0.050    0.5775    0.9307
## marital:default           3    0.05   0.018    0.2067    0.8918
## marital:contact           3    0.08   0.027    0.3198    0.8111
## marital:month            25    5.49   0.220    2.5564 3.001e-05 ***
## marital:day_of_week      12    0.66   0.055    0.6404    0.8093
## education:default         7    0.52   0.074    0.8591    0.5382
## education:contact         7    0.71   0.102    1.1881    0.3055
## education:month          55   13.11   0.238    2.7741 4.457e-11 ***
## education:day_of_week    25    2.26   0.090    1.0502    0.3942
## default:contact           1    3.04   3.036   35.3442 2.791e-09 ***
## default:month             9    3.39   0.377    4.3836 9.623e-06 ***
## default:day_of_week       4    0.63   0.158    1.8378    0.1185
## contact:month             9   42.21   4.691   54.6019 < 2.2e-16 ***
## contact:day_of_week       4    0.54   0.134    1.5648    0.1806
## month:day_of_week        36   22.34   0.620    7.2222 < 2.2e-16 ***
## Residuals             32401 2783.39   0.086
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

That improved the predictive accuracy. Let's further adjust the regression formula based on the significant variables in that ANOVA:

```
lm6 = lm(y~age+job+marital+education+default+contact+factor(month)+factor(day_of_week)+age*job+age*mari
mean((predict(lm6,validationData) - validationData$y)^2)
```

```
## [1] 0.08648047
```

That also improve the model fit. You can use regression statistics helped to train a better predictive model. The above represents a somewhat trained lm model. We can tune an earth model as well, which will be able to capture non-linear relationships:

```
library('earth')

earth1 = earth(y~.,data=trainingData)
earth2 = earth(y~.,data=trainingData,degree=2)
earth3 = earth(y~age+job+marital+education+default+contact+factor(month)+factor(day_of_week)+age*job+age
#Earth 2 is the best of these

mean((predict(earth1,validationData) - validationData$y)^2)
```

```
## [1] 0.08834947
```

```
mean((predict(earth2,validationData) - validationData$y)^2)
```

```
## [1] 0.08739309
```

```
mean((predict(earth3,validationData) - validationData$y)^2)
```

```
## [1] 0.08752177
```

None of these fit as well as the regression model. Instead lets move the tuning parameter `thres` around.

```
earth4 = earth(y~.,data=trainingData,degree=2,thres=0)
earth5 = earth(y~.,data=trainingData,degree=2,thres=0.01)
earth6 = earth(y~.,data=trainingData,degree=2,thres=0.1)
```

```r
mean((predict(earth4,validationData) - validationData$y)^2)
```

```
## [1] 0.08661586
```

```r
mean((predict(earth5,validationData) - validationData$y)^2)
```

```
## [1] 0.08954148
```

```r
mean((predict(earth6,validationData) - validationData$y)^2)
```

```
## [1] 0.09787158
```

```r
#Earth 4 is the best of these
```

Finally, let's try a random forest.

```r
install.packages('randomForest', repos='http://cran.us.r-project.org')
```

```
## Installing package into 'C:/Users/owner/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)

## package 'randomForest' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\owner\AppData\Local\Temp\RtmpsPNRDO\downloaded_packages
```

```r
library('randomForest')
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```r
rf1 = randomForest(y~.,data=trainingData)
rf2 = randomForest(y~.^2,data=trainingData)

mean((predict(rf1,validationData) - validationData$y)^2)
```

```
## [1] 0.08675749
```

```r
mean((predict(rf2,validationData) - validationData$y)^2)
```

```
## [1] 0.08696158
```

After briefly tuning the model formula, degree, and thres, the mean squared error has improved From 0.091 to 0.087. This is actually a significant improvement in predictive terms for the amount of effort that was put in. Before the netflix prize, the model netflix used to suggest new movies had a mean squared error of 0.95. It took collaborations of the worlds best statisticians three year to reduce root mean squared error to 0.85. As you will see in the next section, even these small improvements in the predictive model can have a big effect on outcomes.

Benefits of Improving Predictive Fit

Suppose the firm was using these predictive models to choose which users to call in the future. Using the validation data, we can evaluate how successful each model would have been had it been used for this purpose. For each model, we will see which users in the validation data were predicted start an account. Then, we can check if those users ultimately did open an account. If the model makes good predictions, those who are predicted to open an account will, and the firm can avoid making costly, unnecessary calls.

