Bitcoin: Predicting Market Cycle Tops Using the 20-Week Simple Moving Average

Pablo Marchesi

November 21, 2023

Abstract

This project aims to capture and model the recurring patterns observed in the periodic peaks of Bitcoin's bull markets, driven by the inherent cyclicality resulting from the halving events. The methodology involves computing a trading indicator derived from the logarithmic extension of Bitcoin's price in relation to its 20-week Simple Moving Average (SMA).

1 Introduction

The Bitcoin halving is an event that occurs approximately every four years, or after every 210,000 blocks are mined on the Bitcoin blockchain. It is a built-in mechanism designed to control the supply of new bitcoins and, as a result, influences the overall scarcity of the cryptocurrency.

The supply is cut in half with each halving, and this has led historically to several phases of price appreciation known as Bitcoin's bull markets. The periodic nature of the halving creates recurrent patterns in the price of BTC, thus bull markets can be modeled using a quantitative approach.

In this paper, we will show how the peaks of Bitcoin's bull markets can be predicted by applying exponential regression to the logarithmic extension of the price of BTC from its 20-week SMA. The intuition behind this indicator is that bull markets can be modeled as deviations from the mean price of BTC, thus the points of maximum extension (or deviation from the mean) mark the market cycle tops. As these phases of irrational exuberance are periodic (every four years) these peaks can be modeled using regression.

2 Selected Dataset

The historical data used for this project is a combination of a dataset from Kaggle, which consists of the price of BTC from January 2012 to March 2021 (in CSV format) and data from Yahoo Finance (imported with yfinance) which completes the dataset from Kaggle with the most recent values of BTC price. Thus, the timeframe of the study is from January 2012 to November 2023.

The data is in weekly format as the indicator is based on the 20 week SMA. In addition, and for simplicity, only the closing price has been considered for the study. This choice is explained later in the limitations and improvements section.

3 Modelling Bitcoin Tops

We will begin with the closing price and the 20 week SMA:

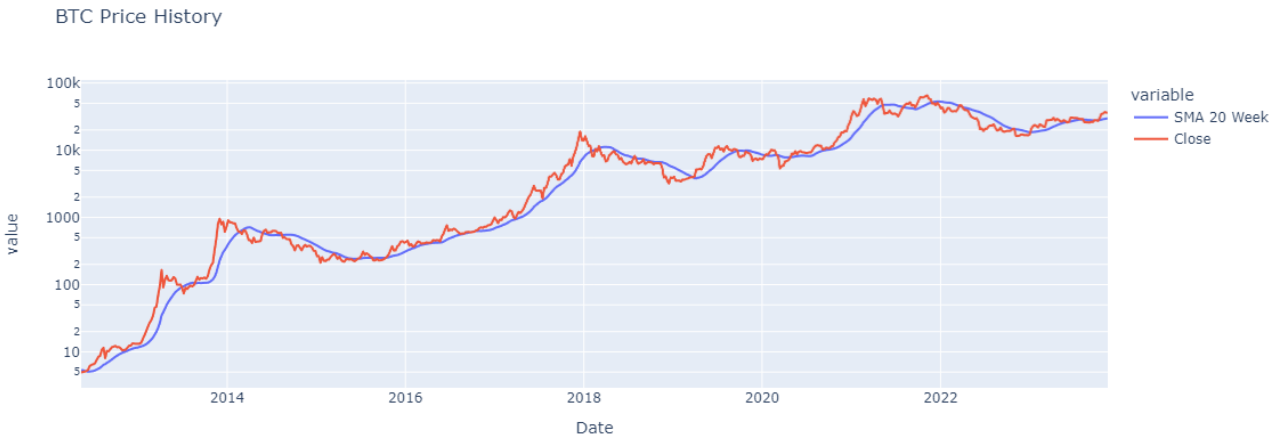

Figure 1: BTC and SMA

The graph is in log scale for easier visualization of the data. We will then compute the extension of the closing price from the SMA, which has the following expression:

$$\lambda = \ln\left(\frac{x}{\mu}\right) \tag{1}$$

Where x is the price of BTC, μ is the 20 week SMA and λ is the extension. The SMA can be computed as follows:

$$\mu = \frac{1}{20}\sum_{i=0}^{19} x[n-i] \tag{2}$$

Where $x[n-i]$ for $i = 0, 1, 2, 3...$ denotes the previous close values of BTC.

