Analysis

Hypothesis: Positive sentiment score leads to price increases and negative sentiment score leads to price decreases for Bitcoin.

Data description

The data comes from Augmento, which constantly collects cryptocurrency related conversations from Twitter, Reddit & Bitcointalk. Using a classifier trained on crypto specific language, the data is analyzed according to 93 sentiments and topics.

At a resolution of 1h, the provided data reflects individual social media post counts starting from November 2016 to October 2019.

```python
import pandas as pd
import numpy as np

augmento = pd.read_csv('augmento_btc.csv')
augmento['date'] = pd.to_datetime(augmento['date'])
augmento = augmento.iloc[:26280]
augmento = augmento.dropna()
print(augmento.tail())
```

```
                     date  listing_close  twitter_hacks  \
26275 2019-10-31 20:00:00        9190.81            0.0
26276 2019-10-31 21:00:00        9135.69            0.0
26277 2019-10-31 22:00:00        9127.50            0.0
26278 2019-10-31 23:00:00        9150.07            0.0
26279 2019-11-01 00:00:00        9134.08            0.0

       twitter_pessimistic_doubtful  twitter_banks  twitter_selling  \
26275                           1.0            2.0              1.0
26276                           1.0            2.0              1.0
26277                           1.0            2.0              1.0
26278                           2.0            0.0              1.0
26279                           0.0            0.0              0.0

       twitter_market_manipulation  twitter_de_centralisation
twitter_angry  \
26275                          2.0                        0.0
0.0
26276                          1.0                        1.0
0.0
26277                          0.0                        0.0
0.0
26278                          0.0                        1.0
0.0
```

```
26279                           1.0                        2.0
0.0

       twitter_etf  ...  reddit_buying  reddit_warning  \
26275          2.0  ...            4.0             0.0
26276          0.0  ...            2.0             1.0
26277          0.0  ...            1.0             0.0
26278          0.0  ...            4.0             0.0
26279          1.0  ...            5.0             0.0

       reddit_annoyed_frustrated  reddit_price
reddit_use_case_applications  \
26275                        0.0          14.0
4.0
26276                        0.0          18.0
7.0
26277                        0.0          11.0
9.0
26278                        0.0           9.0
10.0
26279                        0.0           6.0
7.0

       reddit_rumor  reddit_scam_fraud  reddit_airdrop
reddit_optimistic  \
26275           0.0                3.0             0.0
4.0
26276           0.0                1.0             0.0
1.0
26277           1.0                1.0             0.0
4.0
26278           0.0                4.0             0.0
5.0
26279           0.0                0.0             0.0
2.0

       reddit_negative
26275             21.0
26276             34.0
26277             20.0
26278             21.0
26279             10.0

[5 rows x 281 columns]
```

There are 26280 time stamps with 279 sentiments from 11/1/2016 1:00 to 2019-11-01 0:00. Each of the 3 social media has 93 sentiments.

According to Augmento, the pre-defined positive sentiments are:

- Bullish
- Optimistic
- Happy
- Euphoric/Excited
- Positive

Negative sentiments are:

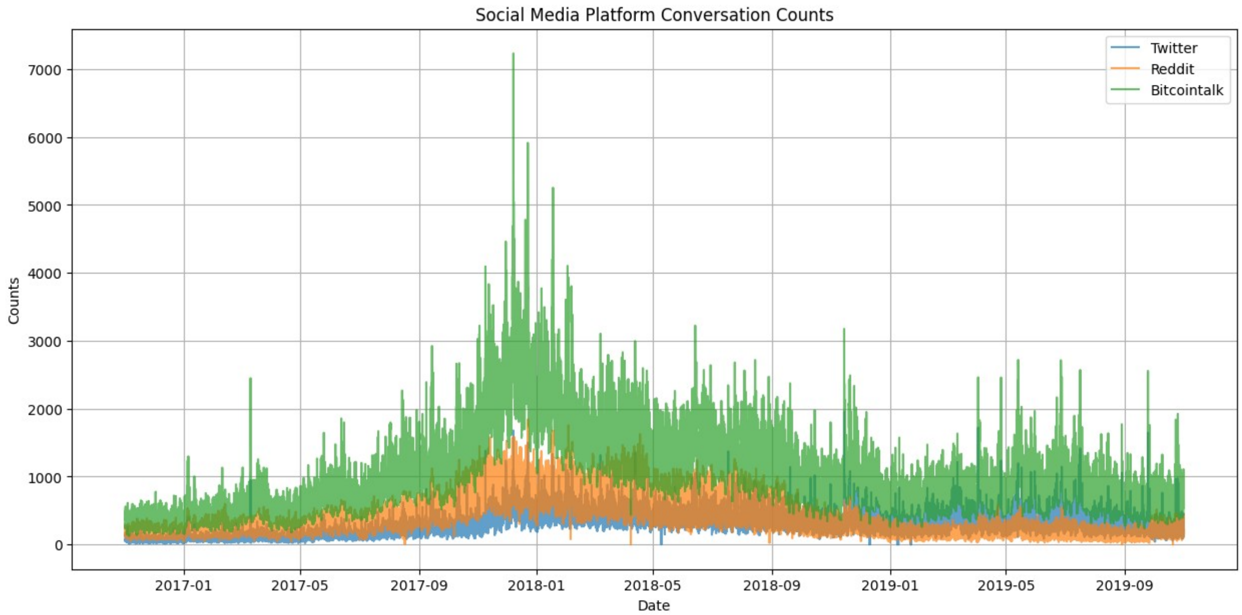
- Bearish
- Pessimistic/Doubtful
- Sad
- Fearful/Concerned
- Angry
- Mistrustful
- Panicking
- Annoyed/Frustrated
- Negative

There are also other sentiments worth considering:

- Cheap
- Expensive
- Bubble
- Good news
- Bad news

However, the detailed weighting method would depend on our selection. Eventually, we will build the final sentiment score based on those individual sentiments.

```python
import matplotlib.pyplot as plt
df = augmento.copy()
df['a'] = df.iloc[:, 2:95].sum(axis=1)
df['b'] = df.iloc[:, 96:189].sum(axis=1)
df['c'] = df.iloc[:, 190:283].sum(axis=1)

plt.figure(figsize=(15, 7))
plt.plot(df['date'], df['a'], label='Twitter', alpha=0.7)
plt.plot(df['date'], df['b'], label='Reddit', alpha=0.7)
plt.plot(df['date'], df['c'], label='Bitcointalk', alpha=0.7)
plt.title('Social Media Platform Conversation Counts')
plt.xlabel('Date')
plt.ylabel('Counts')
plt.legend()
plt.grid(True)
plt.show()
```



The Bitcointalk is way more active than the other two social media platforms. At each timestamp, we will assign weights to each social media platform based on the proportion of conversations occurring on that platform relative to the total number of conversations across all platforms.

