# Crypto Currencies - HarvardX Capstone Report

Thierry Morvany

30 December, 2021

## Contents

# 1 Introduction

The following data science project is my final capstone as part of the HarvardX's Data Science Professional Certificate series. The code for this project can be found on GitHub repository.
Lately, the crypto currencies grab headlines when their value massively varied following bank announcements[1] or Elon Musk's[2]. Then, the dataset of historic crypto currencies available on Kaggle website seems to promise some relatable insights. Following an Exploration Data Analisys, I will seek for the best algorithms to forecast a couple of features available in the dataset.

Please, note that my 16GB-ram laptop was not powerfull enough all along. I then had to create an instance -of type r5.4xlarge on an Amazon Web Services (AWS) account to run the code built with R version 4.0.2. The creation procedure is described here.

# 2 Exploration Data Analysis

## 2.1 Libraries

Diverse tools are necessary to run the project, thus the libraries below will be used:

```r
#--- Exploration Data Analysis
library(tidyverse)    # Organization and visualization data
library(caret)        # Machine learning procedure
library(data.table)   # For table manipulation
library(kableExtra)   # For table presentation
library(corrplot)     # For correlation visualization
library(ggrepel)      # Advanced geometric objects managenment
library(ggpubr)       # Complent to ggplot2
library(lubridate)    # For date management
#--- Forecast
library(rpart)        # Recursive Partitioning and Regression Trees
library(randomForest) # Random Forests for Classification and Regression
library(Rborist)      # Parallel Implementation of the Random Forest Algorithm
library(pls)          # Partial Least Squares and Principal Component Regression
```

## 2.2 Dataset

I downloaded the 23 files from Kaggle website and merge them in one. Below is an extract of the dataset that shows the variables:

Table 1: Data Preview

| SNo | Name | Symbol | Date | High | Low | Open | Close | Volume | Marketcap | Year |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Aave | AAVE | 2020-10-05 | 55.11 | 49.79 | 52.67 | 53.22 | 0 | 89,128,129 | 2,020 |
| 2 | Aave | AAVE | 2020-10-06 | 53.40 | 40.73 | 53.29 | 42.40 | 583,091 | 71,011,441 | 2,020 |
| 3 | Aave | AAVE | 2020-10-07 | 42.41 | 35.97 | 42.40 | 40.08 | 682,834 | 67,130,037 | 2,020 |
| 4 | Aave | AAVE | 2020-10-08 | 44.90 | 36.70 | 39.88 | 43.76 | 1,658,817 | 220,265,142 | 2,020 |
| 5 | Aave | AAVE | 2020-10-09 | 47.57 | 43.29 | 43.76 | 46.82 | 815,538 | 235,632,208 | 2,020 |

This dataset has the historical price information of some of the top crypto currencies by market capitalization where the variables are described below:

- Date : date of observation
- Open : Opening price on the given day
- High : Highest price on the given day
- Low : Lowest price on the given day
- Close : Closing price on the given day
- Volume : Volume of transactions on the given day
- Market Cap : Market capitalization in USD

The observations in the dataset are for 23 crypto currencies:

Table 2: Currencies

| | | | |
|---|---|---|---|
| Aave | Crypto.com Coin | Monero | TRON |
| Binance Coin | Dogecoin | NEM | Uniswap |
| Bitcoin | EOS | Polkadot | USD Coin |
| Cardano | Ethereum | Solana | Wrapped Bitcoin |
| Chainlink | IOTA | Stellar | XRP |
| Cosmos | Litecoin | Tether | |

The file make available different intervals of history for the currencies:

Table 3: Historic intervals

| Name | Symbol | From | To | Days of historic |
|---|---|---|---|---:|
| Bitcoin | BTC | 2013-04-29 | 2021-07-06 | 2990 |
| Litecoin | LTC | 2013-04-29 | 2021-07-06 | 2990 |
| XRP | XRP | 2013-08-05 | 2021-07-06 | 2892 |
| Dogecoin | DOGE | 2013-12-16 | 2021-07-06 | 2759 |
| Monero | XMR | 2014-05-22 | 2021-07-06 | 2602 |
| Stellar | XLM | 2014-08-06 | 2021-07-06 | 2526 |
| Tether | USDT | 2015-02-26 | 2021-07-06 | 2322 |
| NEM | XEM | 2015-04-02 | 2021-07-06 | 2287 |
| Ethereum | ETH | 2015-08-08 | 2021-07-06 | 2159 |
| IOTA | MIOTA | 2017-06-14 | 2021-07-06 | 1483 |
| EOS | EOS | 2017-07-02 | 2021-07-06 | 1465 |
| Binance Coin | BNB | 2017-07-26 | 2021-07-06 | 1441 |
| TRON | TRX | 2017-09-14 | 2021-07-06 | 1391 |
| Chainlink | LINK | 2017-09-21 | 2021-07-06 | 1384 |
| Cardano | ADA | 2017-10-02 | 2021-07-06 | 1373 |
| USD Coin | USDC | 2018-10-09 | 2021-07-06 | 1001 |
| Crypto.com Coin | CRO | 2018-12-15 | 2021-07-06 | 934 |
| Wrapped Bitcoin | WBTC | 2019-01-31 | 2021-07-06 | 887 |
| Cosmos | ATOM | 2019-03-15 | 2021-07-06 | 844 |
| Solana | SOL | 2020-04-11 | 2021-07-06 | 451 |
| Polkadot | DOT | 2020-08-21 | 2021-07-06 | 319 |
| Uniswap | UNI | 2020-09-18 | 2021-07-06 | 291 |
| Aave | AAVE | 2020-10-05 | 2021-07-06 | 274 |

## 2.3 Exploration Data Analysis

### 2.3.1 Variables correlations

An overview of the correlation plot shows a multicollinearity between the price-related variables High, Low, Open and Close. These variables are less correlated to Volume and Marketcap.

