OCEAN Token Sentiment Analysis Challenge

ocean

Nicolas Landry

2023 - 14 - 06

Table of contents

Introduction

Cryptocurrencies have revolutionized the financial landscape, presenting investors with exciting opportunities and challenges. As the digital currency market continues to evolve, it becomes imperative to understand the dynamics that influence investor sentiment. In this data analysis report, we delve into the realm of sentiment analysis, focusing specifically on the OCEAN token, and aim to provide valuable insights into investor sentiment towards this cryptocurrency.

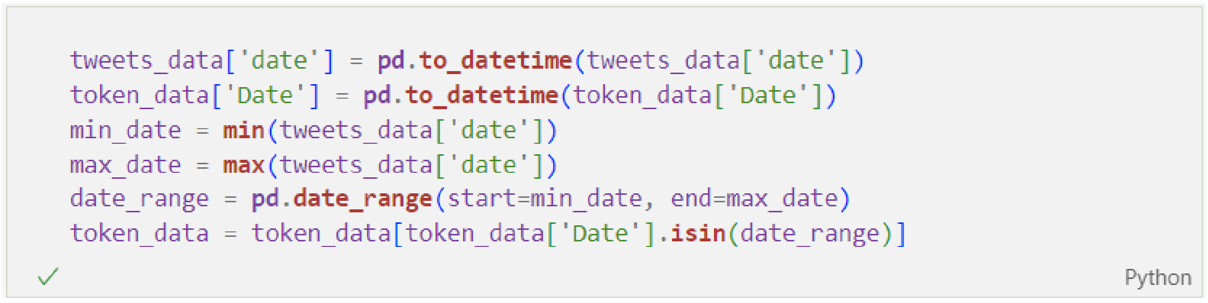
Social media platforms have become reservoirs of opinion and information about crypto-currencies. Twitter, in particular, allows investors to discuss their views, share insightful analysis and express their emotions about the market. The continuous flow of market-related data through tweets offers the opportunity to extract meaningful information.

The objective of this report is as follows. We aim to accurately assess the impact of investor sentiment on the value and popularity of the $OCEAN through a comprehensive analysis of tweet data. Understanding these sentiment dynamics is vital for investors seeking to effectively manage their investments and make well-informed decisions in response to market fluctuations. Our insights can benefit market participants, traders and researchers, enabling them to better understand the sentiments driving the crypto-currency market.

Methodology

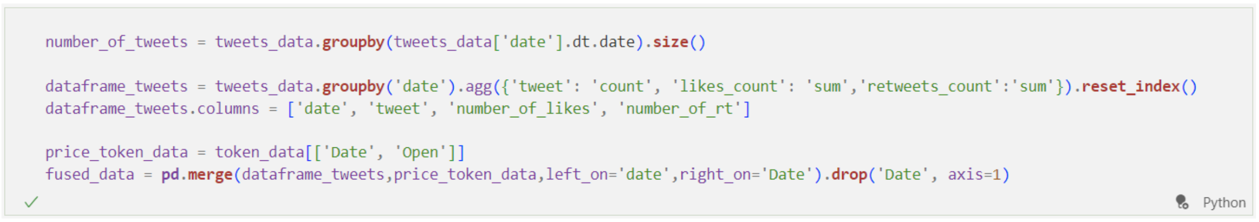
In this report we will use two datasets provided by Ocean Protocol : one containing records of all tweets featuring "$OCEAN" from 2019-12-31 to 2022-10-22 including the tweet content, user's infos, number of likes and retweets. The other contains the price history of the OCEAN token from 2019-05-06 to 2023-05-29.

Firstly, we only select data from 2019-12-31 to 2022-10-22. This time frame allows us to gather both tweet data and the corresponding price information for the $OCEAN token over the same period.



```python
tweets_data['date'] = pd.to_datetime(tweets_data['date'])
token_data['Date'] = pd.to_datetime(token_data['Date'])
min_date = min(tweets_data['date'])
max_date = max(tweets_data['date'])
date_range = pd.date_range(start=min_date, end=max_date)
token_data = token_data[token_data['Date'].isin(date_range)]
```

In our report, the fused_data will refer to a combined dataframe based on the common date variable, it includes for each day the number of tweets, number of likes, number of retweets, and the opening price of the OCEAN token.



