Practicum3

Josiah Veloso

November 20, 2018

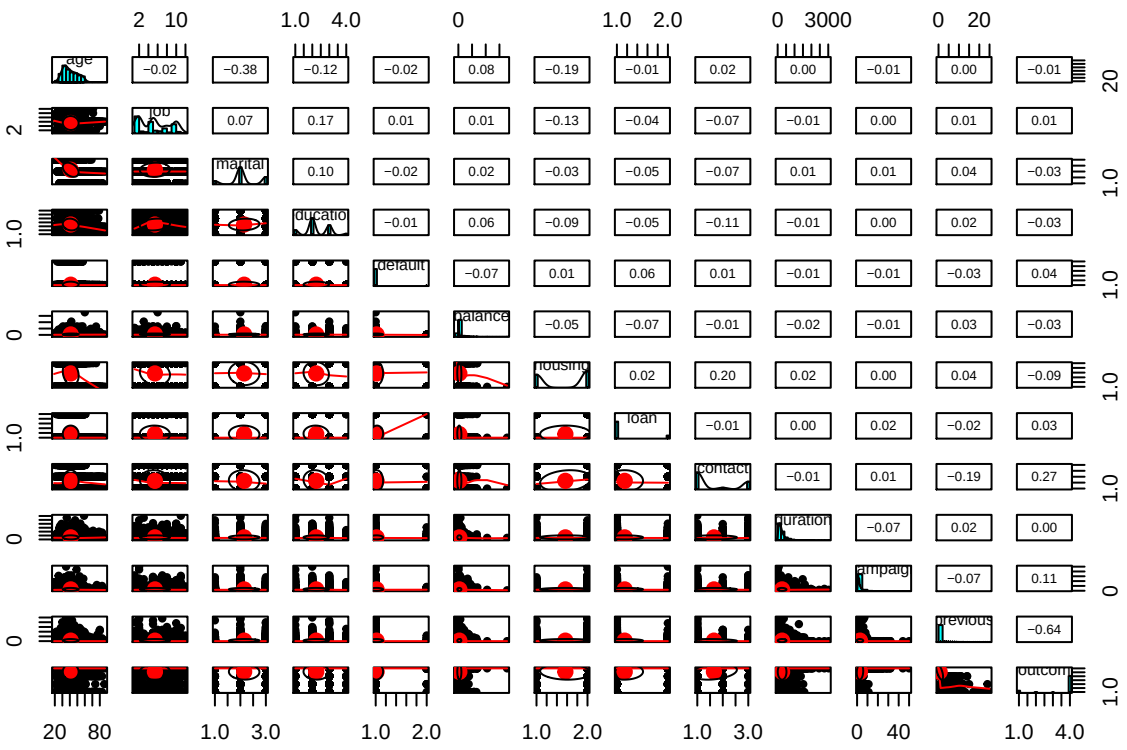
```
bank<-read.csv("bank.csv", sep = ";") #noticably the separator is a ":" - output variable "y" is expect
str(bank)
```

```
## 'data.frame':    4521 obs. of  17 variables:
##  $ age      : int  30 33 35 30 59 35 36 39 41 43 ...
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 11 8 5 5 2 5 7 10 3 8 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 2 2 3 2 2 3 2 2 2 2 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 1 2 3 3 2 3 3 2 3 1 ...
##  $ default  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ balance  : int  1787 4789 1350 1476 0 747 307 147 221 -88 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 1 