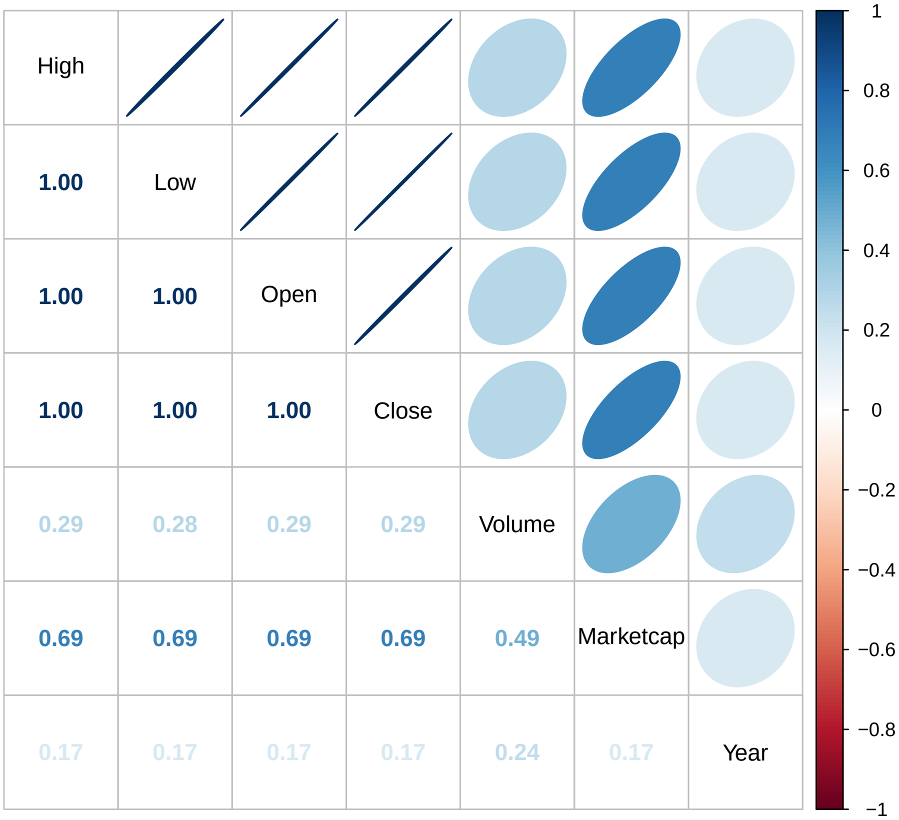
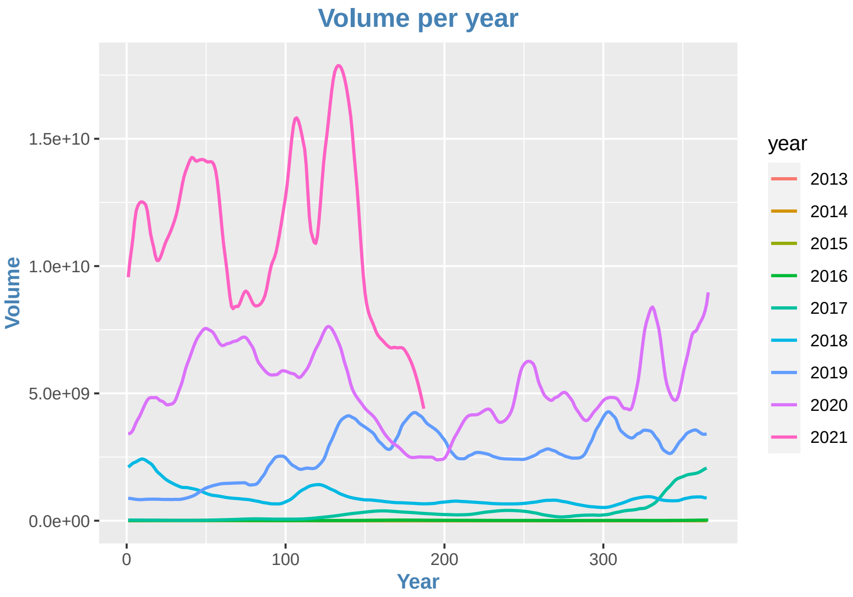


Figure 1: Variables correlations

### 2.3.2 Variables over time



.

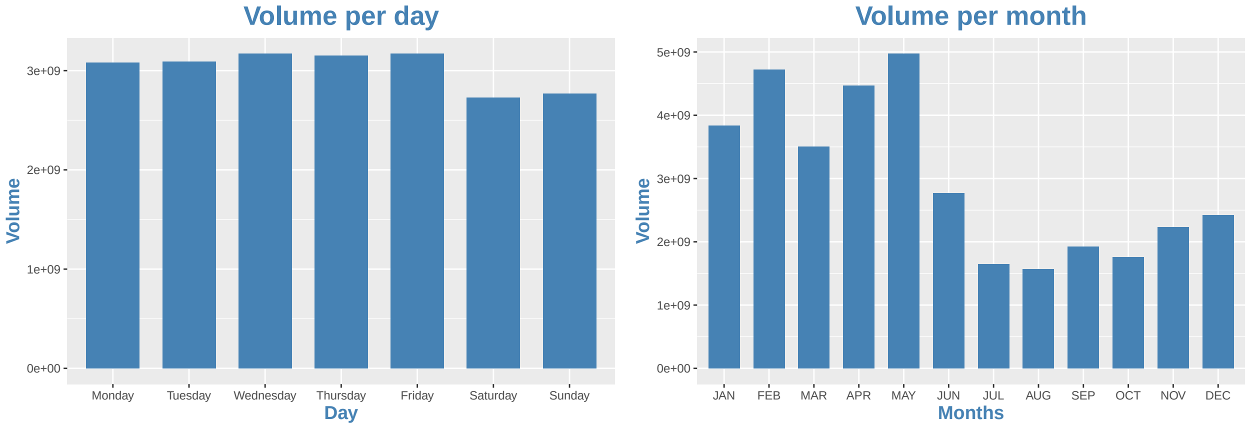


Figure 2: Volume Over Time

Observing fig.2, it appears clearly a higher volume since 2019, a slightly lower volume over the weekend and a larger volume over the first semester:

### 2.3.3 Variable over currencies

I intend to see which currencies had more impact in the market:

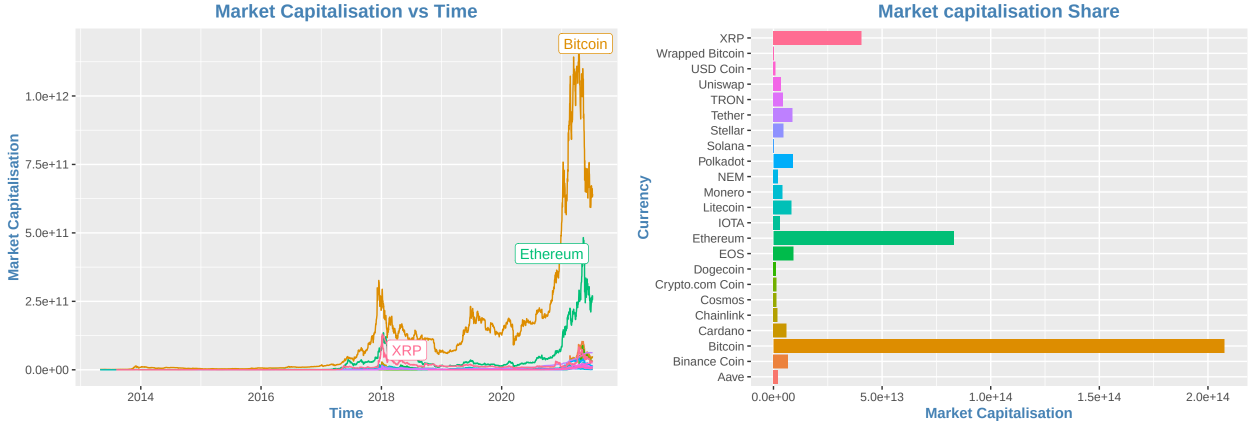


Figure 3: Importance of currencies in market

Bitcoin, Ethereum, and XRP influence the most the market capitalization. Bitcoin is by far the most influent making it one of the most widely accepted and accessible cryptocurrencies on the market.

Bitcoin was invented in 2009, and is the world's oldest and best-known cryptocurrency. It is a digital currency which operates free of any central control or the oversight of banks or governments. Instead it relies on peer-to-peer software and cryptography.

Ethereum is one of the most popular cryptocurrencies in the world and a viable contender for the Bitcoin throne. Ethereum takes much of what Bitcoin offers and improves upon it with faster transactions, smart contracts and native apps.

In the world of cryptocurrencies, XRP is quite different. It is decentralized, but not as much as Bitcoin, Ethereum, and the likes. Unlike them, it doesn't seek to be an alternative to the traditional banking system but rather to collaborate and improve the conventional banking system. Nonetheless, it is a cryptocurrency.

### 2.3.4 Attractiveness and outliers

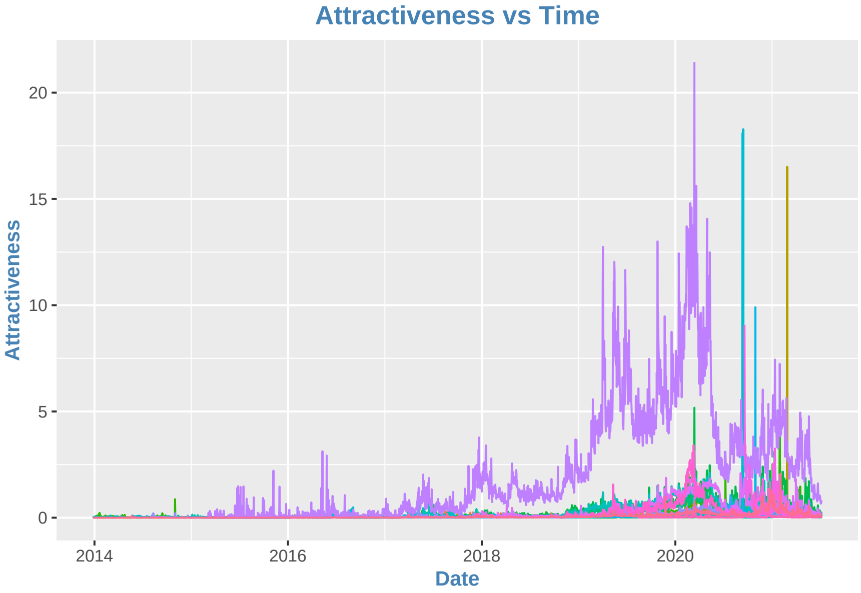
The article *The Confusions between Capitalization and Trading Volume*[3] mentions that

*'You will have a better overview of the market and the coin itself when determining some information as well as the close connection between Capitalization and Trading Volume.'*

To mitigate this confusion and get a better overview I made a connection between Capitalization and Trading Volume by defining the currency attractiveness ratio:

$$Attractiveness = Volume/Marketccap$$

The evolution of the attractiveness over time:



Despite the variations of attractiveness are somehow limited, we can observe some outliers. Outliers are high points that will result in over reactions in a forecast computation. Therefore, spot outliers and smooth them out will allow a better forecast. I intend to implement the automation of their detection and cleaning as described in *Data science for supply chain forecast*[4]. As algorithm uses the standard deviation around the historical average it must be applied to a normal distribution. So, I will check that the errors to the mean of attractiveness is distributed normally (See below fig. 5)

For the ability to apply the algorithm, I had to apply the logarithm transformation to the feature attractiveness.
For the clarity of the graphs, I filtered the historical data of Bitcoin over the years 2014 to 2017:

```
LogAttract <- cryptos %>%
# Subset data: Bitcoin history for 2014 and 2015
   filter(Symbol== "BTC" & Year %in% c("2014":"2017") ) %>%
   mutate(Value= Volume/Marketcap) %>%
  .$Value %>%
  log()         # Log transformation
# Histogram
hist(LogAttract)
```