In the Bitcoin Tradingg Strategies based on Twitter Sentiment Analysis (Tsoulias, K. 2020), the author provided an interesting graph that reveals some correlation between tweets and Bitcoin price.

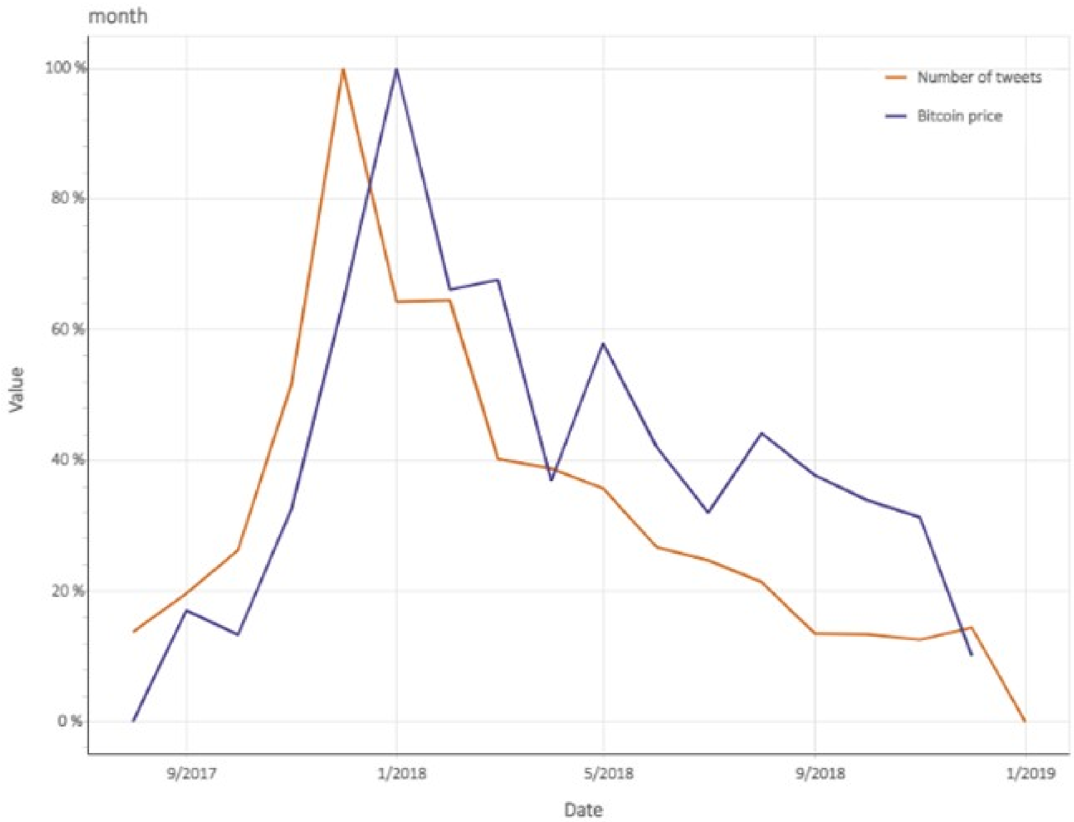
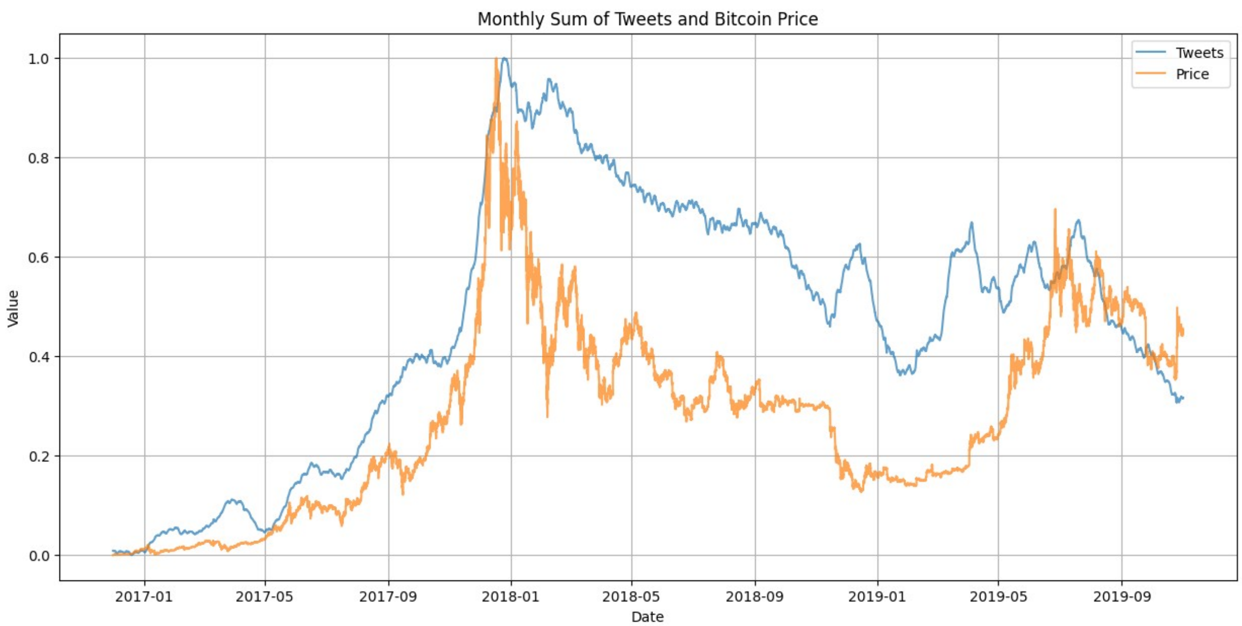


Figure 4.15: Number of unlabelled tweets and Bitcoin price in monthly timeframe.