2 2 2 2 1 2 2 2 2 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 1 2 1 2 1 1 1 1 1 2 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 3 3 1 1 1 3 1 ...
##  $ day      : int  19 11 16 3 5 23 14 6 14 17 ...
##  $ month    : Factor w/ 12 levels "apr","aug","dec",..: 11 9 1 7 9 4 9 9 9 1 ...
##  $ duration : int  79 220 185 199 226 141 341 151 57 313 ...
##  $ campaign : int  1 1 1 4 1 2 1 2 2 1 ...
##  $ pdays    : int  -1 339 330 -1 -1 176 330 -1 -1 147 ...
##  $ previous : int  0 4 1 0 0 3 2 0 0 2 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 4 1 1 4 4 1 2 4 4 1 ...
##  $ y        : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

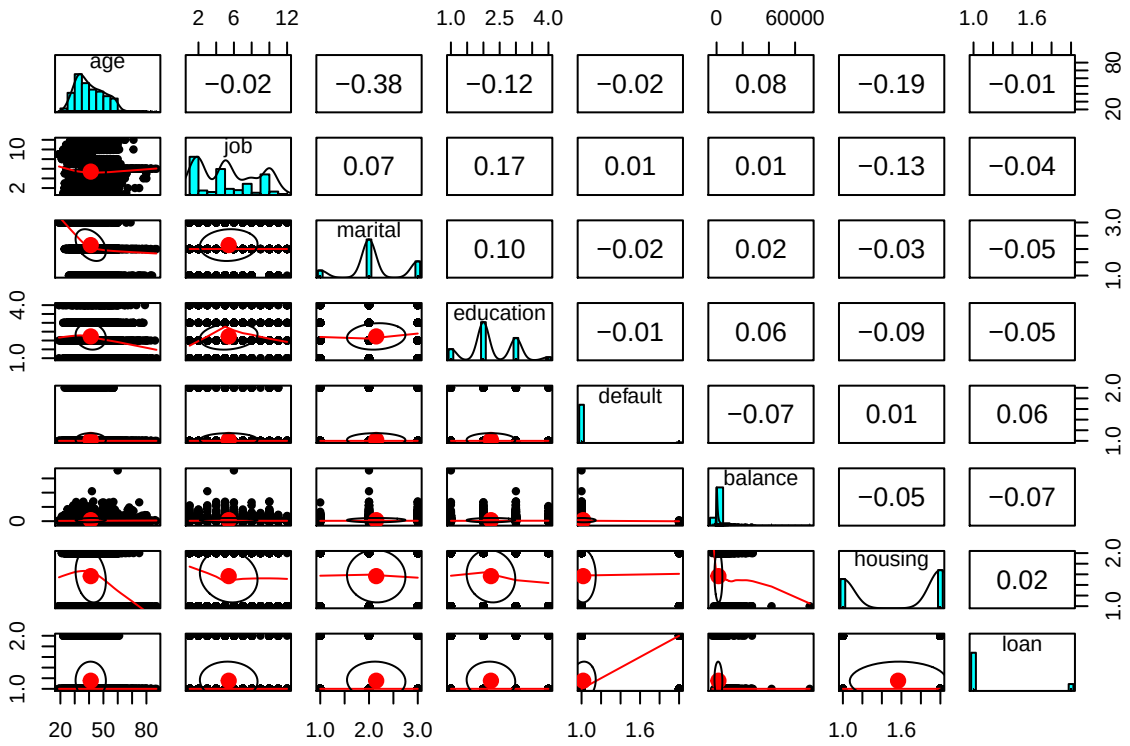
```
library(psych)
```