Plotting the extension, we get:

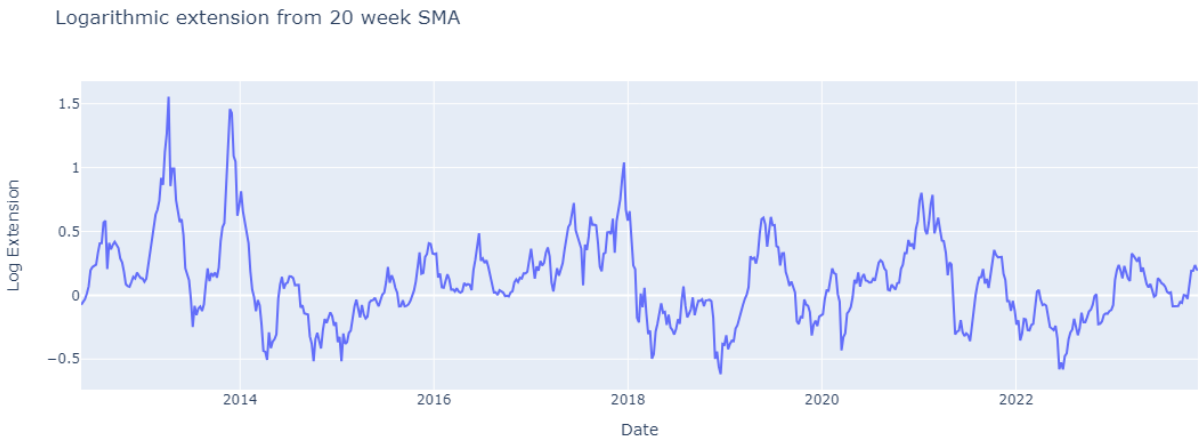

Figure 2: Extension from the 20 week SMA

As we are considering only the Bitcoin market tops, we will remove the negative values of the graph because we are only interested in the moments where BTC extends above the SMA and not below.

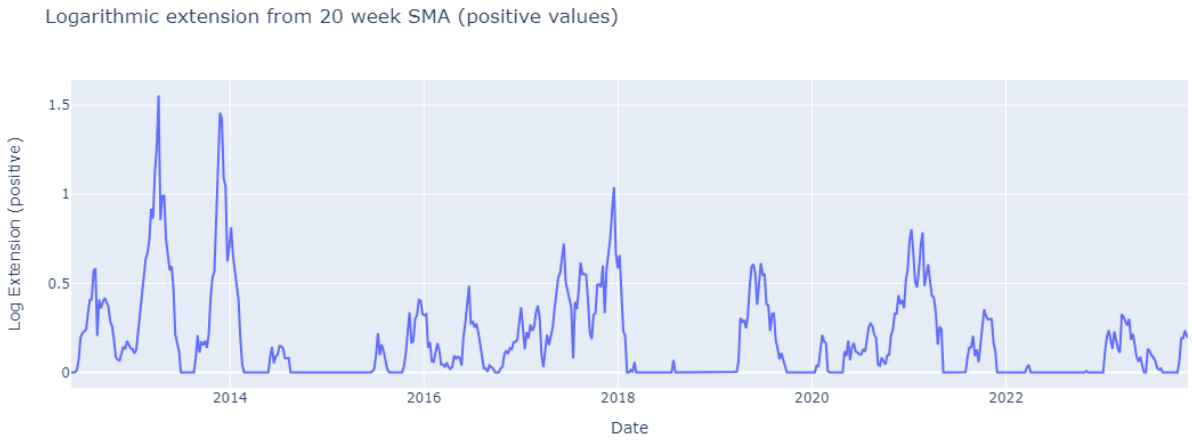

Figure 3: Extension (positive values only)