We will examine this a little bit with the Augmento data.

```
df = augmento.copy()
df['a'] = df.iloc[:, 2:95].sum(axis=1)
df['a_monthly_sum'] = df['a'].rolling(window=720).sum()
df_monthly_sum = df.dropna(subset=['a_monthly_sum']).copy()
df_monthly_sum['a_monthly_sum_normalized'] =
(df_monthly_sum['a_monthly_sum'] -
df_monthly_sum['a_monthly_sum'].min()) /
(df_monthly_sum['a_monthly_sum'].max() -
df_monthly_sum['a_monthly_sum'].min())
df_monthly_sum['listing_close_normalized'] =
(df_monthly_sum['listing_close'] -
df_monthly_sum['listing_close'].min()) /
(df_monthly_sum['listing_close'].max() -
df_monthly_sum['listing_close'].min())

plt.figure(figsize=(15, 7))
plt.plot(df_monthly_sum['date'],
df_monthly_sum['a_monthly_sum_normalized'], label='Tweets', alpha=0.7)
plt.plot(df_monthly_sum['date'],
```

```
df_monthly_sum['listing_close_normalized'], label='Price', alpha=0.7)
plt.title('Monthly Sum of Tweets and Bitcoin Price')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()
```



The result is similar to the graph provided, except that after 2019-06 the correlation may become insignificant or even negative.

Now back to the sentiments, here is the list of all Twitter sentiments:

```
df = augmento.copy()
df['a'] = df.iloc[:, 2:95].sum(axis=1)
df['b'] = df.iloc[:, 96:189].sum(axis=1)
df['c'] = df.iloc[:, 190:283].sum(axis=1)
df.columns[2:95]
```

```
Index(['twitter_hacks', 'twitter_pessimistic_doubtful',
'twitter_banks',
       'twitter_selling', 'twitter_market_manipulation',
       'twitter_de_centralisation', 'twitter_angry', 'twitter_etf',
       'twitter_leverage', 'twitter_bottom',
'twitter_institutional_money',
       'twitter_fomo', 'twitter_prediction', 'twitter_adoption',
       'twitter_fearful_concerned', 'twitter_portfolio',
'twitter_fud_theme',
       'twitter_whitepaper', 'twitter_announcements',
       'twitter_technical_analysis', 'twitter_flippening',
```

```
'twitter_community',
       'twitter_investing_trading', 'twitter_euphoric_excited',
       'twitter_hodling', 'twitter_ico', 'twitter_bearish',
       'twitter_going_short', 'twitter_uncertain', 'twitter_volume',
       'twitter_risk', 'twitter_governance', 'twitter_ban',
'twitter_cheap',
       'twitter_short_term_trading', 'twitter_fork',
'twitter_progress',
       'twitter_shilling', 'twitter_bullish', 'twitter_happy',
       'twitter_bubble', 'twitter_bots', 'twitter_hopeful',
'twitter_bug',
       'twitter_open_source', 'twitter_token_economics',
'twitter_security',
       'twitter_marketing', 'twitter_bad_news',
'twitter_due_diligence',
       'twitter_team', 'twitter_partnerships',
'twitter_pump_and_dump',
       'twitter_sad', 'twitter_panicking', 'twitter_listing',
       'twitter_regulation_politics', 'twitter_dip', 'twitter_launch',
       'twitter_fomo_theme', 'twitter_advice_support',
'twitter_rebranding',
       'twitter_wallet', 'twitter_good_news',
'twitter_problems_and_issues',
       'twitter_mining', 'twitter_waiting', 'twitter_learning',
       'twitter_scaling', 'twitter_fees', 'twitter_roadmap',
       'twitter_recovery', 'twitter_technology',
'twitter_mistrustful',
       'twitter_marketcap', 'twitter_positive', 'twitter_tax',
       'twitter_long_term_investing', 'twitter_strategy',
       'twitter_competition', 'twitter_whales', 'twitter_correction',
       'twitter_stablecoin', 'twitter_buying', 'twitter_warning',
       'twitter_annoyed_frustrated', 'twitter_price',
       'twitter_use_case_applications', 'twitter_rumor',
'twitter_scam_fraud',
       'twitter_airdrop', 'twitter_optimistic', 'twitter_negative'],
      dtype='object')
```

For simplicity, at each timestamp within each social media platform, we will assign the following scores to each useful sentiment:

- Each positve sentiment metioned above has a score of +1
- Each negative sentiment metioned above has a score of -0.5

Then, calculate the final aggregated score based on corresponding weights across all social media.

```
df['twitter'] = (
    df['twitter_bullish'] +
    df['twitter_optimistic'] +
```

```
    df['twitter_happy'] +
    df['twitter_euphoric_excited'] +
    df['twitter_positive'] -
    0.5 * df['twitter_bearish'] -
    0.5 * df['twitter_pessimistic_doubtful'] -
    0.5 * df['twitter_sad'] -
    0.5 * df['twitter_fearful_concerned'] -
    0.5 * df['twitter_angry'] -
    0.5 * df['twitter_mistrustful'] -
    0.5 * df['twitter_panicking'] -
    0.5 * df['twitter_annoyed_frustrated'] -
    0.5 * df['twitter_negative']
)

df['reddit'] = (
    df['reddit_bullish'] +
    df['reddit_optimistic'] +
    df['reddit_happy'] +
    df['reddit_euphoric_excited'] +
    df['reddit_positive'] -
    0.5 * df['reddit_bearish'] -
    0.5 * df['reddit_pessimistic_doubtful'] -
    0.5 * df['reddit_sad'] -
    0.5 * df['reddit_fearful_concerned'] -
    0.5 * df['reddit_angry'] -
    0.5 * df['reddit_mistrustful'] -
    0.5 * df['reddit_panicking'] -
    0.5 * df['reddit_annoyed_frustrated'] -
    0.5 * df['reddit_negative']
)

df['bitcointalk'] = (
    df['bitcointalk_bullish'] +
    df['bitcointalk_optimistic'] +
    df['bitcointalk_happy'] +
    df['bitcointalk_euphoric_excited'] +
    df['bitcointalk_positive'] -
    0.5 * df['bitcointalk_bearish'] -
    0.5 * df['bitcointalk_pessimistic_doubtful'] -
    0.5 * df['bitcointalk_sad'] -
    0.5 * df['bitcointalk_fearful_concerned'] -
    0.5 * df['bitcointalk_angry'] -
    0.5 * df['bitcointalk_mistrustful'] -
    0.5 * df['bitcointalk_panicking'] -
    0.5 * df['bitcointalk_annoyed_frustrated'] -
    0.5 * df['bitcointalk_negative']
)
```