```
## Warning: package 'psych' was built under R version 3.4.4
```

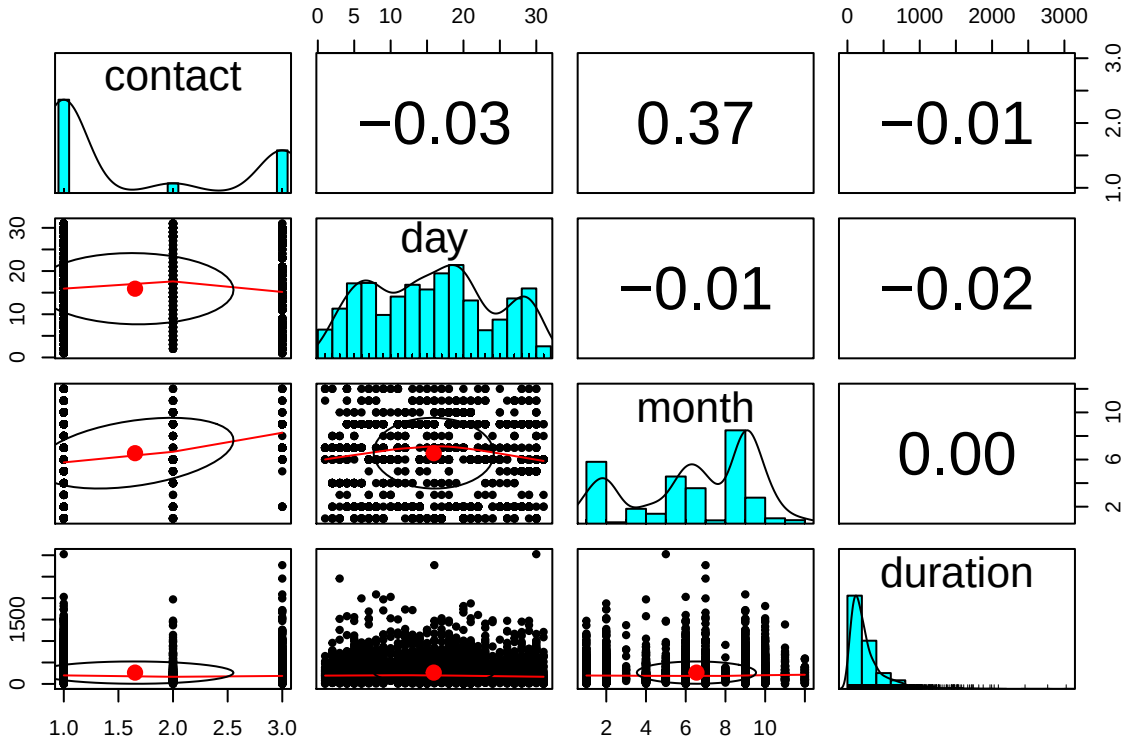
```
pairs.panels(bank[c("age","job","marital","education","default","balance","housing","loan","contact","du
```



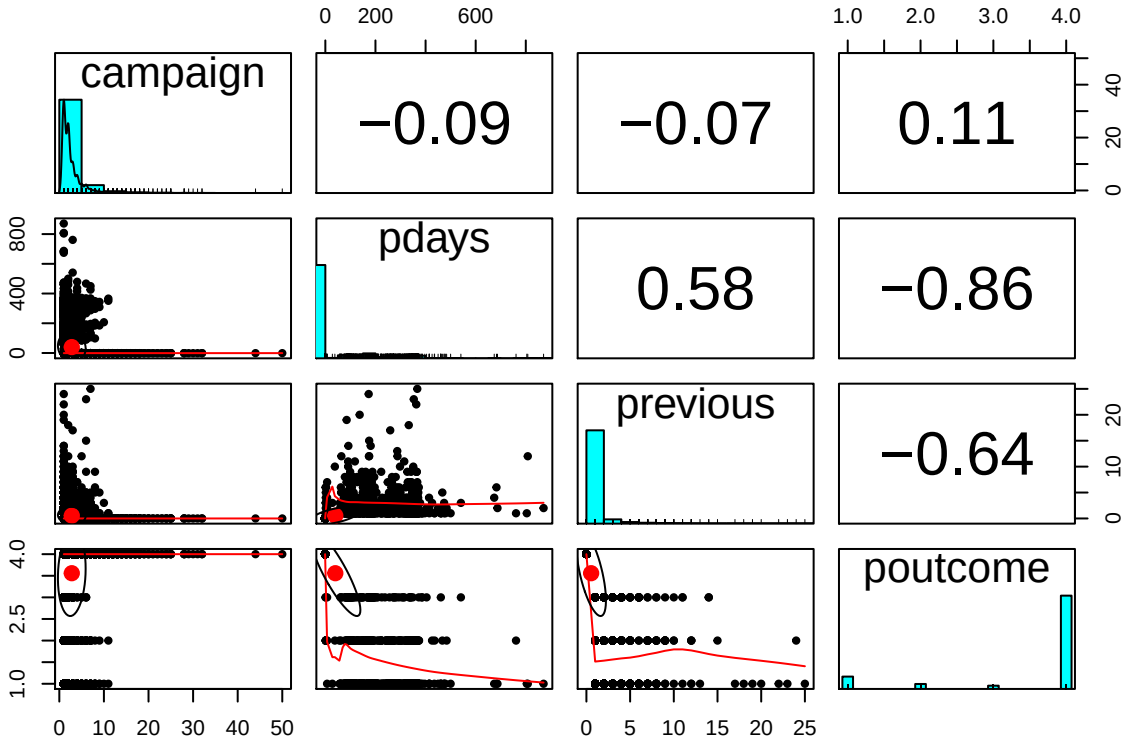

