#Technical Analysis

library(RCurl)

get\_price <- function(){

a <- getURL("https://www.bitcoinwisdom.com/markets/bitstamp/btcusd", ssl.verifypeer=0L, followlocation=1L)

n <- as.numeric(regexpr("id=market\_bitstampbtcusd>", a))

a <- substr(a, n, n + 100)

n <- as.numeric(regexpr(">", a))

m <- as.numeric(regexpr("</span>", a))

a <- substr(a, n + 1, m - 1)

as.numeric(a)

}

library(XML)

as.numeric(xpathApply(htmlTreeParse(a, useInternalNodes = TRUE),

'//span[@id="market\_bitstampbtcusd"]', xmlValue)[[1]])

DrawChart <- function(time\_frame\_in\_minutes,

number\_of\_candles = 25, l = 315.5, u = 316.5) {

OHLC <- matrix(NA, 4, number\_of\_candles)

OHLC[, number\_of\_candles] <- get\_price()

dev.new(width = 30, height = 15)

par(bg = rgb(.9, .9, .9))

plot(x = NULL, y = NULL, xlim = c(1, number\_of\_candles + 1),

ylim = c(l, u), xlab = "", ylab = "", xaxt = "n", yaxt = "n")

abline(h = axTicks(2), v = axTicks(1), col = rgb(.5, .5, .5), lty = 3)

axis(1, at = axTicks(1), las = 1, cex.axis = 0.6,

labels = Sys.time() - (5:0) \* time\_frame\_in\_minutes)

axis(2, at = axTicks(2), las = 1, cex.axis = 0.6)

box()

allpars = par(no.readonly = TRUE)

while(TRUE) {

start\_ <- Sys.time()

while(as.numeric(difftime(Sys.time(), start\_, units = "mins")) <

time\_frame\_in\_minutes) {

OHLC[4,number\_of\_candles] <- get\_price()

OHLC[2,number\_of\_candles] <- max(OHLC[2,number\_of\_candles],

OHLC[4,number\_of\_candles])

OHLC[3,number\_of\_candles] <- min(OHLC[3,number\_of\_candles],

OHLC[4,number\_of\_candles])

frame()

par(allpars)

abline(h = axTicks(2), v=axTicks(1), col = rgb(.5,.5,.5), lty = 3)

axis(1, at = axTicks(1), las = 1, cex.axis = 0.6,

labels = Sys.time()-(5:0)\*time\_frame\_in\_minutes)

axis(2, at = axTicks(2), las = 1, cex.axis = 0.6)

box()

for(i in 1:number\_of\_candles) {

polygon(c(i, i + 1, i + 1, i),

c(OHLC[1, i], OHLC[1, i], OHLC[4, i], OHLC[4, i]),

col = ifelse(OHLC[1,i] <= OHLC[4,i],

rgb(0,0.8,0), rgb(0.8,0,0)))

lines(c(i+1/2, i+1/2), c(OHLC[2,i], max(OHLC[1,i], OHLC[4,i])))

lines(c(i+1/2, i+1/2), c(OHLC[3,i], min(OHLC[1,i], OHLC[4,i])))

}

abline(h = OHLC[4, number\_of\_candles], col = "green",

lty = "dashed")

}

OHLC <- OHLC[, 2:number\_of\_candles]

OHLC <- cbind(OHLC, NA)

OHLC[1,number\_of\_candles] <- OHLC[4,number\_of\_candles-1]

}

}

DrawChart(30,50)

library(quantmod)

bitcoin <- read.table("Bitcoin.csv", header = T, sep = ";", row.names = 1)

bitcoin <- tail(bitcoin, 150)

bitcoin <- as.xts(bitcoin)

dev.new(width = 20, height = 10)

chartSeries(bitcoin, dn.col = "red", TA="addRSI(10);addEMA(10)")

library(quantmod)

bitcoin <- read.table("Bitcoin.csv", header = T, sep = ";", row.names = 1)

bitcoin <- tail(bitcoin, 150)

bitcoin <- as.xts(bitcoin)

dev.new(width = 20, height = 10)

chartSeries(bitcoin, dn.col = "red", TA="addMACD();addSMA(10)")

library(quantmod)

OHLC <- function(d) {

windows(20,10)

chartSeries(d, dn.col = "red")