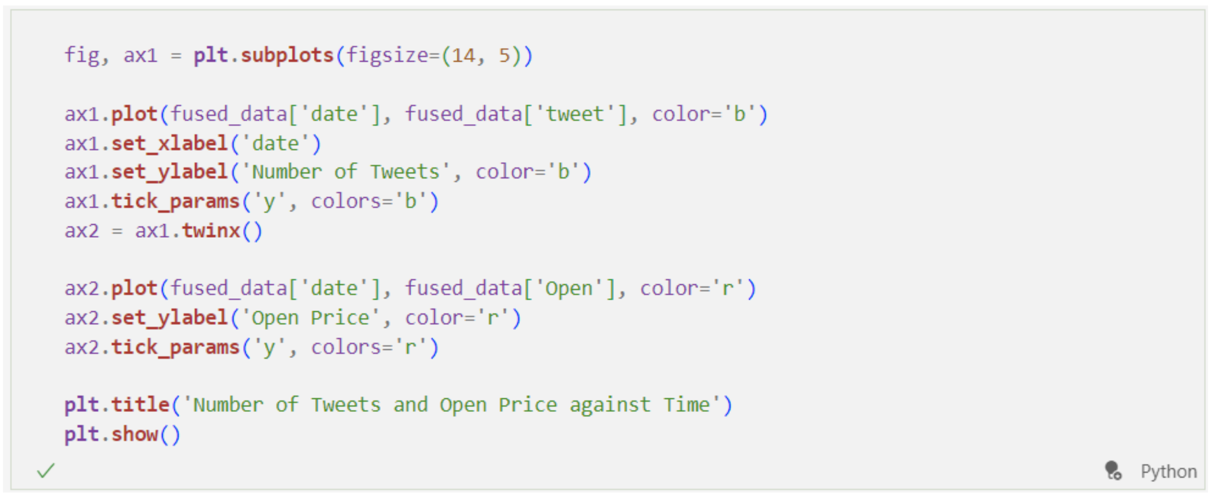
```python
number_of_tweets = tweets_data.groupby(tweets_data['date'].dt.date).size()

dataframe_tweets = tweets_data.groupby('date').agg({'tweet': 'count', 'likes_count': 'sum','retweets_count':'sum'}).reset_index()
dataframe_tweets.columns = ['date', 'tweet', 'number_of_likes', 'number_of_rt']

price_token_data = token_data[['Date', 'Open']]
fused_data = pd.merge(dataframe_tweets,price_token_data,left_on='date',right_on='Date').drop('Date', axis=1)
```

Data analysis

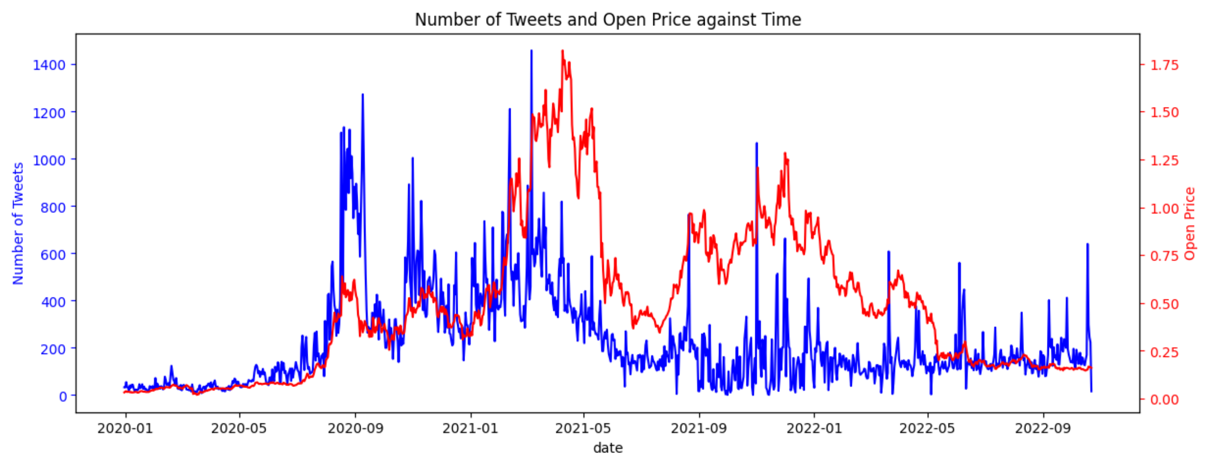
1. Correlation between the price of $OCEAN and the number of tweets containing "$OCEAN"