```
pairs.panels(bank[c("age","job","marital","education","default","balance","housing","loan")]) #"Bank cl
```




```
pairs.panels(bank[c("contact","day","month","duration")]) #"Related with the last contact of the curren
```




```
pairs.panels(bank[c("campaign","pdays","previous","poutcome")]) #"Other attributes" section
```




```
#it seems that not only are most of these variables binary but they are also barely correlated to each
#this is because "poutcome" literally is resultant from the amount of calls in "previous". Of course sa
#notably day and month are not correlated because I guess the callers called consistently over the seve
#if we were to use a ML package that expects normalization "job", "marital" , "education" and especiall
#out of all of them "age","duration" and to some extent "campaign" and "balance" (if looked really clos
summary(bank$month) #I noticed there was plenty of more calls during the summer months of may through a
```

```
##  apr  aug  dec  feb  jan  jul  jun  mar  may  nov  oct  sep
##  293  633   20  222  148  706  531   49 1398  389   80   52
```

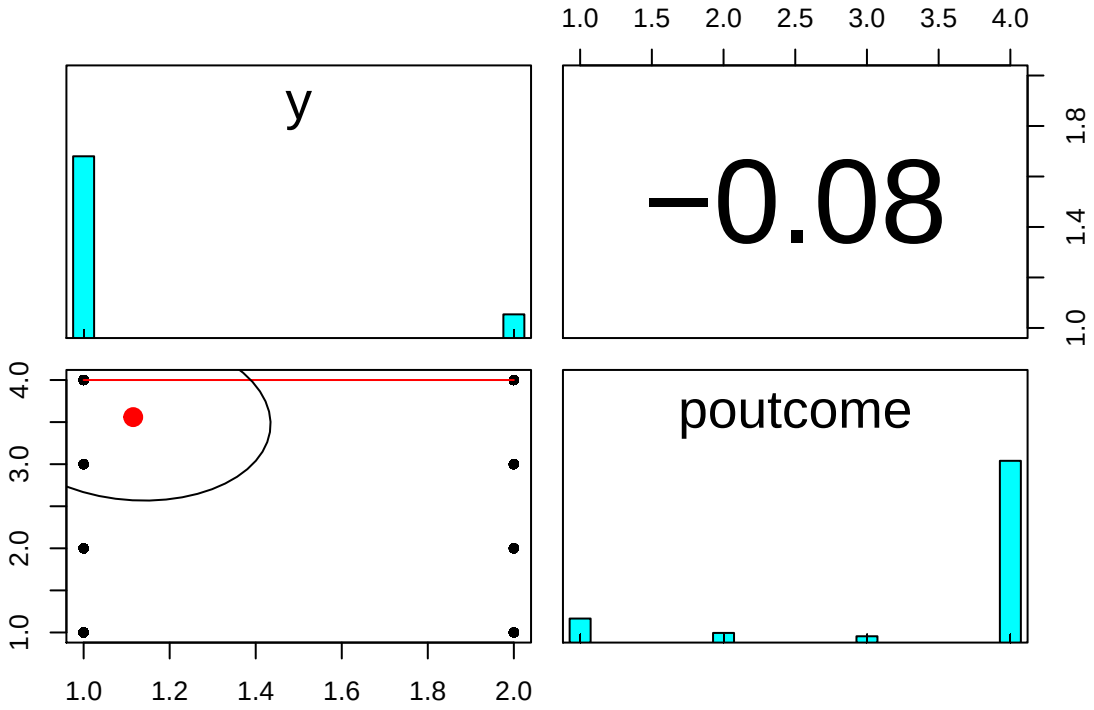
```
summary(bank$campaign) #apparently there was one contact that was contacted 50 times compared to the us
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   2.794   3.000  50.000
```

```
table(bank$y, useNA="ifany") #There is actually no NA values, notably they were able to convert ~13 per
```

```
##
##   no  yes
## 4000  521
```

```
#Because SVM and Neural networks preferably use numerical values also to prevent any kind of unecessary
#will be removed from the prediction features at the "Bank client data" section.
#I believe "contact" and "day" features are negligable features however "duration" (which can denote in
#Although if I were to use "month" I might need to dummy code binary columns denoting each month 11 tim
#In the "other" section "pdays" and "previous" seem redundant and shows high correlation to "poutcome"
#Note I must binary/dummy code "poutcome" to take into account success only - as I assume that is what
pairs.panels(bank[c("y","poutcome")])
```




```
#Logically I would think "poutcome" would be somewhat correlated to "y" however the correlation calcula
#(which literally do not match the factors in "y") are affecting the calculation.
```

Overall I would pick features such as age, default, balance, housing, loan, month (which would need binary dummy coding - for specifically summer months), duration, campaign and poutcome (also needing binary dummy coding towards successes) to use for Machine Learning for SVMs not only to minimize for simplicity but to get rid of potential redundancy and unneeded potential discrimination. Also hopefully for sensible logical reasons as explained above.

