Can We Apply Traditional Forecasting Models to Predicting Bitcoin?

- Supplementary Document 1 – R Code

Data collection / Data management

#Imports Bitcoin data

(df <- read\_csv ("Crypto Data/bitstampUSD\_1-min\_2012-01-01\_to\_2020-09-14.csv")

>dim(df)

[1] 4572257 8

Data Development and Preparation

#remove scientific notation

options (scipen = 999)

#omits NA's from the data frame

df <- na.omit(df)

#Adjust the time formatting

df$Timestamp <- as.POSIXct(df$Timestamp, origin = "1960-01-01")

#gives columns a proper name

names(df) [1] <- "Date\_time"

names(df) [8] <- "Price"

names(df) [4] <- "Volume"

#Removes time from the date column

df\_raw$Timestamp <- as.Date(df\_raw$Timestamp)

#Creates a aggregate of volume by date

agg<-aggregate(Volume\_BTC~Date, df\_raw, sum)

#merges volume with DF

df<-agg %>%

select(Date, Volume\_BTC) %>%

distinct() %>%

right\_join(df\_raw, by = 'Date')

#removes duplicate line items

df<-df%>%

group\_by(Date) %>%

filter(Close == max(Close))

#Cleans up changes

names(df)[2] <- "Volume\_BTC"

df$Volume\_BTC.y <- NULL

Independent & Dependent Variable Creation

#Adds Lags: -----------

df<-mutate (df, Lag1 = lag (Close))

df<-mutate (df, Lag2 = lag (Close, n=2))

df<-mutate (df, Lag3 = lag (Close, n=3))

df<-mutate (df, Lag4 = lag (Close, n=4))

df<-mutate (df, Lag5 = lag (Close, n=5))

#creates a variable determining growth

df<-mutate (df, Direction = ifelse(df$Close-lag(df$Close)>=0, print ("up"), print ("down")))

#converts the direction variable to factor

df$Direction <- as.factor(df$Direction)

Model 1 – Logistic Regression

#Calculates the coefficients of the logistic regression Model----

fit = glm (Direction ~ Volume\_BTC + Open + High + Low + Close + Lag1 + Lag2 + Lag3 + Lag4 + Lag5, data=df, family=binomial, maxit = 100)

Here the newly created variable to predict the probability (‘up’ or ‘down’) of the target variable “fit”:

#predicts the probability of the Direction variable for the model using the generates coefficients

probability = predict (fit, type="response")

probability [1:20]

mean (probability)

#Confusion Matrix & Accuracy----

#prepares the data

df <- na.omit (df)

pred=rep("up",5978)

pred[probability<=.5] = "down"

#Confusion Matrix

table (pred, df$Direction)

#Accuracy

predfactor<-as.factor (pred)

levels(df$Direction) <- levels (predfactor)

confusionmatrix (predfactor, df$Direction)

Model 2: Logistic Regression with Training / Testing data.

#Model 2: Log Regression with training/testing data----

#Generates training and testing data

train <- df [df$Date <= "2008-01-01",]

test <- df [df$Date > "2008-01-01",]

#sub-setting the training data

fit2=glm (Direction ~ Volume\_BTC+Open+High+Low+Close+Price+Lag1+Lag2+Lag3+Lag4+Lag5, data=df, family=binomial, subset=unlist (train), maxit=100)

#making the prediction

prob2 = predict(fit2,test,type="response")

#View fit

summary(fit)

#check for Heteroscedasticity

bptest(fit)

#Confusion Matrix & Accuracy----

pred2=rep("up",1028)

pred2[prob2<=.5] = "down"

#Confusion matrix

table(pred2,test$Direction)

#Confusion matrix analysis

pred2factor<-as.factor(pred2)

levels(test$Direction) <- levels(pred2factor)

confusionMatrix(pred2factor,test$Direction)

Model 3: Logistic Regression with Moving Average

#Alternative- Moving Average:

BTC<-BTC%>%

mutate (Movavg =

(lag (Close, n=1) +

lag (Close, n=2) +

lag (Close, n=3) +

lag (Close, n=4) +

lag (Close, n=5) +

lag (Close, n=6) +

lag (Close, n=7) +

lag (Close, n=8) +

lag (Close, n=9) +

lag (Close, n=10)) / 10)

Because this code calculates a 10-day moving average, the first 10 rows are populated with ‘n/a’s. The following lines of code removes n/a rows from the model.