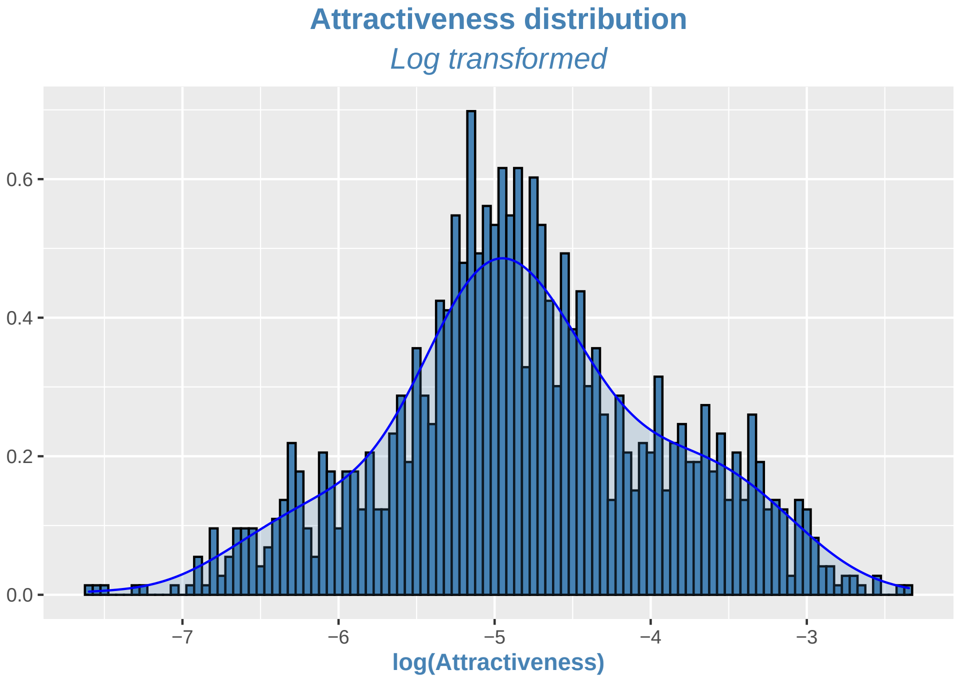


Figure 4: Log-transformed attractiveness distribution

The distribution of the attractiveness is approximately normal.That is quite a good sign as the approach is to look at the atttractiveness standard deviation around the historical average and exclude the values that are exceptionally far from the average. Let's define the attractivess standard deviation:

$$\sigma = \sqrt{\frac{1}{n}\sum \ (Attractiveness - Average)^2}$$

To check the distribution around the average (i.e the error) is normally distributed (See fig. 5 below), let's go through the steps of the algorithm:

- **Start**: Subset the historical data and log-transformed the variable attractiveness:

$$LogAttract = log(Attractiveness)$$

  Then, apply the algorithm to the transformed variable

- **Step 1**: Calculate a naïve forecast that equals the average of the attractiveness by day of the year
- **Step 2**: Compute the error:

$$error = AttractLogScaled - mean(AttractLogScaled)$$

  then the average and standard deviation of the error.

```
#- Data subset
crypto_attract <- cryptos %>%
  filter(Symbol== "BTC" &           # Bitcoin
         Year  %in% c(2014:2017) &  # 2014:2017
         Marketcap !=0  & Volume != 0) %>%
  mutate(Day= yday(Date),
         LogAttract= log(Volume/Marketcap)) # log(Attractiveness)

#- Step 1 - Populate a naive forecast as an average of the historical attractiveness
#           by day of the year
crypto_avgattract <-  crypto_attract %>%
                      group_by(Day) %>%
                      summarise(AvgLogAttract= mean(LogAttract)) %>%
                      mutate(Year= "Avg")

#- Step 2 - Compute the error, then the mean and standard deviation of the error
error <- crypto_attract %>%
  right_join(crypto_avgattract, by= "Day") %>%
  mutate(error=  LogAttract - AvgLogAttract) %>%
  .$error

m1 <- mean(error)      # Average
s1 <- sd(error)        # Standard deviation
```

Checking out the QQ-plot below it becomes reasonable to consider the error is normally distributed, that means the historical attractiveness is actually normally distributed around its average.

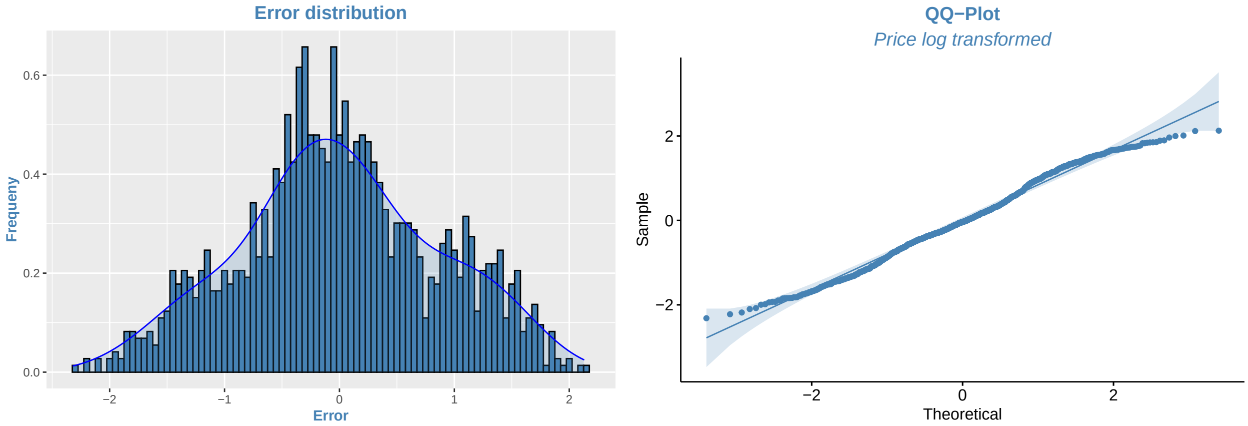
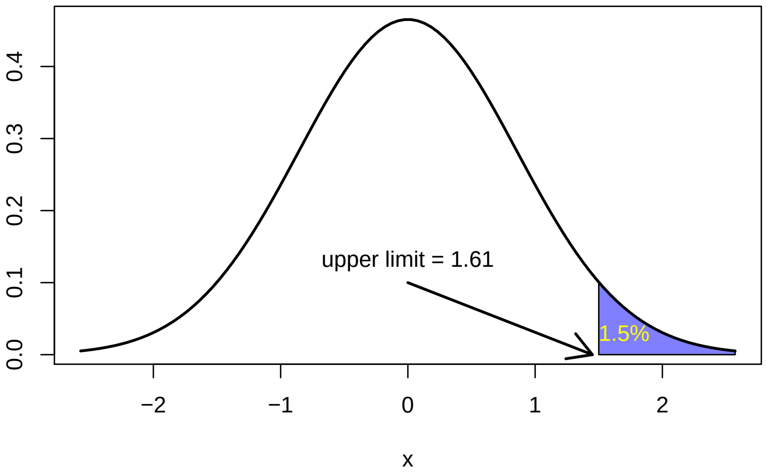