}

is.trend <- function(ohlc,i,j){

avg1 = mean(ohlc[(i-25):i,4])

avg2 = mean(ohlc[(j-25):j,4])

if(avg1 >= avg2) return(FALSE)

ohlc <- ohlc[i:j, ]

n <- nrow(ohlc)

candle\_l <- pmin(ohlc[, 1], ohlc[, 4])

valley <- rep(FALSE, n)

for (k in 2:(n - 1))

valley[k] <- ((candle\_l[k-1] >= candle\_l[k]) &

(candle\_l[k+1] >= candle\_l[k]))

z <- candle\_l[valley]

if (all(z == cummax(z))) return(TRUE)

FALSE

}

is.trend.rev <- function(ohlc, i, j) {

if (is.trend(ohlc, i, j) == FALSE) return(FALSE)

last\_candle <- ohlc[j + 1, ]

reverse\_candle <- ohlc[j + 2, ]

ohlc <- ohlc[i:j, ]

if (last\_candle[4] < last\_candle[1]) return(FALSE)

if (last\_candle[4] < max(ohlc[,c(1,4)])) return(FALSE)

if (reverse\_candle[1] < last\_candle[4] |

reverse\_candle[4] >= last\_candle[1]) return(FALSE)

TRUE

}

bitcoin <- read.table("Bitcoin.csv", header = T, sep = ";", row.names = 1)

n <- nrow(bitcoin)

result <- c(0,0)

for (a in 26:726) {

for (b in (a + 3):min(n - 3, a + 100)) {

if (is.trend.rev(bitcoin, a,b) & b - a > 10 )

result <- rbind(result, c(a,b))

if (b == n)

break

}

}

z <- aggregate(result, by = list(result[, 2]), FUN = min)[-1, 2:3]

for (h in 1:nrow(z)) {

OHLC(bitcoin[z[h, 1]:z[h, 2] + 2,])

title(main = z[h, ])

}

#Neural networks

data <- read.csv("Bitcoin.csv", header = TRUE, sep = ",")

data2 <- data[order(as.Date(data$Date, format = "%Y-%m-%d")), ]

price <- data2$Close

HLC <- matrix(c(data2$High, data2$Low, data2$Close),

nrow = length(data2$High))

bitcoin.lr <- diff(log(price))

install.packages("TTR")

library(TTR)

rsi <- RSI(price)

MACD <- MACD(price)

macd <- MACD[, 1]

will <- williamsAD(HLC)

cci <- CCI(HLC)

STOCH <- stoch(HLC)

stochK <- STOCH[, 1]

stochD <- STOCH[, 1]

Input <- matrix(c(rsi[700:939], cci[700:939], macd[700:939],

will[700:939], stochK[700:939], stochD[700:939]), nrow = 240)

Target <- matrix(c(bitcoin.lr[701:940]), nrow = 240)

trainingdata <- cbind(Input, Target)

colnames(trainingdata) <- c("RSI", "CCI", "MACD", "WILL",

"STOCHK", "STOCHD", "Return")

install.packages("caret")

library(caret)

trainIndex <- createDataPartition(bitcoin.lr[701:940],

p = .9, list = FALSE)

bitcoin.train <- trainingdata[trainIndex, ]

bitcoin.test <- trainingdata[-trainIndex, ]

install.packages("nnet")

library(nnet)

best.network <- matrix(c(5, 0.5))

best.rmse <- 1

for (i in 5:15)

for (j in 1:3) {

bitcoin.fit <- nnet(Return ~ RSI + CCI + MACD +

WILL + STOCHK + STOCHD, data = bitcoin.train,

maxit = 1000, size = i, decay = 0.01 \* j,

linout = 1)

bitcoin.predict <- predict(bitcoin.fit, newdata = bitcoin.test)

bitcoin.rmse <- sqrt(mean((bitcoin.predict –

bitcoin.lr[917:940])^2))

if (bitcoin.rmse<best.rmse) {

best.network[1, 1] <- i

best.network[2, 1] <- j

best.rmse <- bitcoin.rmse

}

}

InputTest <- matrix(c(rsi[940:969], cci[940:969],

macd[940:969], will[940:969], stochK[940:969],

stochD[940:969]), nrow = 30)

TargetTest <- matrix(c(bitcoin.lr[941:970]), nrow = 30)