The next step will be fitting a curve to the points of maximum extension. We will choose a decreasing exponential, as a linear regression would mean that the log extension tends to only negative values at some point (which is unrealistic). The following expression gives the exponential fit:

$$f(x) = ae^{-bx} \tag{3}$$

We have selected only the first three peaks, and the coefficients are $a = 1.671$ and $b = 0.0016$. As the fourth and fifth peaks are not fitted, we can check how good the fit is by looking at these values and their corresponding fit.

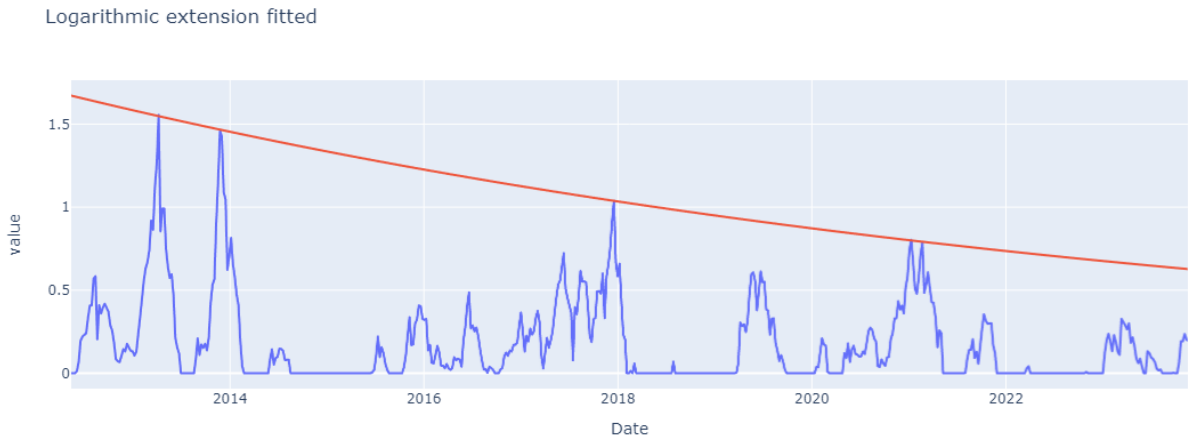

Figure 4: Decreasing exponential fit

Now, we will turn these calculations into an oscillator. An oscillator is an indicator ranging from 0 to 1 that shows how present a feature is in the data we are observing. In our case, the value of 1 means that the price of BTC is at its maximum extension point from the SMA, thus marking a market cycle top. In other words, the oscillator gives the degree of extension of the price according to historical values.