Indicator

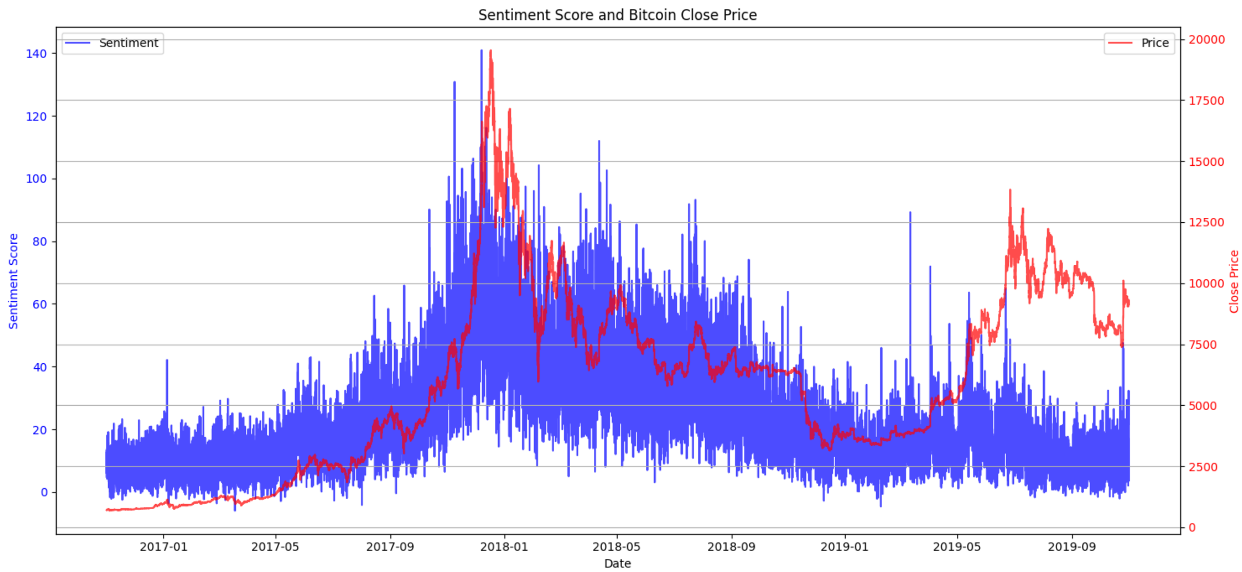
The indicator is this final sentiment score, and we will test its correlation with the Bitcoin price.

```
df['sentiment'] = (
    df['twitter']*df['a']/(df['a']+df['b']+df['c'])+
    df['reddit']*df['b']/(df['a']+df['b']+df['c'])+
    df['bitcointalk']*df['c']/(df['a']+df['b']+df['c'])
)
df['sentiment'].corr(df['listing_close'])
```

```
0.5230581447341971
```

The correlation of this sentiment score with the Bitcoin price is +0.52, which is a really good value considering the volatility of Bitcoin. This indicator will indeed be served as the triggering signal in the strategy.

```
fig, ax1 = plt.subplots(figsize=(15, 7))
ax1.plot(df['date'], df['sentiment'], label='Sentiment', color='blue',
alpha=0.7)
ax1.set_xlabel('Date')
ax1.set_ylabel('Sentiment Score', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')
ax2 = ax1.twinx()
ax2.plot(df['date'], df['listing_close'], label='Price', color='red',
alpha=0.7)
ax2.set_ylabel('Close Price', color='red')
ax2.tick_params(axis='y', labelcolor='red')
plt.title('Sentiment Score and Bitcoin Close Price')
fig.tight_layout()
plt.grid(True)
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')

plt.show()
```




```
import statsmodels.api as sm
y = df['listing_close']
X = df['sentiment']
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
print(model.summary())

                            OLS Regression Results

==============================================================================
========
Dep. Variable:          listing_close   R-squared:
0.274
Model:                            OLS   Adj. R-squared:
0.274
Method:                 Least Squares   F-statistic:
9896.
Date:                Mon, 20 May 2024   Prob (F-statistic):
0.00
Time:                        17:42:55   Log-Likelihood:                 -
2.4882e+05
No. Observations:               26278   AIC:
4.976e+05
Df Residuals:                   26276   BIC:
4.977e+05
Df Model:                           1

Covariance Type:            nonrobust

==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
```

```
0.975]
------------------------------------------------------------------------------
const       3279.1575     32.554    100.728      0.000    3215.349
3342.966
sentiment    111.9738      1.126     99.481      0.000     109.768
114.180
==============================================================================
Omnibus:                     2196.806   Durbin-Watson:
0.118
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
2792.129
Skew:                           0.796   Prob(JB):
0.00
Kurtosis:                       2.883   Cond. No.
48.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

The coefficient for sentiment is significantly positive, as indicated by a very high t-statistic and a p-value of 0.000. This means that there is a statistically significant relationship between sentiment score and Bitcoin prices, where a one-unit increase in sentiment score is associated with an increase of approximately 111.9738 in the Bitcoin price.

We have tested our hypothesis and conclude that the hypothesis is true. Therefore, we will develop a trading strategy based on the sentiment score.

Strategy

Constraints, Benchmark, Objective

Constraints: We will start with initial cash of $10,000. We will inplement stop loss of 90% into the strategy. No transaction fees.

Benchmark: Bitcoin prices.

Objective: Replicate the price movement of Bitcoin.

```
initial_cash = 10000
stop_loss_percent = 0.90
max_drawdown = 0
benchmark_value = initial_cash / df['listing_close'].iloc[0] *
df['listing_close']
```

Signal process

Sentiment > 0, buy and hold.

Sentiment < 0, sell.

Below is a simple test on mock data.

```
def test_trading_strategy(data, stop_loss_percent):
    cash = 10000
    btc_held = 0
    peak_price = 0
    portfolio_values = []

    for index, row in data.iterrows():
        current_price = row['listing_close']
        sentiment = row['sentiment']
        if btc_held > 0:
            peak_price = max(peak_price, current_price)
        if btc_held > 0 and (current_price <= peak_price * 
stop_loss_percent or sentiment <= 0):
            cash = btc_held * current_price
            btc_held = 0
            peak_price = 0
        elif sentiment > 0 and cash > 0:
            btc_held = cash / current_price
            cash = 0
            peak_price = current_price

        portfolio_values.append(cash + btc_held * current_price)

    return portfolio_values
mock_data = pd.DataFrame({
    'listing_close': [40000, 42000, 41000, 38000, 39000, 37000, 36000, 
38000],
    'sentiment': [1, 1, -1, 1, 1, -1, 1, 1]
})

portfolio_values = test_trading_strategy(mock_data, 0.90)
print(portfolio_values)
```

```
[10000.0, 10500.0, 10250.0, 10250.0, 10519.736842105263, 
9980.263157894737, 9980.263157894737, 10534.722222222223]
```

Rule process

- Positive Sentiment Check: If the sentiment is positive and there is cash available, buy Bitcoin.
- Update Peak Price: If Bitcoin is held, update the peak_price to the maximum of the current peak_price or the current price.

- Stop Loss Check: If the current price falls below a set percentage (e.g., 90%) of the peak_price or if the sentiment score turns negative, sell all Bitcoin.
- Performance Measure: Maximum Drawdown and annualized volatility.