    a) Correlation between $OCEAN and number of tweets during the whole period

First, we display the number of tweets related to $OCEAN and the open price of $OCEAN over time to get a graphic idea of the correlation or otherwise between these two variables using the piece of code below.

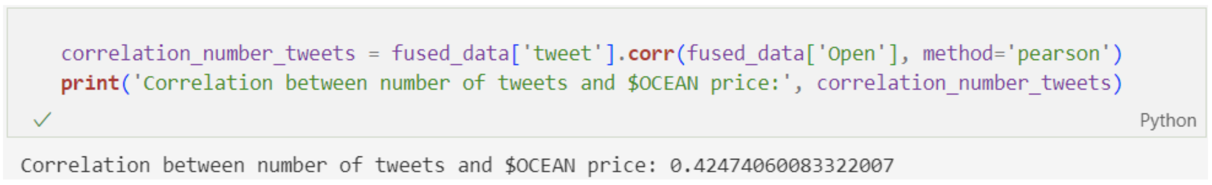


```python
fig, ax1 = plt.subplots(figsize=(14, 5))

ax1.plot(fused_data['date'], fused_data['tweet'], color='b')
ax1.set_xlabel('date')
ax1.set_ylabel('Number of Tweets', color='b')
ax1.tick_params('y', colors='b')
ax2 = ax1.twinx()

ax2.plot(fused_data['date'], fused_data['Open'], color='r')
ax2.set_ylabel('Open Price', color='r')
ax2.tick_params('y', colors='r')

plt.title('Number of Tweets and Open Price against Time')
plt.show()
```



The graph above shows that the number of tweets (in blue) follows the open price curve (in red) fairly closely, particularly between 2020-01 and 2021-09. However, we can see that the curve for the number of tweets is much more spiky than that for the price.

Then, to quantify the correlation between the two variables numerically, we calculate the Pearson correlation coefficient, a statistical measure that defines the strength of the relationship between two variables and their association with each other.



```python
correlation_number_tweets = fused_data['tweet'].corr(fused_data['Open'], method='pearson')
print('Correlation between number of tweets and $OCEAN price:', correlation_number_tweets)
```

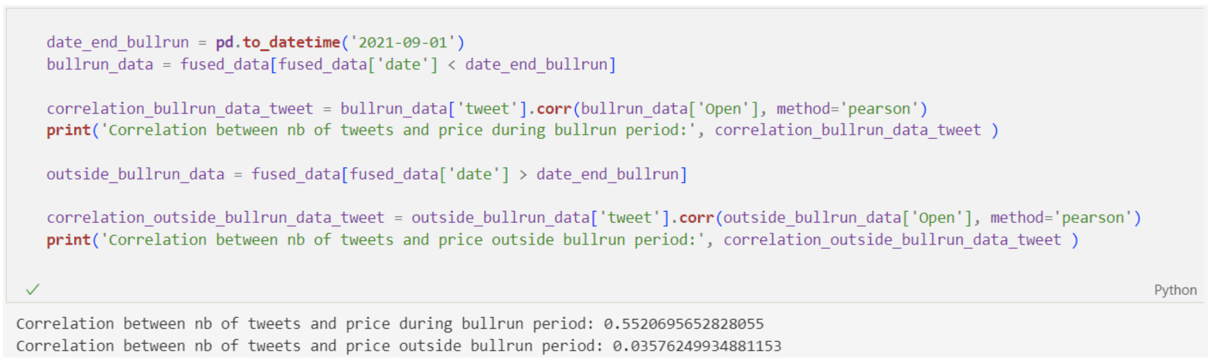
```
Correlation between number of tweets and $OCEAN price: 0.42474060083322007
```