#removes the first 10 rows with NAs due to the lag's algorithm

df <- na.omit (BTC)

Autocorrelation Function

Figure 10: Model 4 ACF Plot

#check for autocorrelation

acf (lndata, lag.max = 500,

main="ACF plot - 500 Lags",

col = "#CD0000")

Partial Autocorrelation Function (PACF)

#check for partial autocorrelation

pacf (lndata, lag.max = 200, main="PACF plot")

ARIMA Analysis

#Dickey-Fuller Test for Time series and seasonality

adf.test (lndata)

difmodel = diff (lndata,1)

adf.test (difmodel)

#Auto Arima for Time Series Analysis

#Converts the data to Time Series

priceclose <- ts (lndata, start = 1, frequency = 365)

fitlndata <- auto.arima (priceclose)

#fitlndata

#plots the data in ln format

plot (priceclose, type = 'l',main = 'Bitcoin Price')

#exp (lndata)

#Forecast

forecastln = forecast (fitlndata, h = 180)

plot (forecastln, col = "seagreen")

#Reverts forecast data from exponential

forecast\_extracted = as.numeric (forecastln$mean)

final\_forecast = exp (forecast\_extracted)

final\_forecast #displays the data

#Creates a data frame for comparison

test = data.frame (model2$Close[2001:2582], final\_forecast)

names(test) <- c ("actual", "forecast")

attach(test)

#Export's the data

write.csv(test,'Acutal\_VS\_Forecast.csv')

#Creates a plot to display Actual vs Forecast----

plot (test$actual, type = "b", frame = FALSE, pch = 19,

col = "red", xlab = "Time", ylab = "Price", cex =.1, main = "Actual VS. Forecast")

lines (test$forecasted, pch = 18, col = "blue", type = "b", lty = 2, cex=.5)

legend ("topleft", legend=c("Actual", "Forecast"),

col=c ("red", "blue"), lty = 1:2, cex=0.8)

#calculate percent of error----

percent\_error <- ((test$actual - test$forecast) / test$actual)

mean (percent\_error)

Model 5: ARIMA with Moving Average:

The major changes are highlighted below:

#Moving Average and data prep: ----

model2 <- model2%>%

mutate (Movavg =

(lag (Close, n=1) +

ag (Close, n=2) +

lag (Close, n=3) +

lag (Close, n=4) +

lag (Close, n=5)) /5)

#filter out unnecessary attributes

model2 <- na.omit (model2)

model2 <- model2 %>%

select (Date, Movavg)

#generate test data + convert to Log

lndata = log(model2$Movavg[1:2000])

#check for autocorrelation----

acf (lndata, lag.max = 500,

main = "ACF plot - 500 Lags",

col = "#CD0000")

#autocorrelation coefficients

(acf(lndata, lag.max = 200, main="ACF plot"))

#check for partial autocorrelation

pacf (lndata, lag.max = 200,

main = "PACF plot",

col = "#CD0000")

# Augmented Dickey-Fuller Test for Time series and seasonality

Difmodel = diff (lndata,1)

adf.test (lndata)

adf.test (difmodel)

Forecasting: Below, the time series(ts) function frequency is set to 52. This is because each observation represents five days’ worth of data.

#Converts the data to time series format

priceclose <- ts(lndata, start = 1, frequency = 52)

#start-c(2013,10)

#fitlndata

fitlndata <- auto.arima (priceclose)

#plots the data in ln format

plot (priceclose, type = 'l', main='Bitcoin Price')

#exp(lndata)

#Forecast

forecastln = forecast (fitlndata, h=572)

plot (forecastln, col = "seagreen")

#Reverts forecast data from exponential

forecast\_extracted = as.numeric (forecastln$mean)

final\_forecast = exp(forecast\_extracted)

final\_forecast

#Creates a data frame for comparison

test = data.frame (model2$Movavg[2001:2572],final\_forecast)

names(test) <- c("actual", "forecast")

attach(test)

#export's the data

write.csv(test,'Acutal\_VS\_Forecast.csv')

#Creates a plot to display Actual vs Forecast----

plot (test$actual, type = "b", frame = FALSE, pch = 19,

col = "red", xlab = "Time", ylab = "Price", cex =.1, main = "Actual VS. Forecast")

lines (test$forecasted, pch = 18, col = "blue", type = "b", lty = 2, cex=.5)

legend ("topleft", legend=c("Actual", "Forecast"),

col=c ("red", "blue"), lty = 1:2, cex=0.8)