```
initial_cash = 10000
stop_loss_percent = 0.90
max_drawdown = 0
benchmark_value = initial_cash / df['listing_close'].iloc[0] *
df['listing_close']
cash = initial_cash
btc_held = 0
peak_price = 0
portfolio_value = []
peak_portfolio_value = 0
returns = []
max_drawdown = 0

def execute_trade(cash, btc_held, action, current_price):
    if action == 'buy':
        btc_held = cash / current_price
        cash = 0
    elif action == 'sell':
        cash = btc_held * current_price
        btc_held = 0
    return cash, btc_held

for index, row in df.iterrows():
    current_price = row['listing_close']
    if btc_held > 0:
        peak_price = max(peak_price, current_price)
    if btc_held > 0 and (current_price <= peak_price *
stop_loss_percent or row['sentiment'] <= 0):
        cash, btc_held = execute_trade(cash, btc_held, 'sell',
current_price)
        peak_price = 0
    elif row['sentiment'] > 0 and cash > 0:
        cash, btc_held = execute_trade(cash, btc_held, 'buy',
current_price)
        peak_price = current_price

    current_portfolio_value = cash + btc_held * current_price
    portfolio_value.append(current_portfolio_value)

    if len(portfolio_value) > 1:
        daily_return = (portfolio_value[-1] - portfolio_value[-2]) /
portfolio_value[-2]
        returns.append(daily_return)

    peak_portfolio_value = max(peak_portfolio_value,
```

```
current_portfolio_value)
    current_drawdown = (peak_portfolio_value - 
current_portfolio_value) / peak_portfolio_value
    max_drawdown = max(max_drawdown, current_drawdown)

volatility = np.std(returns) * np.sqrt(252)

df['portfolio_value'] = portfolio_value
df['benchmark_value'] = benchmark_value
df['drawdown'] = [(peak_portfolio_value - pv) / peak_portfolio_value 
for pv in portfolio_value]

plt.figure(figsize=(14, 7))
plt.plot(df['portfolio_value'], label='Portfolio Value')
plt.plot(df['benchmark_value'], label='Benchmark Value', color='red')
plt.title('Trading Strategy vs. Benchmark')
plt.legend()
plt.show()

print(f"Maximum Drawdown: {max_drawdown * 100:.2f}%")
print(f"Annualized Volatility: {volatility * 100:.2f}%")
```


```
Maximum Drawdown: 85.45%
Annualized Volatility: 14.97%
```

With the stop-loss set at 90% of the initial value, this strategy follows under the price of Bitcoin closely, where our objective of replicating Bitcoin prices is quite successful. However, this is an unrealistic set up where there is no transaction fee. In this case, the sentiment score fluctuates a lot, so with the existence of transaction fee, the difference between the actual and replicating portfolios will be larger and larger.

Optimization

In this strategy, the parameters is actually the weights for different sentiments, however, it is not an easy task to optimize them.

Here, we could test the simplest weighting methods.

```
df['twitter'] = (
    df['twitter_bullish'] +
    df['twitter_optimistic'] +
    df['twitter_happy'] +
    df['twitter_euphoric_excited'] +
    df['twitter_positive'] -
    df['twitter_bearish'] -
    df['twitter_pessimistic_doubtful'] -
    df['twitter_sad'] -
    df['twitter_fearful_concerned'] -
    df['twitter_angry'] -
    df['twitter_mistrustful'] -
    df['twitter_panicking'] -
    df['twitter_annoyed_frustrated'] -
    df['twitter_negative']
)

df['reddit'] = (
    df['reddit_bullish'] +
    df['reddit_optimistic'] +
    df['reddit_happy'] +
    df['reddit_euphoric_excited'] +
    df['reddit_positive'] -
    df['reddit_bearish'] -
    df['reddit_pessimistic_doubtful'] -
    df['reddit_sad'] -
    df['reddit_fearful_concerned'] -
    df['reddit_angry'] -
    df['reddit_mistrustful'] -
    df['reddit_panicking'] -
    df['reddit_annoyed_frustrated'] -
    df['reddit_negative']
)

df['bitcointalk'] = (
    df['bitcointalk_bullish'] +
    df['bitcointalk_optimistic'] +
    df['bitcointalk_happy'] +
    df['bitcointalk_euphoric_excited'] +
    df['bitcointalk_positive'] -
```

```
    df['bitcointalk_bearish'] -
    df['bitcointalk_pessimistic_doubtful'] -
    df['bitcointalk_sad'] -
    df['bitcointalk_fearful_concerned'] -
    df['bitcointalk_angry'] -
    df['bitcointalk_mistrustful'] -
    df['bitcointalk_panicking'] -
    df['bitcointalk_annoyed_frustrated'] -
    df['bitcointalk_negative']
)
df['sentiment'] = (
    df['twitter']/3+
    df['reddit']/3+
    df['bitcointalk']/3
)
df['sentiment'].corr(df['listing_close'])
```

```
0.0771109606291325
```

This simple method assigns positive sentiments score of +1, negative sentiments score of -1, and then equal weighting for the three social media platforms. However, the ending correlation 0.07711 is significantly lower.