Figure 5: Error distribution

Therefore, we can use the standard deviation and compute the probability for the attractiveness to be below an upper limit.



- **Step 3**: Compute the upper acceptable limit from the error mean and standard deviation.

- **Step 4**: Identify the outliers. Note that the upper limit just calculated in step 3 is based on parameters (mean & sigma) that uses the outliers themselves. These outliers increase the error variation so that the upper limit is biased and over estimated. To adjust this, one could exclude the outliers just identified to recompute the error distribution mean and standard deviation, then the upper limit *Note I do not correct far low values. Since the data is log transformed low outliers are actually next to null values in the original set and therefore do not request to be adjusted.*

- **Step 5**: Recompute the error distribution average and standard deviation without the outliers we just detected.

- **Step 6**: Update the upper limit based on the new values of the average and the standard deviation

- **Step 7**: Update the outliers based on the new upper limit and set their values to upper limit.

```r
# Step 3  - Set upper limit that implies the error has 1.5% chance to be higher
prob <- 0.97
quant1 <- qnorm(prob, m1, s1)                              # Quantile
upper_limit <- crypto_avgattract$AvgLogAttract + quant1    # Upper limit = Average + quantile(0.97)

#- Step 4 -  Indentify outliers i.e for which error is greater than the upper limit
ind_outliers <- which(error > quant1)

#- Step 5 - Recompute error distribution parameters excluding outliers
m2 <- mean(error[-ind_outliers])      # Average excluding outliers
s2 <- sd(error[-ind_outliers])        # Standard deviation excluding outliers

#- Step 6 - Update the upper limit with the new average and standard deviation
quant2 <- qnorm(prob, m2, s2)                          # Quantile
upper_limit <- crypto_avgattract$AvgLogAttract + quant2    # Upper limit = Average + quantile(0.97)

#- Step 7 - Update the outliers based on the new upper limit
ind_outliers <- which(error > quant2)
crypto_updattract <- crypto_attract
crypto_updattract$LogAttract[ind_outliers] <- upper_limit
```

Below is the result of the algorithm that shows outliers of bitcoin historical data over year 2014 to 2017

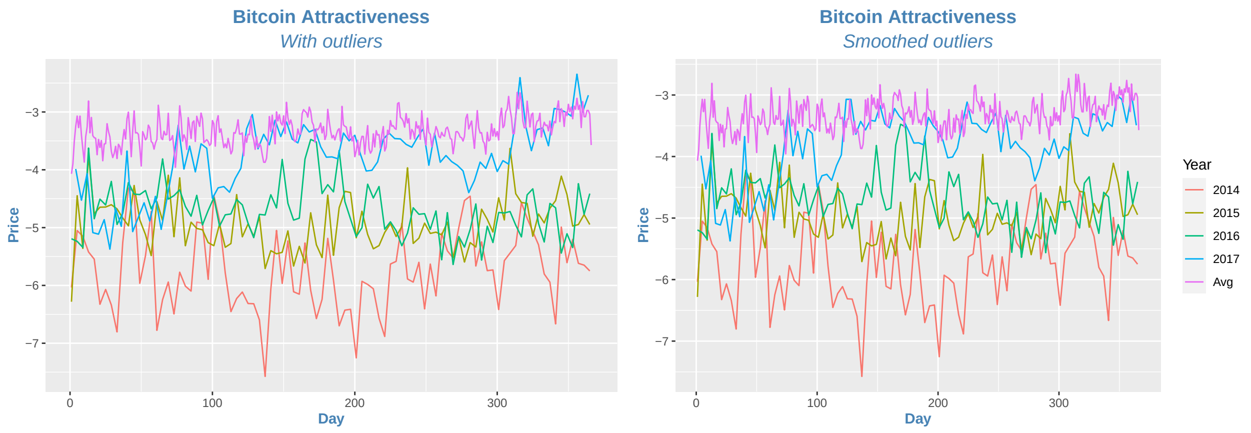


Figure 6: Loged Attractiveness distribution

Then, are noticeable the outliers in 2017. Transforming the data back to its original scale is done by applying the exponential to LogAttract:

$$Attractiveness = exp(log(Attractiveness)) = exp(LogAttract)$$

The following graph compares the attractivemes with vs witout outliers. We show the results on the subset to 2014 to 2017