```
library(kernlab)
```

```
## Warning: package 'kernlab' was built under R version 3.4.4
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:psych':
##
##     alpha
```

```
normalize<-function(x){
  return((x-min(x))/(max(x)-min(x)))
} #min max normalization prepared
#dummy code for summer months
bank$Summ<-ifelse(as.character(as.vector(bank$month))=="may" , 1,
                  ifelse(as.character(as.vector(bank$month))=="jun", 1,
                         ifelse(as.character(as.vector(bank$month))=="jul", 1,
                                ifelse(as.character(as.vector(bank$month))=="aug", 1, 0))))
#Conversion of selected categorical values "month" and "poutcome"
```

```r
summary(bank$month)
```

```
##  apr  aug  dec  feb  jan  jul  jun  mar  may  nov  oct  sep
##  293  633   20  222  148  706  531   49 1398  389   80   52
```

```r
summary(bank$Summ) #Confirms most of the calls were made at summer so my assumption has a degree of rea
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.7228  1.0000  1.0000
```

```r
#dummy coding for poutcome taking into account confirmed successes
bank$pSuccess<-ifelse(as.character(as.vector(bank$poutcome))=="success",1,0)
summary(bank$pSuccess) #notably they only converted around ~3% previously (taken from the Mean)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.02853 0.00000 1.00000
```

```r
#- this actually matches up almost to the current data confirming some viability to my assumption of us
#Conversion of binary categorical values into binary numericals
bank$defaultB<-ifelse(as.character(as.vector(bank$default))=="yes",1,0) #binary coding default
bank$housingB<-ifelse(as.character(as.vector(bank$housing))=="yes",1,0) #binary coding housing
bank$loanB<-ifelse(as.character(as.vector(bank$loan))=="yes",1,0) #binary coding loan
bank$By<-ifelse(as.character(as.vector(bank$y))=="yes",1,0) #binary coding y
bank_prep<-as.data.frame(bank[,c(1,20,6,21,22,18,12,13,19,17)])#pre normalization version of new bank d
#using features age+defaultB+balance+housingB+loanB+Summ+duration+campaign+pSuccess and y from original
bank_norm<-as.data.frame(lapply(bank_prep[,1:9],normalize))#applying normalization and storing to new d
bank_norm$y<-bank_prep[,10]
subs_train<-bank_norm[1:3617,] #train ~80 percent of data
subs_test<-bank_norm[3618:4521,] #test ~20 percent of data
```

```r
subs_classifier<-ksvm(y~age+defaultB+balance+housingB+loanB+Summ+duration+campaign+pSuccess, data=subs_
```

```
##  Setting default kernel parameters
```

```r
#I assumed sensibly factors/features I picked from the last question could affect the conversion aka "y
#I also confirmed by adding the other variables if they even affected accuracy by adding them in and it
#This package of svm apparently normalizes and scales the values automatically for use but I will have
subs_classifier
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 822
##
## Objective Function Value : -766.0543
## Training error : 0.105889
```

```r
#svm function derived from package using all features to determine "y" note this kernel separator is li
#subs_classifier #apparently the training error is ~.10.5 percent
```

```r
sub_predictions<-predict(subs_classifier,subs_test)#predict using svm
table(sub_predictions,subs_test$y)
```

```
##
```

```
## sub_predictions  no yes
##             no  784  92
##             yes   9  19
```

```
Accuracy<-sub_predictions==subs_test$y #bool to compare accuracy of classifications
table(Accuracy)
```

```
## Accuracy
## FALSE  TRUE
##   101   803
```

```
prop.table(table(Accuracy)) # approximately really similiar error rate of ~11.2% of predicted ~10.5% er
```

```
## Accuracy
##     FALSE      TRUE
## 0.1117257 0.8882743
```

```
#apparently has an accuracy of 88 percent - note the algorithim seems to be geared towards less false p
```

The data shows they will have converted 3%~ (28/876) of the contacts previously however the algorithim predicts up to 2.4%~ (19/784) correctly. The best fit line(s) splitting optimal linear solutions in SVM must have struggled alot with keeping the "no's" inside due to the large amount of false positives (actual no's vs predicted yes's). Maybe using another kernel would be more optimal in splitting less rigidly.