```
initial_cash = 10000
stop_loss_percent = 0.90
max_drawdown = 0
cash = initial_cash
btc_held = 0
peak_price = 0
portfolio_value = []
peak_portfolio_value = 0
returns = []
max_drawdown = 0
benchmark_value = initial_cash / df['listing_close'].iloc[0] *
df['listing_close']

def execute_trade(cash, btc_held, action, current_price):
    if action == 'buy':
        btc_held = cash / current_price
        cash = 0
    elif action == 'sell':
        cash = btc_held * current_price
        btc_held = 0
    return cash, btc_held

for index, row in df.iterrows():
    current_price = row['listing_close']
    if btc_held > 0:
        peak_price = max(peak_price, current_price)
```

```
    if btc_held > 0 and (current_price <= peak_price *
stop_loss_percent or row['sentiment'] <= 0):
        cash, btc_held = execute_trade(cash, btc_held, 'sell',
current_price)
        peak_price = 0
    elif row['sentiment'] > 0 and cash > 0:
        cash, btc_held = execute_trade(cash, btc_held, 'buy',
current_price)
        peak_price = current_price

    current_portfolio_value = cash + btc_held * current_price
    portfolio_value.append(current_portfolio_value)

    if len(portfolio_value) > 1:
        daily_return = (portfolio_value[-1] - portfolio_value[-2]) /
portfolio_value[-2]
        returns.append(daily_return)

    peak_portfolio_value = max(peak_portfolio_value,
current_portfolio_value)
    current_drawdown = (peak_portfolio_value -
current_portfolio_value) / peak_portfolio_value
    max_drawdown = max(max_drawdown, current_drawdown)

volatility = np.std(returns) * np.sqrt(252)

df['portfolio_value'] = portfolio_value
df['benchmark_value'] = benchmark_value
df['drawdown'] = [(peak_portfolio_value - pv) / peak_portfolio_value
for pv in portfolio_value]

plt.figure(figsize=(14, 7))
plt.plot(df['portfolio_value'], label='Portfolio Value')
plt.plot(df['benchmark_value'], label='Benchmark Value', color='red')
plt.title('Trading Strategy vs. Benchmark')
plt.legend()
plt.show()

print(f"Maximum Drawdown: {max_drawdown * 100:.2f}%")
print(f"Annualized Volatility: {volatility * 100:.2f}%")
```


```
Maximum Drawdown: 38.15%
Annualized Volatility: 9.05%
```

Surprisingly, although this strategy failed at replicating the price of Bitcoin, it ended up being in the similar porfolio value as the original strategy. The maximum drawdown decreased from 85.45% to 38.15% significantly, and the volatility decreased to 9.05%. So this would be the case where the investor is more risk averse.

Walk Forward Analysis

```
df['date'] = pd.to_datetime(df['date'])
df.set_index('date', inplace=True)
df['twitter'] = (
    df['twitter_bullish'] +
    df['twitter_optimistic'] +
    df['twitter_happy'] +
    df['twitter_euphoric_excited'] +
    df['twitter_positive'] -
    0.5 * df['twitter_bearish'] -
    0.5 * df['twitter_pessimistic_doubtful'] -
    0.5 * df['twitter_sad'] -
    0.5 * df['twitter_fearful_concerned'] -
    0.5 * df['twitter_angry'] -
    0.5 * df['twitter_mistrustful'] -
    0.5 * df['twitter_panicking'] -
    0.5 * df['twitter_annoyed_frustrated'] -
    0.5 * df['twitter_negative']
)

df['reddit'] = (
```

```
    df['reddit_bullish'] +
    df['reddit_optimistic'] +
    df['reddit_happy'] +
    df['reddit_euphoric_excited'] +
    df['reddit_positive'] -
    0.5 * df['reddit_bearish'] -
    0.5 * df['reddit_pessimistic_doubtful'] -
    0.5 * df['reddit_sad'] -
    0.5 * df['reddit_fearful_concerned'] -
    0.5 * df['reddit_angry'] -
    0.5 * df['reddit_mistrustful'] -
    0.5 * df['reddit_panicking'] -
    0.5 * df['reddit_annoyed_frustrated'] -
    0.5 * df['reddit_negative']
)

df['bitcointalk'] = (
    df['bitcointalk_bullish'] +
    df['bitcointalk_optimistic'] +
    df['bitcointalk_happy'] +
    df['bitcointalk_euphoric_excited'] +
    df['bitcointalk_positive'] -
    0.5 * df['bitcointalk_bearish'] -
    0.5 * df['bitcointalk_pessimistic_doubtful'] -
    0.5 * df['bitcointalk_sad'] -
    0.5 * df['bitcointalk_fearful_concerned'] -
    0.5 * df['bitcointalk_angry'] -
    0.5 * df['bitcointalk_mistrustful'] -
    0.5 * df['bitcointalk_panicking'] -
    0.5 * df['bitcointalk_annoyed_frustrated'] -
    0.5 * df['bitcointalk_negative']
)
df['sentiment'] = (
    df['twitter']*df['a']/(df['a']+df['b']+df['c'])+
    df['reddit']*df['b']/(df['a']+df['b']+df['c'])+
    df['bitcointalk']*df['c']/(df['a']+df['b']+df['c'])
)
df['sentiment'].corr(df['listing_close'])
in_sample_years = 1
out_of_sample_months = 3

def execute_trade(cash, btc_held, action, current_price):
    if action == 'buy':
        btc_held = cash / current_price
        cash = 0
    elif action == 'sell':
        cash = btc_held * current_price
        btc_held = 0
    return cash, btc_held
```

```
def walk_forward_analysis(df, in_sample_years, out_of_sample_months):
    offset_in_sample = pd.DateOffset(years=in_sample_years)
    offset_out_sample = pd.DateOffset(months=out_of_sample_months)

    start_date = df.index.min()
    end_date = df.index.max()

    while start_date + offset_in_sample + offset_out_sample <=
end_date:
        in_sample_data = df[start_date:start_date+offset_in_sample]
        out_sample_data =
df[start_date+offset_in_sample:start_date+offset_in_sample+offset_out_
sample]
        optimized_stop_loss = 0.90
        results = simulate_strategy(out_sample_data,
optimized_stop_loss)
        print(f"Results from {start_date} to
{start_date+offset_in_sample+offset_out_sample}: {results}")
        start_date += offset_out_sample

def simulate_strategy(df, stop_loss_percent):
    cash = 10000
    btc_held = 0
    peak_price = 0
    portfolio_value = []
    peak_portfolio_value = 0
    max_drawdown = 0

    for index, row in df.iterrows():
        current_price = row['listing_close']
        if btc_held > 0:
            peak_price = max(peak_price, current_price)
        if btc_held > 0 and (current_price <= peak_price *
stop_loss_percent or row['sentiment'] <= 0):
            cash, btc_held = execute_trade(cash, btc_held, 'sell',
current_price)
            peak_price = 0
        elif row['sentiment'] > 0 and cash > 0:
            cash, btc_held = execute_trade(cash, btc_held, 'buy',
current_price)
            peak_price = current_price

        current_portfolio_value = cash + btc_held * current_price
        portfolio_value.append(current_portfolio_value)
        peak_portfolio_value = max(peak_portfolio_value,
current_portfolio_value)
        current_drawdown = (peak_portfolio_value -
current_portfolio_value) / peak_portfolio_value
        max_drawdown = max(max_drawdown, current_drawdown)
```