As you can see above, we obtain a Pearson coefficient of 0.425, which corresponds to a moderate positive correlation.

b) Correlation between $OCEAN and number of tweet during the “Bullrun”

As we noted earlier on the graph, it seems that the price and the number of tweets follow a very similar trend over the period 2020-01 and 2021-09. This period corresponds to the Bullrun A bull run in cryptocurrencies refers to a sustained period of significant price increases across the market, often accompanied by widespread investor optimism and positive market sentiment.

2020 and 2021 until around September/November were inside the bullrun.
Thus, we will examine the correlation between the number of OCEAN-related tweets during this period, and then outside it.



```python
date_end_bullrun = pd.to_datetime('2021-09-01')
bullrun_data = fused_data[fused_data['date'] < date_end_bullrun]

correlation_bullrun_data_tweet = bullrun_data['tweet'].corr(bullrun_data['Open'], method='pearson')
print('Correlation between nb of tweets and price during bullrun period:', correlation_bullrun_data_tweet )

outside_bullrun_data = fused_data[fused_data['date'] > date_end_bullrun]

correlation_outside_bullrun_data_tweet = outside_bullrun_data['tweet'].corr(outside_bullrun_data['Open'], method='pearson')
print('Correlation between nb of tweets and price outside bullrun period:', correlation_outside_bullrun_data_tweet )
```

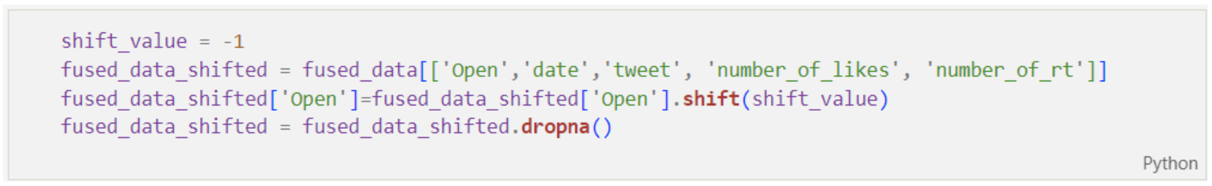
```
Correlation between nb of tweets and price during bullrun period: 0.5520695652828055
Correlation between nb of tweets and price outside bullrun period: 0.035762499348811 53
```

For the bullrun period, we obtain a pearson correlation coefficient of 0.552, whereas after the bullrun period, we have a coefficient of 0.036.
Therefore, there is a fairly significant difference of 0.516 between the two periods. We deduce that it is because in the Bullrun period the market price becomes very influenced by the general enthusiasm on projects and people are more prone to follow others based on sentiment. When it stops or starts to slow down, a lot of people stop following the crypto market and tweeting about it, especially if they lost money.

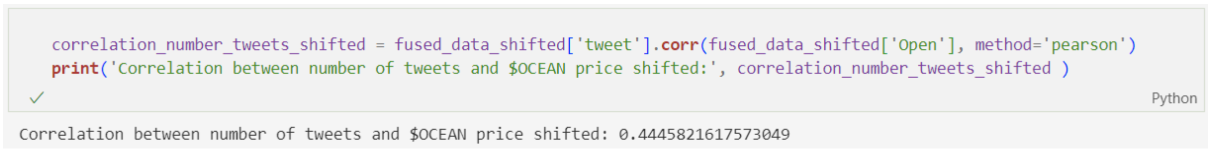
c) Correlation between $OCEAN and number of tweet shifted

After calculating these correlations, we wondered whether it was the price of the OCEAN token that influenced the number of community tweets, or vice versa.