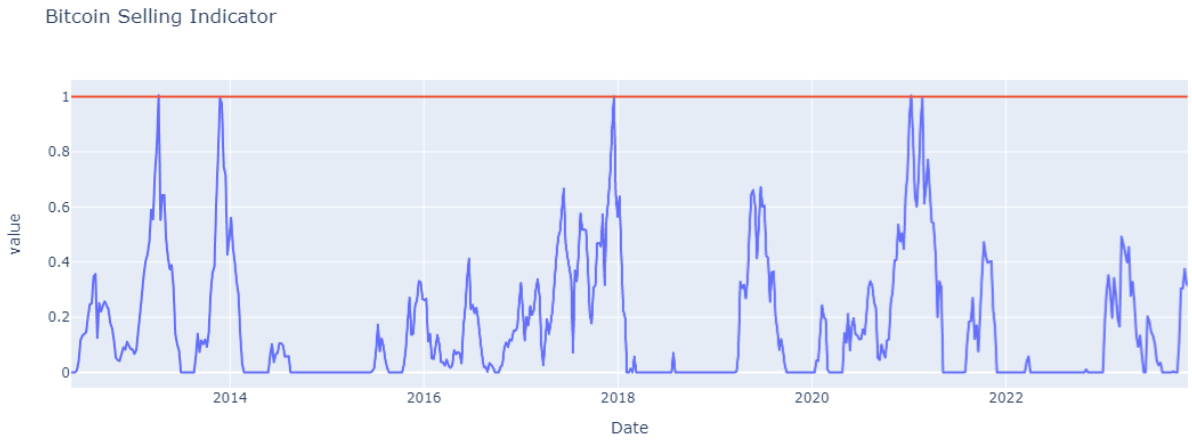

Figure 5: Oscillator

If we check how close the fit is to the fourth and fifth peaks, we get a difference of around 0.5% for both peaks, which means the fit is pretty accurate. Creating a band is one way to visually check the oscillator's effectiveness. The points where the price crosses the band are supposed to be good selling points because they potentially represent market cycle tops. The band has the following expression:

$$\lambda_B = \mu e^{f(x)} \tag{4}$$

Where λ_B is the selling band, μ is the 20 week SMA, and $f(x)$ is the fit.

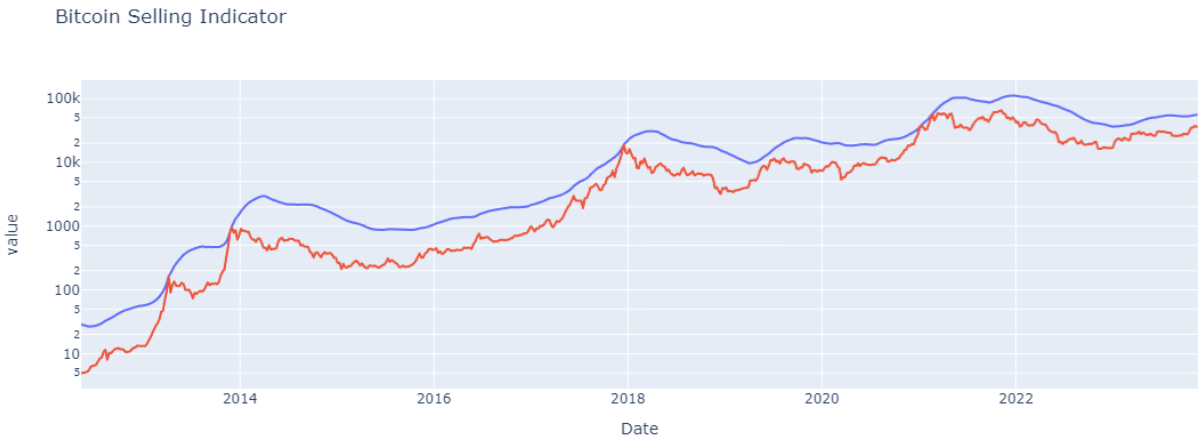

Figure 6: Selling band

4 Results and Accuraccy

Having fitted the first three peaks, we can now check the price difference between the band and the market cycle top of the unfitted peaks. In the 2020-2021 bull market, BTC reached a maximum value of 69K$. Our band highlighted two selling points, one at 38K$ and the other at 57K$. The first selling point is quite far from the top. However, it was a local maximum and represented a good point to take some profits, as BTC had been increasing significantly. On the other hand, the second selling point was closer to the top of 69K$, about a 20% difference, which is very accurate, and also represented a good point to take profits as it was a local maximum.

The conclusion of our study is that the indicator is able to predict local maximums in the context of a Bitcoin bull market. For this reason, it can be used as a taking-profit strategy effectively.

5 Limitations and Improvements

In this section, we will explain some of the limitations of our model, and then we will propose some improvements.

