Bank Authentication Project

Kevin Torres

**Task:** I will be using Wavelet transformation data to see if a bank note is authentic or fake by building a neural net and making predictions based off it. I will then compare accuracy using a random forest model.

**Dataset:** The dataset consists of statistical information of images, therefore exploratory data analysis of this data is not easily interpretable.

Load csv file:

bank <- read.csv('bank\_note\_data.csv')  
head(bank)

## Image.Var Image.Skew Image.Curt Entropy Class  
## 1 3.62160 8.6661 -2.8073 -0.44699 0  
## 2 4.54590 8.1674 -2.4586 -1.46210 0  
## 3 3.86600 -2.6383 1.9242 0.10645 0  
## 4 3.45660 9.5228 -4.0112 -3.59440 0  
## 5 0.32924 -4.4552 4.5718 -0.98880 0  
## 6 4.36840 9.6718 -3.9606 -3.16250 0

str(bank)

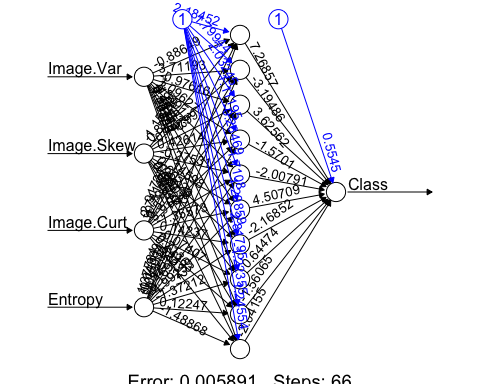
## 'data.frame': 1372 obs. of 5 variables:  
## $ Image.Var : num 3.622 4.546 3.866 3.457 0.329 ...  
## $ Image.Skew: num 8.67 8.17 -2.64 9.52 -4.46 ...  
## $ Image.Curt: num -2.81 -2.46 1.92 -4.01 4.57 ...  
## $ Entropy : num -0.447 -1.462 0.106 -3.594 -0.989 ...  
## $ Class : int 0 0 0 0 0 0 0 0 0 0 ...

Load necessary libraries and create train and test data sets

library(caTools)  
  
split <- sample.split(bank$Class, SplitRatio = .7)  
train <- subset(bank, split == T)  
test <- subset(bank, split == F)

Build the Neural Net

library(neuralnet)  
nn.bank <- neuralnet(Class ~ Image.Var + Image.Skew + Image.Curt + Entropy, data = train, hidden = 10, linear.output = F )  
plot(nn.bank, rep = 'best')



Predictions

predicted.nn.values <- compute(nn.bank, test[,1:4])  
str(predicted.nn.values)

## List of 2  
## $ neurons :List of 2  
## ..$ : num [1:412, 1:5] 1 1 1 1 1 1 1 1 1 1 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ : chr [1:412] "1" "3" "4" "7" ...  
## .. .. ..$ : chr [1:5] "" "Image.Var" "Image.Skew" "Image.Curt" ...  
## ..$ : num [1:412, 1:11] 1 1 1 1 1 1 1 1 1 1 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ : chr [1:412] "1" "3" "4" "7" ...  
## .. .. ..$ : NULL  
## $ net.result: num [1:412, 1] 6.73e-05 3.58e-05 9.17e-05 1.25e-05 1.16e-05 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ : chr [1:412] "1" "3" "4" "7" ...  
## .. ..$ : NULL

head(predicted.nn.values$net.result)

## [,1]  
## 1 6.726342e-05  
## 3 3.577231e-05  
## 4 9.171594e-05  
## 7 1.251290e-05  
## 13 1.164716e-05  
## 16 1.469538e-05

Here we notice the net results are still probabilities and we could use the round function to fix this

predictions <- sapply(predicted.nn.values$net.result, round)  
head(predictions)

## [1] 0 0 0 0 0 0

Create a confusion matrix to see how we predicted

table(predictions, test$Class)

##   
## predictions 0 1  
## 0 229 0  
## 1 0 183

# We should be suspicious of perfect results since we did not even normalize the data  
# We would typically normalize our data if there is a large range of min and max values between the column features

Compare to a Random Forest Model

Load necessary libraries and create train and test data sets

library(randomForest)

## randomForest 4.6-14

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

# First we need to set the Class column to be a factor, not an int like neural nets  
bank$Class <- factor(bank$Class)  
  
split <- sample.split(bank$Class, SplitRatio = .7)  
train <- subset(bank, split == T)  
test <- subset(bank, split == F)

Build the model

nn.rf.model <- randomForest(Class ~ ., train)  
nn.rf.model$confusion

## 0 1 class.error  
## 0 529 4 0.007504690  
## 1 2 425 0.004683841

Now we can predict

rf.model.predict <- predict(nn.rf.model, test)  
table(rf.model.predict, test$Class)

##   
## rf.model.predict 0 1  
## 0 226 1  
## 1 3 182

# This model was almost perfect, therefore we can conclude that we should not be suspicious of our perfect neural net model