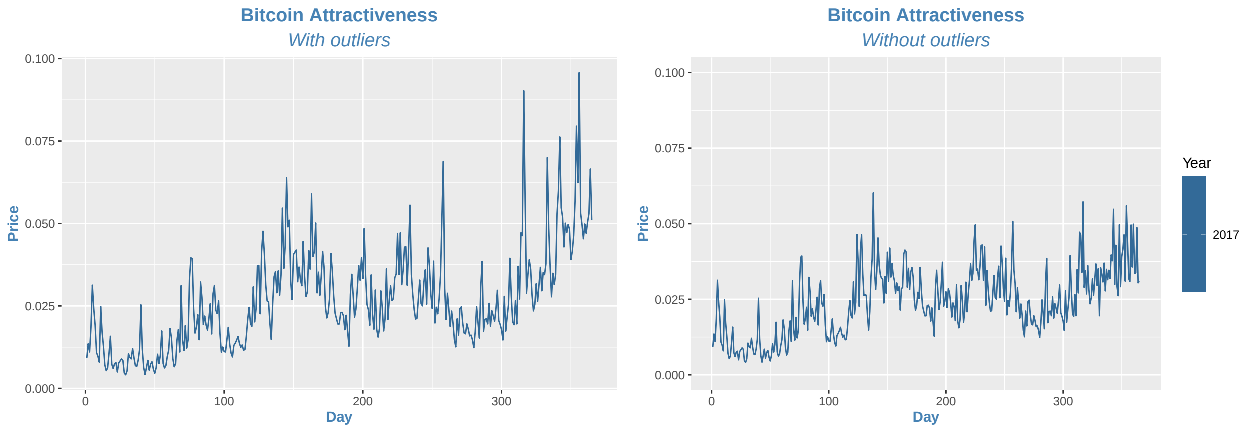


Figure 7: Outliers management

We apply the algorithm to smooth the outliers out to the result below:

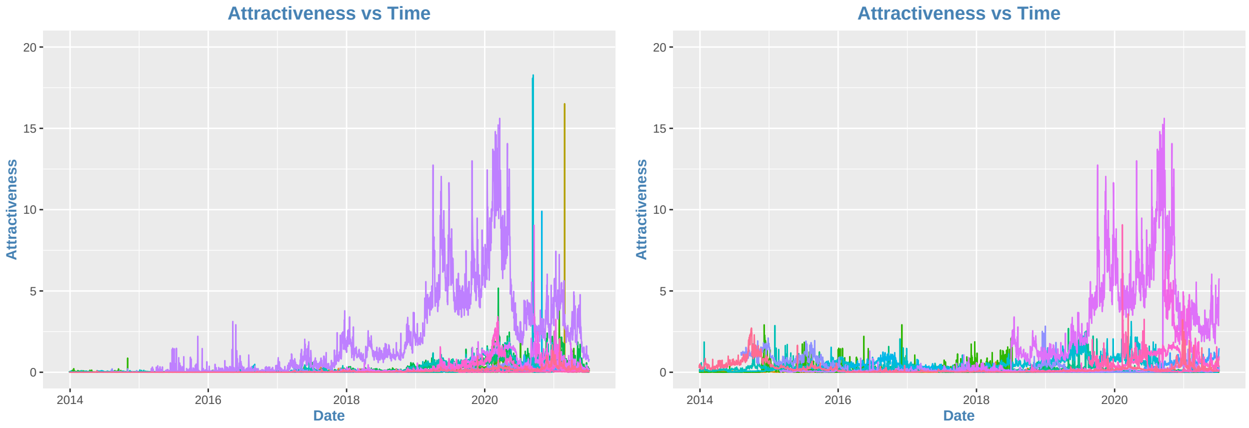


Figure 8: Outliers management - Full data

The outliers has been smoothed out so that the forecast can be computed without the variation they might introduced.

# 3 Forecast

Traditional statistical models use a predefined relationship (model) and apply it to populate a forecast (attractiveness or price) based on historical data. Whereas a machine learning algorithm will not assume *à priori* a particular relationship like seasonality or a linear trend. It will *learn* these patterns from the historical data. As a first machine learning algorithm, we will use a decision tree. However, none of this means that we can't build useful algorithms that are much better than guessing, and in some cases better than expert opinions.

## 3.1 Forecast error measure

In Hyndman's article[5], we can read that *'historically, the RMSE and MSE have been popular, largely because of their theoretical relevance in statistical modeling.'* Conveniently, we also detected and smoothed out the outliers so that the RMSE and MSE sensitivity to outliers will shrunk. The same article also concludes that MASE is widely applicable in our current configuration, that yet requires a Naive forecast.

**RMSE** (Root Mean Suare Error) is calculated as below:

$$RMSE = \sqrt{\frac{1}{N}\sum (Predicted - True)^2}$$

**MASE** (Mean Absolute Scaled Error) is calculated from MAE (Mean Absolute Error)

$$MAE = \frac{1}{N}\sum |Predicted - True|$$

scaled to the Naive Forecast Mean Absolute Error:

$$MASE = \frac{MAE}{MAE_{Naive}}$$

```r
#--- RMSE
RMSE <- function(true, predicted){
  sqrt(mean((true - predicted)^2))
}
#--- MAE
MAE <- function(true, predicted){
  sum(abs(true-predicted))/length(true)
}
#--- MASE
MASE <- function(mae_model, mae_naive){
  mae_model/mae_naive
}
```

Mathematical optimisation shows that a good forecast of the mean will generate a good RMSE and a good forecast of the median will output a small MAE (and MASE). The approach below will prove this by seeking the minimum of the mathematical function when their derivartive is set to zero:

**RMSE**

$$\frac{\partial RMSE}{\partial pred} = \frac{1}{2}\frac{\partial \frac{1}{n}\sum(pred_t - true_t)^2}{\partial pred}$$

$$\frac{1}{n}\sum(pred_t - true_t) = 0$$

$$\sum pred_t = \sum true_t$$

Therefore, to optimise a forecast RMSE, the model means that the total predicted is equal to the total true values. To say it an other way, optimising RMSE aims to produce a prediction that is correct *on average* and so, unbiased.