In the first case, we will shift the price curve by -1 day. In other words, we want to show the correlation between the price on day D-1 and the number of tweets on D.



```python
shift_value = -1
fused_data_shifted = fused_data[['Open','date','tweet', 'number_of_likes', 'number_of_rt']]
fused_data_shifted['Open']=fused_data_shifted['Open'].shift(shift_value)
fused_data_shifted = fused_data_shifted.dropna()
```

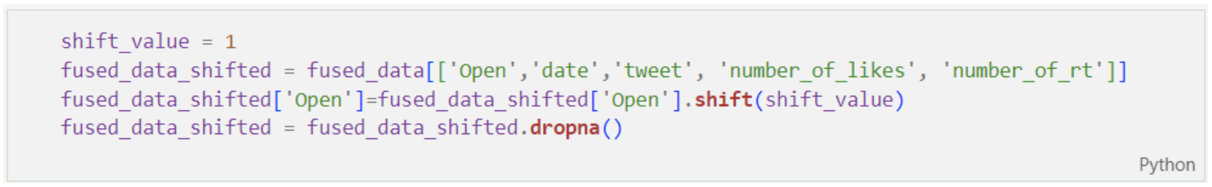


```python
correlation_number_tweets_shifted = fused_data_shifted['tweet'].corr(fused_data_shifted['Open'], method='pearson')
print('Correlation between number of tweets and $OCEAN price shifted:', correlation_number_tweets_shifted )
```

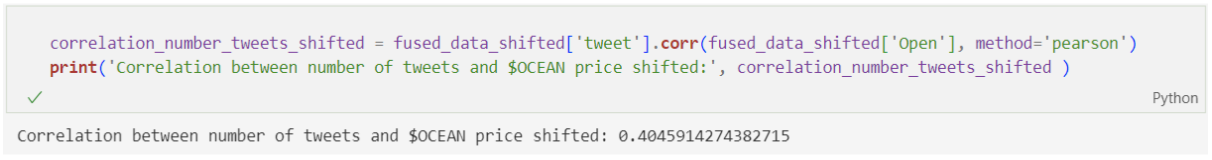
```
Correlation between number of tweets and $OCEAN price shifted: 0.4445821617573049
```

As you can see above, we obtain a Pearson coefficient of 0.445,

In the second case, we will shift the price curve by + 1 day. In other words, we want to show the correlation between the number of tweets on day D and the price on D+1.



```python
shift_value = 1
fused_data_shifted = fused_data[['Open','date','tweet', 'number_of_likes', 'number_of_rt']]
fused_data_shifted['Open']=fused_data_shifted['Open'].shift(shift_value)
fused_data_shifted = fused_data_shifted.dropna()
```



```python
correlation_number_tweets_shifted = fused_data_shifted['tweet'].corr(fused_data_shifted['Open'], method='pearson')
print('Correlation between number of tweets and $OCEAN price shifted:', correlation_number_tweets_shifted )
```

```
Correlation between number of tweets and $OCEAN price shifted: 0.4045914274382715
```

We obtain a Pearson coefficient of 0.405.