The first limitation is related to the data used for the regression. As we have just a few peaks, we have performed the fit with only three data points, which is not ideal. In addition, we have chosen the closing price as our source data; this presents a problem since close values ignore the high values of the week, thus masking the actual peaks. Finally, we are assuming a periodic behavior of BTC (due to the halving effect), which might disappear in the future, resulting in poor performance of the indicator.

About the improvements, fitting more peaks would definitely improve the model's effectiveness. We could start by fitting the last two peaks to predict the following ones more accurately. Additionally, it would be interesting to explore a model where the source data is the high values instead of the close values of BTC, as the BTC market cycle tops correspond to high values and not close values (in most cases).

6 Appendix: Python Code

```
import yfinance as yf
import pandas as pd
import plotly.express as px
import numpy as np
from scipy.signal import find_peaks
from scipy.optimize import curve_fit


# Data from kaggle
data = pd.read_csv(r'C:\Users\Usuario\Desktop\PYTHON\Indicador BTC\data.csv').dropna()

# Group data into weekly intervals
data['Date'] = pd.to_datetime(data['Timestamp'], unit='s')
data.set_index('Date', inplace=True)
data_wk = data.resample('W').last()
data_kg = data_wk[['Open','Close','High','Low']]

# Get the most recent data with yfinance
start = '2021-04-13'
data_yf = yf.download('BTC-USD', start = start, interval = '1wk', progress = False )
data_yf = data_yf[['Open','Close','High','Low']]

# Concatenate both datsets
btc = pd.concat([data_kg,data_yf], axis = 0)

# Compute 20 week SMA
btc['SMA 20 Week'] = pd.Series(btc['Close']).rolling(20).mean()
btc = btc.dropna()

# Plot dataset
px.line(btc, y = [btc['SMA 20 Week'],btc['Close']], log_y=True, title = 'BTC Price History')

# Extension from 20 week SMA:
btc['SMA Extension'] = btc['Close']/btc['SMA 20 Week']
px.line(btc, y = btc['SMA Extension'], title = 'Extension from 20 week SMA')

# Logarithmic extension:
btc['Log Extension'] = np.log(abs(btc['SMA Extension']))
px.line(btc, y = btc['Log Extension'], title = 'Logarithmic extension from 20 week SMA')

# Logarithmic extension (only positive values):
btc['Log Extension (positive)'] = btc['Log Extension']*(btc['Log Extension']>0)
```

```
px.line(btc, y = btc['Log Extension (positive)'], title = '
    Logarithmic extension from 20 week SMA (positive values)')

# Select the 3 highest peaks:
y = btc['Log Extension (positive)'].values
x = np.arange(0,len(y))
peaks, _ = find_peaks(y, height = 1, distance = 20)
y_data = y[peaks]; x_data = x[peaks]

# Define the exponential function:
def exponential_func(x, a, b):
    return a * np.exp(-b * x)

# Provide initial guesses for parameters a and b:
initial_guesses = [1.0, 0.01]

# Fit the exponential curve to the data with initial guesses:
params, covariance = curve_fit(exponential_func, x_data, y_data,
    p0=initial_guesses)
a_fit, b_fit = params
btc['Fit'] = exponential_func(x, a_fit, b_fit)

fig = px.line(btc, y = [btc['Log Extension (positive)'], btc['
    Fit']], title = 'Logarithmic extension fitted')
fig.update_layout(showlegend=False)

# Oscillator:
btc['Oscillator'] = btc['Log Extension (positive)']/btc['Fit']
btc['Line'] = 1
fig = px.line(btc, y = [btc['Oscillator'],btc['Line']], title =
    'Bitcoin Selling Indicator')
fig.update_layout(showlegend=False)

# Selling Band:
btc['Band'] = np.exp(btc['Fit'])*btc['SMA 20 Week']
fig = px.line(btc, y = [btc['Band'],btc['Close']], title = '
    Bitcoin Selling Indicator', log_y= True)
fig.update_layout(showlegend=False)
fig.show()
```