**MASE**

$$\frac{\partial MASE}{\partial pred} = 0$$

is equivalent to

$$\frac{1}{Naive} * \frac{\partial \frac{1}{n}\sum(pred_t - true_t)}{\partial pred} = 0$$

When aknowledging the following

$$\frac{\partial |pred_t - true_t|}{\partial pred} = 1\ if\ true_t < pred_t$$

and

$$\frac{\partial |pred_t - true_t|}{\partial pred} = -1\ if\ true_t > pred_t$$

the derivative of MAE becomes

$$\frac{\partial MAE}{\partial pred} = \frac{1}{n}\sum 1\ if\ true_t < pred_t\ or\ -1\ if\ true_t > pred_t$$

Therefore, optimising MAE (i.e derivative = 0), the predicted values need to be as many times higher than the true values as it is lower than the true values. That means the data set is split into to equal parts, which is the definition of the median.

## 3.2 Data preparation

The variable *Attractiveness* was computed in Attractiveness and outliers and will be predicted. Therefore, the feature is added to the data set.
Also, some currencies has not much historical data (See table **??**tab:Historic intervals)). To run a machine able to learn and mitigate the variablity these currencies would introduce, the ones with less than 2 years of historical data are filtered out:

```
cryptos_feat <- cryptos %>%
  bind_cols(Attractiveness= unlist(AttractSmooth)) %>% # Outliers smoothed out
  filter(Duration > 800)  # More than 800 days (~2 years) of historical data
```

.

We have to convert the time series data to a machine learning one by creating features from the time variable. In the code below, I create time features like day of the year, month, quarter, and year.

```
cryptos_feat <- cryptos_feat %>%
  mutate(Day     = as.factor(yday(Date)),
         Month   = as.factor(month(Date)),
         Quarter = as.factor(quarter(Date)),
         Year    = as.factor(year(Date)))
```

Fig. 2 that hilighted a lower traded volume during the weekends and the second semester suggests some specific time features. consequently, the features weekend and semester will help fitting the data when using machine learning:

```
cryptos_feat <- cryptos_feat %>%
  mutate(Weekdays = as.factor(weekdays(Date)),
         Weekend  = as.factor(if_else(Weekdays %in% c("Saturday", "Sunday") , 1, 0)),
         Semester = as.factor(if_else(Quarter %in% c("1", "2"),  1, 2))) %>%
  select(Symbol, Name, Low, Volume, Attractiveness, Volume, Day,
         Weekend, Month, Semester, Year, Date)
```

## 3.3 Machine learning

In order to make a forecast, the machine learning algorithm should answer the following question:
*Based on the variable over the last n periods, what will the variable (price or attractiveness) be during the next period(s)?*

To answer this question:

- The algorithm is fed with the dataset prepared in chapter Data preparation and that contains the currencies with more than 2 years of historical data.
- With no préjugés, some machine learning algorithms from different family will be tested:
  - linear from the regression family as the naïve model
  - Kernel Nearest Neighbors (KNN) as clustering tool
  - Tree regression and random forest from comonly used for classification
  - Principal Component Analysis (PCA) from dimensinality reduction family

The data partition aimed to forecast a period of 6 months. That's defined a reasonable period of time to *learn* about a potential seasonality for the currencies with more than 2 years of historical data:

```r
set.seed(1997, sample.kind="Rounding")

cryptos_feat <- cryptos_feat %>% filter(Year== 2013:2021)
# Validation partition
feat_set  <- cryptos_feat %>% filter(Date <  "2021-01-01")
valid_set <- cryptos_feat %>% filter(Date >= "2021-01-01")

# Test partition
train_set <- feat_set %>% filter(Date <  "2020-06-01")
test_set  <- feat_set %>% filter(Date >= "2020-06-01")
```

## 3.4 Forecast

The usage of the package *caret* allows similar codings for the different algorithms.

### 3.4.1 Naive forecast

The naive forecast will be the linear regression:

```r
# Train a linear model to predict "Low" price
fit_low_lm <- lm(Low ~ Volume+Day+Weekend+Month+Semester+Year,
                 data = train_set)

# Train a linear model to predict Attractiveness
fit_attract_lm <- lm(Attractiveness ~ Low+Volume+Day+Weekend+Month+Semester+Year,
                     data = train_set)
```

### 3.4.2 KNN

```r
grid <- data.frame(k = seq(5, 25, 5))
# Train a knn model to predict "Low" price
fit_low_knn <- train(Low ~ Volume+Day+Weekend+Month+Semester+Year,
                 data     = train_set,
                 method   = "knn",
                 tuneGrid = grid)

# Train a knn model to predict Attractiveness
fit_attract_knn <- train(Attractiveness ~ Low+Day+Weekend+Month+Semester+Year,
                     data     = train_set,
                     method   = "knn",
                     tuneGrid = grid)
```

### 3.4.3 Regression tree

```r
grid <- data.frame(cp = seq(0, 0.1, 0.01))
# Train a tree model to predict "Low" price
fit_low_rpart <- train(Low ~ Volume+Day+Weekend+Month+Semester+Year,
                     data     = train_set,
                     method   = "rpart",
                     tuneGrid = grid)
# Train a knn model to predict attractiveness
fit_attract_rpart <- train(Attractiveness ~ Low+Day+Weekend+Month+Semester+Year,
                     data     = train_set,
                     method   = "rpart",
                     tuneGrid = grid)
```

### 3.4.4 Random forest

```r
# Train a random forest model to predict "Low" price
fit_low_Rborist <- train(Low ~ Volume+Day+Weekend+Month+Semester+Year,
                method   = "Rborist",
                tuneGrid = data.frame(predFixed = seq(33,45,2),
                                      minNode   = seq(33, 45, 2)),
                data     = train_set)
# Train a random forest model to predict attractiveness
fit_Attract_Rborist <- train(Attractiveness ~ Low+Day+Weekend+Month+Semester+Year,
                method   = "Rborist",
                tuneGrid = data.frame(predFixed = seq(33,45,2),
                                      minNode   = seq(33, 45, 2)),
                data     = train_set)
```