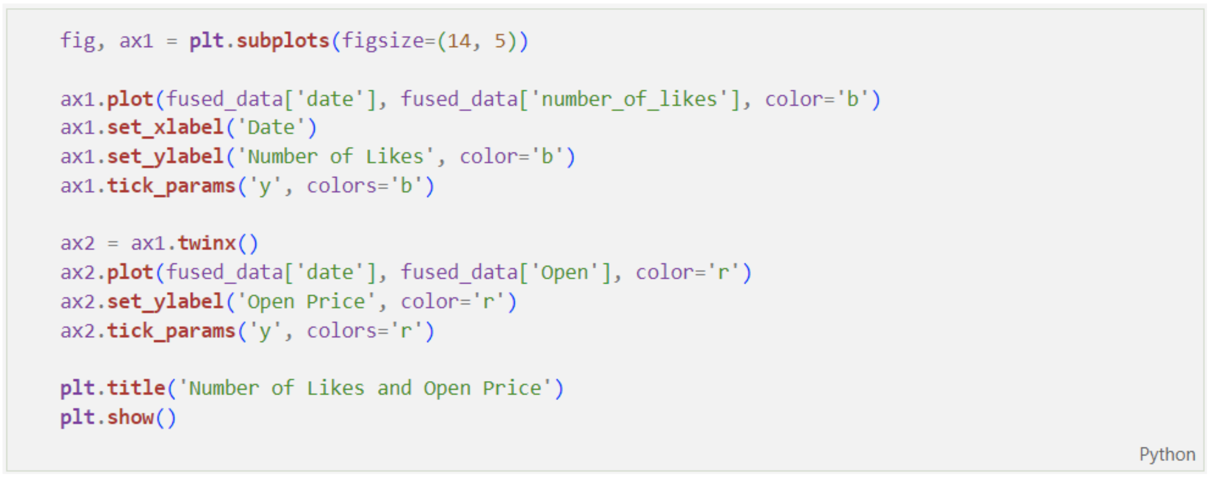
From the coefficients obtained, we can see that the correlation rate between number of tweets and price increases in the first case and decreases in the second. Even if the difference with the unshifted correlation is not huge, given the size of our dataset, this gives us the following insight : the price is more likely to influence the number of tweets.
To obtain more precise figures, it would be interesting to analyze hour-by-hour data, so that comparisons can be made with a shift of less than a day.

2. Correlation between the price of $OCEAN and the number of likes received by tweets containing "$OCEAN"

For this question, we will proceed in the same way as the previous one, to ensure consistency in our analysis.

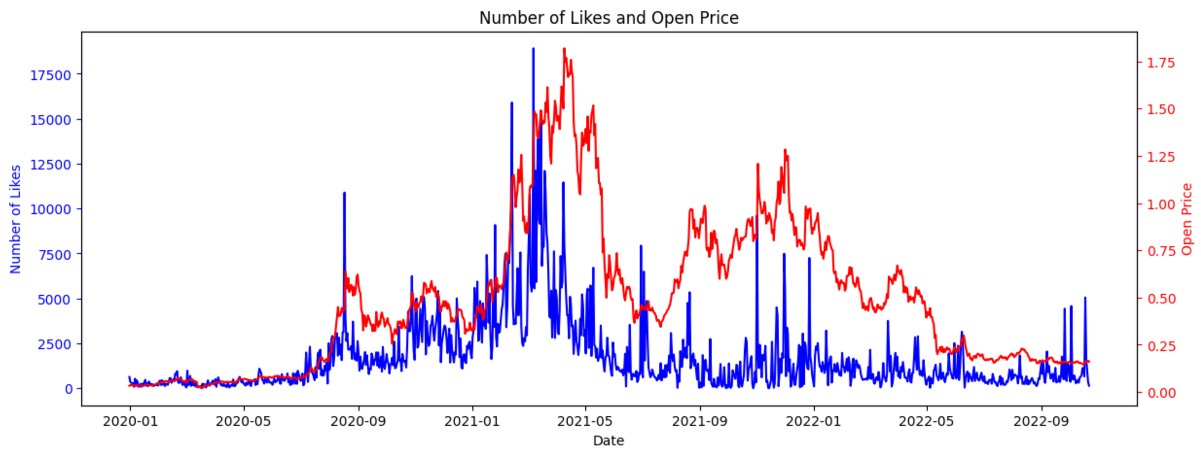
a) Correlation between $OCEAN and number of likes during the whole period

First, we visualize the number of likes on tweets about OCEAN, as well as the price of the OCEAN token over time.