```
bank_prep2<-as.data.frame(bank[,c(1,20,6,21,22,18,12,13,19,23)])#pre normalization version of new bank
#using features age+defaultB+balance+housingB+loanB+Summ+duration+campaign+pSuccess and y from original
bank_norm2<-as.data.frame(lapply(bank_prep2,normalize))#applying normalization and storing to new data
subsN_train<-bank_norm2[1:3617,] #train ~80 percent of data
subsN_test<-bank_norm2[3618:4521,] #test ~20 percent of data
```

```
#installed neuralnet package
library(neuralnet)
```

```
## Warning: package 'neuralnet' was built under R version 3.4.4
```

```
set.seed(100)
subs_model<-neuralnet(By~age+defaultB+balance+housingB+loanB+Summ+duration+campaign+pSuccess, data=subsl
#initial test with neural net model made with a default single neuron with the several features to take
plot(subs_model)

model_results<-compute(subs_model,subsN_test[1:9]) #tests using features involved
predicted_strength<-model_results$net.result #probability collection of subs_model
cor(predicted_strength,subsN_test$By) #checking correlation - - of ~.534 which is shows these values ar
```

```
##             [,1]
## [1,] 0.533525813
```

```
set.seed(100)
subs_model2<-neuralnet(By~age+defaultB+balance+housingB+loanB+Summ+duration+campaign+pSuccess, data=subs
#2nd test with neural net model made with 3 hidden perceptrons with the several features to take into a
plot(subs_model2)

model_results2<-compute(subs_model2,subsN_test[1:9]) #tests using features involved
predicted_strength2<-model_results2$net.result #probability collection of subs_model2
cor(predicted_strength2,subsN_test$By) #checking correlation - of ~.530 which is shows these values are
```

```
##              [,1]
## [1,] 0.5302828208
```

```
#HOWEVER the prediction actually correlation somehow SLIGHTLY got worse, maybe not enough neurons to ge
```

```
pred1<-ifelse(predicted_strength>=.5,1,0) #converting .5 probability thresholds into assumed prediction
pred2<-ifelse(predicted_strength2>=.5,1,0)
table(pred1,as.factor(subsN_test$By))
```

```
##
## pred1   0   1
##     0 769  70
##     1  24  41
```

```
table(pred2,as.factor(subsN_test$By)) #the 3 hidden perceptron version noticably got less false positiv
```

```
##
## pred2   0   1
##     0 774  75
##     1  19  36
```

```
AccuracyNN1<-as.data.frame(pred1)==subsN_test$By #bool to compare accuracy of single neuron prediction
table(AccuracyNN1) #~10.3 error rate!
```

```
## AccuracyNN1
## FALSE  TRUE
##    94   810
```

```
AccuracyNN2<-as.data.frame(pred2)==subsN_test$By #bool to compare accuracy of 3 hidden neuron predictio
table(AccuracyNN2) #~10.3% error rate as well!
```

```
## AccuracyNN2
## FALSE  TRUE
##    94   810
```

```
#Literally the same amount of mistakes and accuracy albeit the 3 hidden perceptron version is more pron
#in this situation if the bank wants to have a slightly wider call net to get every new client and is f
#- the bank should go with defaultly the no hidden perceptron version of pred1
```

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 3.4.4
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
#subs_test$By<-ifelse(as.character(as.vector(subs_test$y))=="yes",1,0)
subs_predictionsB<-ifelse(as.character(as.vector(sub_predictions))=="yes",1,0) #binary coding for AUC u
roc_svm <- roc(subs_test$y,subs_predictionsB)
auc(roc_svm)
```

```
## Area under the curve: 0.5799109
```

```
roc_NN<-roc(subsN_test$By,as.numeric(pred1)) #choosing pred1 for preferred likelihood of more true posi
auc(roc_NN) #neural network is apparently superior fitting ~67% AUC compared to the rigid svm using a l
```

```
## Area under the curve: 0.6695523
```

```
#SVM accuracy: 88.8% (803/904)
#Neural Network accuracy: 89.6% (810/904)
#Regardless of the fact the SVM's linear kernel method was probably very rigid - the accuracy is very s
#the NN model has slightly better accuracy and a better fitting model as represented by the AUC.
#From my observations - the more rigid the predictions are - the more false negatives there would be, s
#Whether that be in the form of more perceptrons in NN or using a very rigid kernel for your SVM algori
```