### 3.4.5 Principal Components Analysis - PCA

```r
# Train a PCA model to predict Low price
rctrl <- trainControl(method = "cv", number= 5, returnResamp = "all")
fit_low_pca  <- train(Low ~ Volume+Day+Weekend+Month+Semester+Year,
                method= "pcr",
                data= train_set,
                trControl= rctrl,
                preProc= c("center", "scale"),
                tuneGrid= data.frame(ncomp= seq(1,10))
                )
summary(fit_low_pca)
# Train a PCA  model to predict Attractiveness
fit_attract_pca  <- train(Attractiveness ~ Low+Day+Weekend+Month+Semester+Year,
                    method= "pcr",
                    data= train_set,
                    trControl= rctrl,
                    preProc= c("center", "scale"),
                    tuneGrid= data.frame(ncomp= seq(1,10))
                    )
summary(fit_attract_pca)
```

# 4 Results

## 4.1 Global

Table 4: Price forecast error

| Model | Description | RMSE | MAE | MASE |
|---|---|---|---|---|
| MODEL 0 | Naïve: Linear Regression | 4414 | 12095 | 1.0000 |
| MODEL 1 | knn | 3986 | 1688 | 0.1395 |
| MODEL 2 | Tree | 3917 | 1525 | 0.1261 |
| MODEL 3 | Random forest | 3942 | 1495 | 0.1236 |
| MODEL 5 | PCA | 4283 | 1659 | 0.1372 |

Table 5: Attractiveness forecast error

| Model | Description | RMSE | MAE | MASE |
|---|---|---|---|---|
| MODEL 0 | Naïve: Linear Regression | 1.765 | 4.0158 | 1.0000 |
| MODEL 1 | knn | 1.760 | 0.6246 | 0.1555 |
| MODEL 2 | Tree | 1.079 | 0.3866 | 0.0963 |
| MODEL 3 | Random forest | 1.608 | 0.5632 | 0.1402 |
| MODEL 5 | PCA | 1.845 | 0.6235 | 0.1553 |

## 4.2 Case of XRP

We additionally focused on XRP because:

- It was one of the first crypto currencies in use as we can see in Dataset-table *Historic intervals*.
- It is one the tree that has the biggestinfluence in the market (See fig. 5)
- As I mentioned in chapter Data preparation, it also does not seek to be an alternative to the traditional banking system but rather to collaborate and improve the conventional banking system.

When predicting Price, it appears Random forest is the best model:

Table 6: Price forecast error

| Currency | Model | Description | RMSE | MAE | MASE |
|---|---|---|---|---|---|
| **XRP** | MODEL 0 | Naïve: Linear Regression | 1.6038 | 0.7728 | 1.0000 |
| **XRP** | MODEL 1 | KNN | 0.1685 | 0.1473 | 0.1906 |
| **XRP** | MODEL 2 | Regression tree | 0.1551 | 0.1068 | 0.1382 |
| **XRP** | **MODEL 3** | **Random forest** | **0.1212** | **0.0687** | **0.0889** |
| **XRP** | MODEL 4 | PCA | 0.2573 | 0.1642 | 0.2124 |

I used *varImp* that extracts variable importance random forest model. It appears below that Year 2018, Thursdays and Attractiveness accounts the most importnat variables.

Table 7: Important variables for price

| | Importance |
|---|---|
| Volume | 100.000 |
| Year2018 | 67.456 |
| Day4 | 31.851 |
| Attractiveness | 11.917 |
| Year2016 | 6.693 |

When predicting Attractiveness, it appears Kernel Nearest Neighbors (KNN) is the best model:

Table 8: Attracivenees forecast error

| Currency | Model | Description | RMSE | MAE | MASE |
|---|---|---|---|---|---|
| XRP | MODEL 0 | Naïve: Linear Regression | 0.4127 | 2.6358 | 1.0000 |
| **XRP** | **MODEL 1** | **KNN** | **0.2407** | **0.1158** | **0.0439** |
| XRP | MODEL 2 | Regression tree | 0.2437 | 0.1589 | 0.0603 |
| XRP | MODEL 3 | Random forest | 0.2492 | 0.1315 | 0.0499 |
| XRP | MODEL 4 | PCA | 0.3646 | 0.2092 | 0.0794 |

# 5 Conclusion

While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data. It could be considered beyond the content of this report but classical Decomposition, exponential approach and other classical algorithm could be applied to the data and their result compared to the machine learning models. Many machine learning models are available and deep learning or self-learning algorithm may worth being tested. On purpose, I ignored exogenous factors but it is well known that financial and currencies markets are quite sensitive to external events (Ref. Elon Musk's tweets, Warren buffet's articles, central bank's reports, Covid crisis, countries' GDP etc). These external events would explain the outliers and could be modelised in the dataset as binary variables to describe the presence or the absence of an influencable external event.
Despite these missing analysis that may shrink the forecast error, let's keep in mind that the value of AI lies not only in improving the forecast but rather in making better decisions than what humans alone can do.
Additionally, the next step -started but not presentable would be to actually use our model to predict what are the expected future values for the price and the attractiveness oof the cryptomoney XRP.

As a conclusion of the conclusion, I would highly recommend the course to anyone interested in data science and thank the staff for their very prompt replies to my questions but also HarvardX and the instructor Rafael Irizarry for the quality of his course.

# References

[1] R. Shevlin, “Which uk banks will let me buy cryptocurrencies? (2021)”, (2021).

[2] R. Shevlin, “How elon musk moves the price of bitcoin with his twitter activity”, Forbes (2021).

[3] P. News, “Confusions between capitalisation and trading volume”, (2019).

[4] N. Vandeput, *Data science for supply chain forecast* (2018), pp. 87–98.

[5] R. J. Hyndman and A. B. Koehler, “Another look at measures of forecast accuracy”, International Journal of Forecasting **22**, 679–688 (2006).