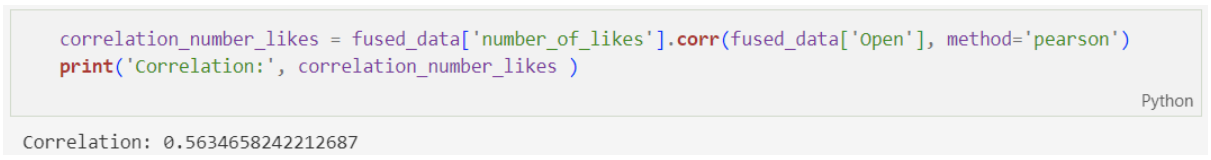
```python
fig, ax1 = plt.subplots(figsize=(14, 5))

ax1.plot(fused_data['date'], fused_data['number_of_likes'], color='b')
ax1.set_xlabel('Date')
ax1.set_ylabel('Number of Likes', color='b')
ax1.tick_params('y', colors='b')

ax2 = ax1.twinx()
ax2.plot(fused_data['date'], fused_data['Open'], color='r')
ax2.set_ylabel('Open Price', color='r')
ax2.tick_params('y', colors='r')

plt.title('Number of Likes and Open Price')
plt.show()
```

Python



Thanks to the chart above, we can see that the number of likes (in blue) and the open price (in red) follow a similar trend, and this is even more noticeable in the period between 2020-01 and 2021-09, which we previously defined as the last Bullrun.

Then, we calculate the Pearson correlation coefficient to obtain a numerical result.



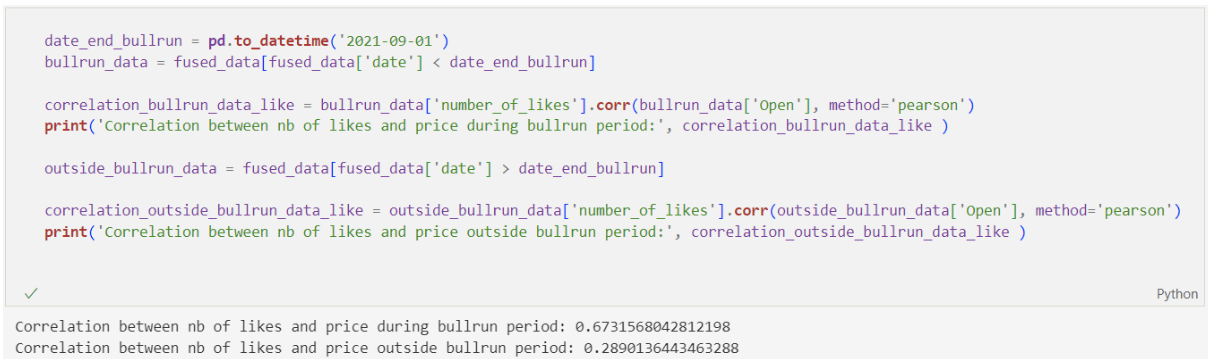
```python
correlation_number_likes = fused_data['number_of_likes'].corr(fused_data['Open'], method='pearson')
print('Correlation:', correlation_number_likes )
```

```
Correlation: 0.5634658242212687
```

We have a correlation coefficient of 0.563, which corresponds to a fairly high positive correlation.

b) Correlation between $OCEAN and number of likes during the “Bullrun”

As already mentioned, the like number and price curves are more similar in the first period. Hence we will now calculate the correlation coefficient for the two periods and compare it with the coefficient obtained for the whole period.



```python
date_end_bullrun = pd.to_datetime('2021-09-01')
bullrun_data = fused_data[fused_data['date'] < date_end_bullrun]

correlation_bullrun_data_like = bullrun_data['number_of_likes'].corr(bullrun_data['Open'], method='pearson')
print('Correlation between nb of likes and price during bullrun period:', correlation_bullrun_data_like )

outside_bullrun_data = fused_data[fused_data['date'] > date_end_bullrun]

correlation_outside_bullrun_data_like = outside_bullrun_data['number_of_likes'].corr(outside_bullrun_data['Open'], method='pearson')
print('Correlation between nb of likes and price outside bullrun period:', correlation_outside_bullrun_data_like )
```

