R Notebook

In this notebook we will use ANN to predict the market movement

```r
library(quantmod)
```

```
## Warning: package 'quantmod' was built under R version 4.0.5
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 4.0.4
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
getSymbols("^DJI", src="yahoo")
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## [1] "^DJI"
```

```r
dow_jones <- DJI[,"DJI.Close"]

return <- Delt(dow_jones)

average10 <- rollapply(dow_jones,10,mean)
average20 <- rollapply(dow_jones,20,mean)

std10 <- rollapply(dow_jones,10,sd)
std20 <- rollapply(dow_jones,20,sd)

rsi5 <- RSI(dow_jones,5,"SMA")
rsi14 <- RSI(dow_jones,14,"SMA")

macd12269 <- MACD(dow_jones,12,26,9,"SMA")
macd7205 <- MACD(dow_jones,7,20,5,"SMA")

bollinger_bands <- BBands(dow_jones,20,"SMA",2)

direction <- data.frame(matrix(NA,dim(dow_jones)[1],1))

lagreturn <- (dow_jones - Lag(dow_jones,20)) / Lag(dow_jones,20)

direction[lagreturn > 0.02] <- "1"
direction[lagreturn < -0.02] <- "-1"
direction[lagreturn < 0.02 & lagreturn > -0.02] <- "0"

dow_jones <- cbind(dow_jones,average10,average20,std10,std20,rsi5,rsi14,macd12269,macd7205,bollinger_ba

train_sdate<- "2010-01-01"
train_edate<- "2013-12-31"
vali_sdate<- "2016-01-01"
vali_edate<- "2016-12-31"
test_sdate<- "2017-01-01"
test_edate<- "2017-09-10"

trainrow<- which(index(dow_jones) >= train_sdate& index(dow_jones) <= train_edate)
valirow<- which(index(dow_jones) >= vali_sdate& index(dow_jones) <= vali_edate)
testrow<- which(index(dow_jones) >= test_sdate& index(dow_jones) <= test_edate)

traindji<- dow_jones[trainrow,]
validji<- dow_jones[valirow,]
testdji<- dow_jones[testrow,]
```

```r
trainme <- apply(traindji,2,mean)
trainstd <- apply(traindji,2,sd)
```

```r
trainidn <- (matrix(1,dim(traindji)[1],dim(traindji)[2]))
valiidn <- (matrix(1,dim(validji)[1],dim(validji)[2]))
testidn <- (matrix(1,dim(testdji)[1],dim(testdji)[2]))
```

Normalizing the three datasets and only then can we use them in the neural networks. We will do it by using the mean and the std

```r
norm_traindji <- (traindji - t(trainme*t(trainidn)))/t(trainstd*t(trainidn))
norm_validji <- (validji - t(trainme*t(valiidn)))/t(trainstd*t(valiidn))
norm_testdji <- (testdji - t(trainme*t(testidn)))/t(trainstd*t(testidn))
```

```r
traindir <- direction[trainrow,1]
validir <- direction[valirow,1]
testdir <- direction[testrow,1]
```

This is the NNET package that is going to help us with the

```r
#install.packages("nnet")
library(nnet)
```

```
## Warning: package 'nnet' was built under R version 4.0.5
```

Implementing ANN

```r
set.seed(1)
neural_network <- nnet(norm_traindji,class.ind(traindir),size = 4, trace = T)
```

```
## # weights:  79
## initial  value 898.113450
## iter  10 value 269.711770
## iter  20 value 190.530198
## iter  30 value 171.952660
## iter  40 value 161.227273
## iter  50 value 148.005807
## iter  60 value 140.509509
## iter  70 value 137.051225
## iter  80 value 135.296999
## iter  90 value 134.927918
## iter 100 value 134.611833
## final  value 134.611833
## stopped after 100 iterations
```

```r
neural_network
```

```
## a 15-4-3 network with 79 weights
## options were -
```

```r
dim(norm_traindji)
```

```
## [1] 1006   15
```

```r
vali_pred <- predict(neural_network,norm_validji)
head(vali_pred)
```

```
##            -1          0 1
## 2016-01-04  1 0.03919498 0
## 2016-01-05  1 0.03919498 0
## 2016-01-06  1 0.03919498 0
## 2016-01-07  1 0.03919498 0
## 2016-01-08  1 0.03919498 0
## 2016-01-11  1 0.03919498 0
```

```
vali_pred_class <- data.frame(matrix(NA,dim(vali_pred)[1],1))

vali_pred_class[vali_pred[,"-1"]>0.5,1] <- "-1"
vali_pred_class[vali_pred[,"1"]>0.5,1] <- "1"
vali_pred_class[vali_pred[,"0"]>0.5,1] <- "0"
```

Now we will be checking the forecast accuracy using the caret library

```
#install.packages("caret")
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.5
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.0.4
```

```
matrix <- confusionMatrix(factor(vali_pred_class[,1],levels=1:2),factor(validir,levels = 1:2))
matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 89  0
##          2  0  0
##
##                Accuracy : 1
##                  95% CI : (0.9594, 1)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : NaN
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity :  1
##             Specificity : NA
##          Pos Pred Value : NA
##          Neg Pred Value : NA
##              Prevalence :  1
```

```
##          Detection Rate :  1
##    Detection Prevalence :  1
##       Balanced Accuracy : NA
##
##        'Positive' Class : 1
##
```

```r
class(vali_pred_class[,1])
```

```
## [1] "character"
```

```r
class(validir)
```

```
## [1] "character"
```

```r
#confusionMatrix(validir,vali_pred_class[, 1])
```