First, let's rank users in order of their likelihood to say yes according to our three models:

```r
lm1CallRanking = order(predict(lm1,validationData),decreasing = TRUE)
earth1CallRanking = order(predict(earth1,validationData),decreasing = TRUE)
#Best, partially tuned model:
earth4CallRanking = order(predict(earth4,validationData),decreasing = TRUE)
```

Now, we can check how frequently the top 2000 users identified by each model actually said yes:

```r
nCalls = 2000
#If we called randomly, we'd get the average success rate
randomSuccess = round(mean(validationData$y)*nCalls)

#Below is the success rate using different models
lm1Success = sum(validationData$y[lm1CallRanking[1:nCalls]])
earth1Success = sum(validationData$y[earth1CallRanking[1:nCalls]])
earth4Success = sum(validationData$y[earth4CallRanking[1:nCalls]])
print(c(paste(randomSuccess),paste(lm1Success),paste(earth1Success),paste(earth4Success)))
```

```
## [1] "220" "393" "484" "510"
```

If the firm were to do 2000 calls, they would get 220 successes if they chose randomly. Using a statistical model boosts the success rate. The linear regression would have had 393 successes, while the two earth models have 484 and 510 successes. Fitting tuned, predictive models has a clear effect on how successful these call would be. Better tuned models will lead to even greater successes.

Causal Questions

Can we make causal inferences with this dataset? That is, are there any of these coefficients that We might interpret as causal?

Consider the full set of variables in the data: age, job, martial, education, default, housing, loan, contact, month, day_of_week.

1. Which variables would the firm be interesting in manipulating? What could they change?
2. Of those variables, what could bias the observed effects?
3. What can the firm conclude with confidence?

Bonus/Optional Module: Neuralnet and Deep Learning.

This module is not part of the official course content, but it was requested by several students. The material is not testable, and it will not appear on any assignment.

In this module we will estimate a neural network using the 'neuralnet' package. This package isn't as robust as nnet, and it does not use the predict function. However, it can implement 'deep learning' (aka multilayer) neural networks. The predictive performance of these untuned models is weaker than that of earth in the previous module. However, with proper tuning, they might be stronger. Be warned, these models take a long time to estimate.

An online tutorial can be found here: http://www.learnbymarketing.com/tutorials/neural-networks-in-r-tutorial/

```r
#Install the package
install.packages('neuralnet', repos='http://cran.us.r-project.org')
```

```
## Installing package into 'C:/Users/owner/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
## also installing the dependency 'Deriv'

##
##   There is a binary version available but the source version is
##   later:
##       binary source needs_compilation
## Deriv    4.0  4.1.1             FALSE
##
## package 'neuralnet' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\owner\AppData\Local\Temp\RtmpsPNRDO\downloaded_packages

## installing the source package 'Deriv'
```

```r
library('neuralnet')
bankDataMat = model.matrix(~.,data=bankData)

#Rename some of the columns so the method works correctly
colnames(bankDataMat)[3] <- "jobbluecollar"
colnames(bankDataMat)[8] <- "jobselfemployed"

#Split into training/validatoin data
trainingMat = bankDataMat[isTraining,]
validationMat = bankDataMat[!isTraining,]


#Generate a correct formula
col_list <- paste(c(colnames(validationMat[,-c(1,44)])),collapse="+")
col_list <- paste(c("yTRUE~",col_list),collapse="")
f <- formula(col_list)


#This fit a simple neural network with 3 units in the hidden layer.
basicSingleLayerNNet <- neuralnet(f, data=trainingMat,algorithm = "rprop+",hidden=c(3),threshold=0.1,st

#Get predictions for the validation data (this is super finicky)
output <- compute(basicSingleLayerNNet, validationMat[,-c(1,44)],rep=1)
#Calculate out of sample performance
mean((validationMat[,44] - output$net.result)^2)
```

```
## [1] 0.08760814
```

This next section fits a deep learning neural network. Notice that 'hidden' is now a vector that equal (3,3). This means that a neural network with two hidden layers, each with 3 nodes, will be fit. The second model has 10 units in the first layer, and 3 units in the second.

```r
#This will take a lot of time to compute.
deepLearningNnet1 <- neuralnet(f, data=trainingMat,algorithm = "rprop+",hidden=c(3,3),
                               threshold=0.1,stepmax = 1e+06)
output <- compute(deepLearningNnet1, validationMat[,-c(1,44)],rep=1)
mean((validationMat[,44] - output$net.result)^2)
```

```
## [1] 0.08925159
```

```r
plot(deepLearningNnet1)
```

```r
deepLearningNnet2 <- neuralnet(f, data=trainingMat,algorithm = "rprop+",hidden=c(10,3),
                               threshold=0.1,stepmax = 1e+06)
output <- compute(deepLearningNnet2, validationMat[,-c(1,44)],rep=1)
mean((validationMat[,44] - output$net.result)^2)
```

```
## [1] 0.09185171
```

```r
plot(deepLearningNnet2)
```