Testdata <- cbind(InputTest,TargetTest)

colnames(Testdata) <- c("RSI", "CCI", "MACD", "WILL",

"STOCHK", "STOCHD", "Return")

bitcoin.fit <- nnet(Return ~ RSI + CCI + MACD + WILL +

STOCHK + STOCHD, data = trainingdata,

maxit = 1000, size = best.network[1, 1],

decay = 0.1 \* best.network[2, 1], linout = 1)

bitcoin.predict1 <- predict(bitcoin.fit, newdata = Testdata)

for (i in 1:20) {

bitcoin.fit <- nnet(Return ~ RSI + CCI + MACD + WILL +

STOCHK + STOCHD, data = trainingdata,

maxit = 1000, size = best.network[1, 1],

decay = 0.1 \* best.network[2, 1], linout = 1)

bitcoin.predict <- predict(bitcoin.fit, newdata = Testdata)

bitcoin.predict1 <- (bitcoin.predict1 + bitcoin.predict) / 2

}

money <- money2 <- matrix(0,31)

money[1,1] <- money2[1,1] <- 100

for (i in 2:31) {

direction1 <- ifelse(bitcoin.predict1[i - 1] < 0, -1, 1)

direction2 <- ifelse(TargetTest[i - 1] < 0, -1, 1)

money[i, 1] <- ifelse((direction1 - direction2) == 0,

money[i-1,1]\*(1+abs(TargetTest[i - 1])),

money[i-1,1]\*(1-abs(TargetTest[i - 1])))

money2[i, 1] <- 100 \* (price[940 + I - 1] / price[940])

}

x <- 1:31

matplot(cbind(money, money2), type = "l", xaxt = "n",

ylab = "", col = c("black", "grey"), lty = 1)

legend("topleft", legend = c("Neural network", "Benchmark"),

pch = 19, col = c("black", "grey"))

axis(1, at = c(1, 10, 20, 30),

lab  = c("2014-04-08", "2014-04-17", "2014-04-27", "2014-05-07"))

box()

mtext(side = 1, "Test dataset", line = 2)

mtext(side = 2, "Investment value", line = 2)

#Logoptimal portfolios

d <- read.table(file.path("data", all\_files[[1]]),

sep = ",", header = FALSE)

colnames(d) = c("date", substr(all\_files[[1]], 1,

nchar(all\_files[[1]]) - 4))

for (i in 2:length(all\_files)) {

d2 <- read.table(file.path("data", all\_files[[i]]),

sep = ",", header = FALSE)

colnames(d2) = c("date", substr(all\_files[[i]], 1,

nchar(all\_files[[i]])-4))

d <- merge(d, d2, sort = FALSE)

}

log\_opt <- function(x, d, r = NA) {

x <- c(x, 1 - sum(x))

n <- ncol(d) - 1

d["distance"] <- c(1, dist(d[2:ncol(d)])[1:(nrow(d) - 1)])

if (is.na(r)) r <- quantile(d$distance, 0.05)

d["similarity"] <- d$distance <= r

d["similarity"] <- c(d[2:nrow(d), "similarity"], 0)

d <- d[d["similarity"] == 1, ]

log\_return <- log(as.matrix(d[, 2:(n + 1)]) %\*% x)

sum(log\_return)

}

log\_optimization <- function(d, r = NA) {

today <- d[1, 1]

m <- ncol(d)

constr\_mtx <- rbind(diag(m - 2), rep(-1, m - 2))

b <- c(rep(0, m - 2), -1)

opt <- constrOptim(rep(1 / (m - 1), m - 2),

function(x) -1 \* log\_opt(x, d), NULL, constr\_mtx, b)

result <- rbind(opt$par)

rownames(result) <- today

result

}

simulation <- function(d) {

a <- Position( function(x) substr(x, 1, 2) == "96", d[, 1])

b <- Position( function(x) substr(x, 1, 2) == "97", d[, 1])

result <- log\_optimization(d[b:a,])

result <- cbind(result, 1 - sum(result))

result <- cbind(result, sum(result \* d[b + 1, 2:6]),

sum(rep(1 / 5, 5) \* d[b + 1, 2:6]))

colnames(result) = c("w1", "w2", "w3", "w4", "w5",

"Total return", "Benchmark")

for (i in 1:2490) {

print(i)

h <- log\_optimization(d[b:a + i, ])

h <- cbind(h, 1 - sum(h))

h <- cbind(h, sum(h \* d[b + 1 + i, 2:6]),

sum(rep(1/5,5) \* d[b + 1 + i, 2:6]))

result <- rbind(result,h)

}

result

}

A <- simulation(d)

matplot(cbind(cumprod(A[, 6]), cumprod(A[, 7])), type = "l",

xaxt = "n", ylab = "", col = c("black","grey"), lty = 1)

legend("topright", pch = 19, col = c("black", "grey"),

legend = c("Logoptimal portfolio", "Benchmark"))

axis(1, at = c(0, 800, 1600, 2400),

lab = c("1997-01-02", "2001-03-03", "2003-05-13", "2006-07-17"))