```
    return {'end_cash': cash, 'max_drawdown': max_drawdown}

walk_forward_analysis(df, in_sample_years, out_of_sample_months)
```

```
Results from 2016-11-01 01:00:00 to 2018-02-01 01:00:00: {'end_cash': 0, 'max_drawdown': 0.5904964726657327}
Results from 2017-02-01 01:00:00 to 2018-05-01 01:00:00: {'end_cash': 0, 'max_drawdown': 0.4801502128124135}
Results from 2017-05-01 01:00:00 to 2018-08-01 01:00:00: {'end_cash': 8502.4861439475, 'max_drawdown': 0.4088794361432457}
Results from 2017-08-01 01:00:00 to 2018-11-01 01:00:00: {'end_cash': 0, 'max_drawdown': 0.22618750992159684}
Results from 2017-11-01 01:00:00 to 2019-02-01 01:00:00: {'end_cash': 0, 'max_drawdown': 0.5222022725041982}
Results from 2018-02-01 01:00:00 to 2019-05-01 01:00:00: {'end_cash': 0, 'max_drawdown': 0.12023392680462824}
Results from 2018-05-01 01:00:00 to 2019-08-01 01:00:00: {'end_cash': 0, 'max_drawdown': 0.363119186257962}
```

Objective function: since we used maximum drawdown as the performance measure in the strategy, the objective function in the walk forward analysis is to minimize it.

Optimizing for Maximum Drawdown: This will favor strategies that avoid large losses, which may result in frequent trading to cut losses or avoid entering positions during periods of high market uncertainty. It can help in maintaining the portfolio's stability but may reduce profitability.

Overfitting

We will test the problem of overfitting in the test data set, which is the rest of the data ranging from November 2016 to April 2024.

```
augmento = pd.read_csv('augmento_btc.csv')
augmento['date'] = pd.to_datetime(augmento['date'])
test = augmento.iloc[26281:]
test = test.dropna()
dft = test.copy()
dft['a'] = dft.iloc[:, 2:95].sum(axis=1)
dft['b'] = dft.iloc[:, 96:189].sum(axis=1)
dft['c'] = dft.iloc[:, 190:283].sum(axis=1)
dft['twitter'] = (
    dft['twitter_bullish'] +
    dft['twitter_optimistic'] +
    dft['twitter_happy'] +
    dft['twitter_euphoric_excited'] +
    dft['twitter_positive'] -
    0.5 * (dft['twitter_bearish'] +
           dft['twitter_pessimistic_doubtful'] +
           dft['twitter_sad'] +
```

```
           dft['twitter_fearful_concerned'] +
           dft['twitter_angry'] +
           dft['twitter_mistrustful'] +
           dft['twitter_panicking'] +
           dft['twitter_annoyed_frustrated'] +
           dft['twitter_negative'])
)

dft['reddit'] = (
    dft['reddit_bullish'] +
    dft['reddit_optimistic'] +
    dft['reddit_happy'] +
    dft['reddit_euphoric_excited'] +
    dft['reddit_positive'] -
    0.5 * (dft['reddit_bearish'] +
           dft['reddit_pessimistic_doubtful'] +
           dft['reddit_sad'] +
           dft['reddit_fearful_concerned'] +
           dft['reddit_angry'] +
           dft['reddit_mistrustful'] +
           dft['reddit_panicking'] +
           dft['reddit_annoyed_frustrated'] +
           dft['reddit_negative'])
)

dft['bitcointalk'] = (
    dft['bitcointalk_bullish'] +
    dft['bitcointalk_optimistic'] +
    dft['bitcointalk_happy'] +
    dft['bitcointalk_euphoric_excited'] +
    dft['bitcointalk_positive'] -
    0.5 * (dft['bitcointalk_bearish'] +
           dft['bitcointalk_pessimistic_doubtful'] +
           dft['bitcointalk_sad'] +
           dft['bitcointalk_fearful_concerned'] +
           dft['bitcointalk_angry'] +
           dft['bitcointalk_mistrustful'] +
           dft['bitcointalk_panicking'] +
           dft['bitcointalk_annoyed_frustrated'] +
           dft['bitcointalk_negative'])
)

dft['sentiment'] = (
    dft['twitter'] * dft['a'] / (dft['a'] + dft['b'] + dft['c']) +
    dft['reddit'] * dft['b'] / (dft['a'] + dft['b'] + dft['c']) +
    dft['bitcointalk'] * dft['c'] / (dft['a'] + dft['b'] + dft['c'])
)
print(dft['sentiment'].corr(dft['listing_close']))

initial_cash = 10000
```

```
stop_loss_percent = 0.90
cash = initial_cash
btc_held = 0
peak_price = 0
portfolio_value = []
returns = []
peak_portfolio_value = initial_cash
benchmark_value = initial_cash / dft['listing_close'].iloc[0] *
dft['listing_close']

def execute_trade(cash, btc_held, action, current_price):
    if action == 'buy':
        btc_held = cash / current_price
        cash = 0
    elif action == 'sell':
        cash = btc_held * current_price
        btc_held = 0
    return cash, btc_held

for index, row in dft.iterrows():
    current_price = row['listing_close']
    if btc_held > 0:
        peak_price = max(peak_price, current_price)

    if btc_held > 0 and (current_price <= peak_price *
stop_loss_percent or row['sentiment'] <= 0):
        cash, btc_held = execute_trade(cash, btc_held, 'sell',
current_price)
        peak_price = 0
    elif row['sentiment'] > 0 and cash > 0:
        cash, btc_held = execute_trade(cash, btc_held, 'buy',
current_price)
        peak_price = current_price

    current_portfolio_value = cash + btc_held * current_price
    portfolio_value.append(current_portfolio_value)

    if len(portfolio_value) > 1:
        daily_return = (portfolio_value[-1] - portfolio_value[-2]) /
portfolio_value[-2]
        returns.append(daily_return)

    peak_portfolio_value = max(peak_portfolio_value,
current_portfolio_value)

max_drawdown = max((peak_portfolio_value - pv) / peak_portfolio_value
for pv in portfolio_value)
volatility = np.std(returns) * np.sqrt(252)

dft['portfolio_value'] = portfolio_value
```

```
dft['benchmark_value'] = benchmark_value
dft['drawdown'] = [(peak_portfolio_value - pv) / peak_portfolio_value
for pv in portfolio_value]

plt.figure(figsize=(14, 7))
plt.plot(dft['portfolio_value'], label='Portfolio Value')
plt.plot(dft['benchmark_value'], label='Benchmark Value', color='red')
plt.title('Trading Strategy vs. Benchmark')
plt.legend()
plt.show()
print(f"Maximum Drawdown: {max_drawdown * 100:.2f}%")
print(f"Annualized Volatility: {volatility * 100:.2f}%")

0.10120205589665006
```


```
Maximum Drawdown: 93.37%
Annualized Volatility: 11.12%
```

When testing on the test set, it seems like the strategy is not really overfitting because it is performing similarly on the train set, with the maximum drawdown increase to 93.37% but volatility down to 11.12%.

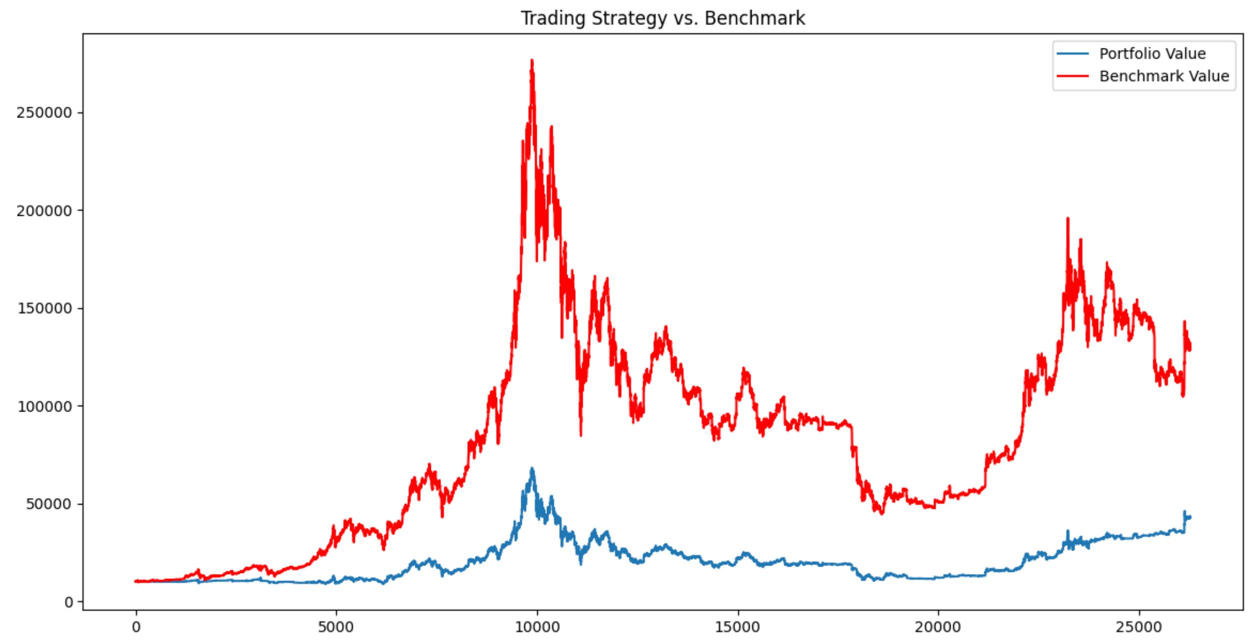
Extension

It is time to develop a more advanced signal process, where you buy and hold if the sentiment score exceeds a certain positive threshold, and sell if it falls below a certain negative threshold.

```
cash = initial_cash
btc_held = 0
```

```
peak_price = 0
portfolio_value = []
peak_portfolio_value = 0
returns = []
max_drawdown = 0
benchmark_value = initial_cash / df['listing_close'].iloc[0] *
df['listing_close']

positive_threshold =  df['sentiment'].quantile(0.5)
negative_threshold =  df['sentiment'].quantile(0.1)

def execute_trade(cash, btc_held, action, current_price):
    if action == 'buy':
        btc_held = cash / current_price
        cash = 0
    elif action == 'sell':
        cash = btc_held * current_price
        btc_held = 0
    return cash, btc_held

for index, row in df.iterrows():
    current_price = row['listing_close']
    if btc_held > 0:
        peak_price = max(peak_price, current_price)
    if btc_held > 0 and (current_price <= peak_price *
stop_loss_percent or row['sentiment'] <= negative_threshold):
        cash, btc_held = execute_trade(cash, btc_held, 'sell',
current_price)
        peak_price = 0
    elif row['sentiment'] > positive_threshold and cash > 0:
        cash, btc_held = execute_trade(cash, btc_held, 'buy',
current_price)
        peak_price = current_price

    current_portfolio_value = cash + btc_held * current_price
    portfolio_value.append(current_portfolio_value)

    if len(portfolio_value) > 1:
        daily_return = (portfolio_value[-1] - portfolio_value[-2]) /
portfolio_value[-2]
        returns.append(daily_return)

    peak_portfolio_value = max(peak_portfolio_value,
current_portfolio_value)
    current_drawdown = (peak_portfolio_value -
current_portfolio_value) / peak_portfolio_value
    max_drawdown = max(max_drawdown, current_drawdown)

volatility = np.std(returns) * np.sqrt(252)
```

```
df['portfolio_value'] = portfolio_value
df['benchmark_value'] = benchmark_value
df['drawdown'] = [(peak_portfolio_value - pv) / peak_portfolio_value
for pv in portfolio_value]

plt.figure(figsize=(14, 7))
plt.plot(df['portfolio_value'], label='Portfolio Value')
plt.plot(df['benchmark_value'], label='Benchmark Value', color='red')
plt.title('Trading Strategy vs. Benchmark')
plt.legend()
plt.show()

print(f"Maximum Drawdown: {max_drawdown * 100:.2f}%")
print(f"Annualized Volatility: {volatility * 100:.2f}%")
```




```
Maximum Drawdown: 84.65%
Annualized Volatility: 13.86%
```

By manipulating the positive and negative thresholds, we could make some different investing policies. And the threshold could also be dynamic where it could be dependent on some external factors like the inflation rates or currency exchange rates.

Conclusion

Unlike any other trading algorithms dealing with pairs trading, moving average crossovers, or momentum strategies, this trading strategy is easy to understand and to execute. It accomplishes its task to replicate the Bitcoin prices by focusing on a straightforward methodology that can be executed without advanced mathematical modeling. The drawback of this strategy is the diffculty to gather social media sentiment data, and categorize them into positive and negative sentiments using natural language processing. However, this strategy does have a bright future.