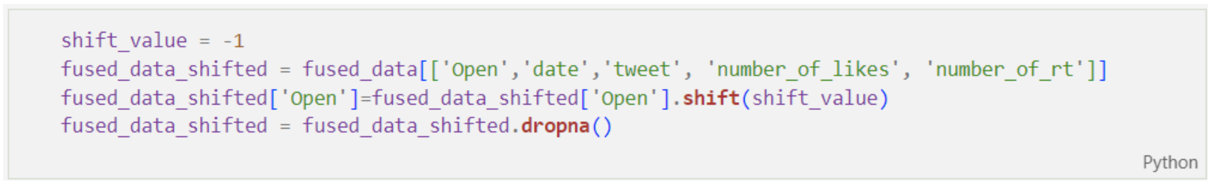
```
Correlation between nb of likes and price during bullrun period: 0.6731568042812198
Correlation between nb of likes and price outside bullrun period: 0.2890136443463288
```

We therefore have a correlation of 0.673 until 2021-09 versus 0.289 for the period after and 0.563 for the full period. Clearly, the correlation is higher in the first part, which corresponds to the Bullrun period.

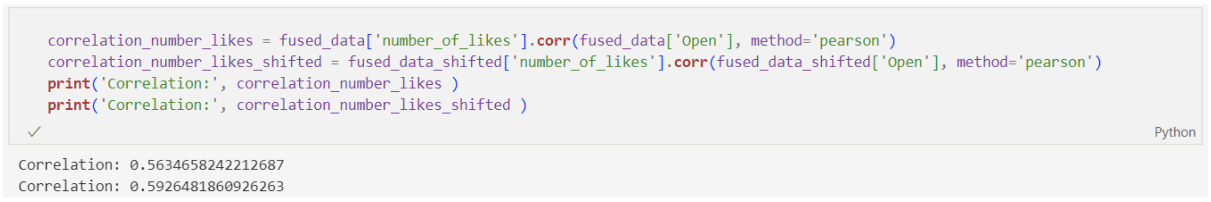
c) Correlation between $OCEAN and number of likes shifted

To understand whether it's the price that influences the number of likes, or vice versa, we're going to shift the two variables by one day.

First, we will look at the number of likes on day D with the price of the OCEAN token on day D-1.



```python
shift_value = -1
fused_data_shifted = fused_data[['Open','date','tweet', 'number_of_likes', 'number_of_rt']]
fused_data_shifted['Open']=fused_data_shifted['Open'].shift(shift_value)
fused_data_shifted = fused_data_shifted.dropna()
```

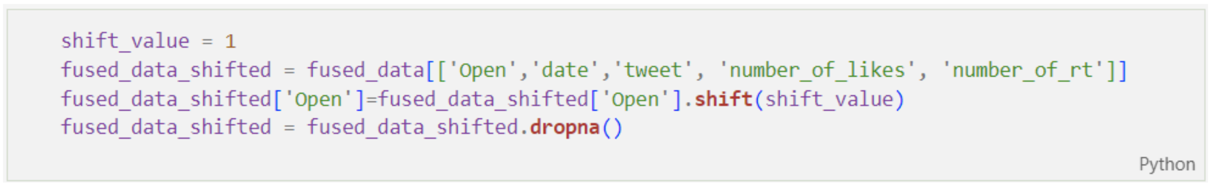


```python
correlation_number_likes = fused_data['number_of_likes'].corr(fused_data['Open'], method='pearson')
correlation_number_likes_shifted = fused_data_shifted['number_of_likes'].corr(fused_data_shifted['Open'], method='pearson')
print('Correlation:', correlation_number_likes )
print('Correlation:', correlation_number_likes_shifted )
```

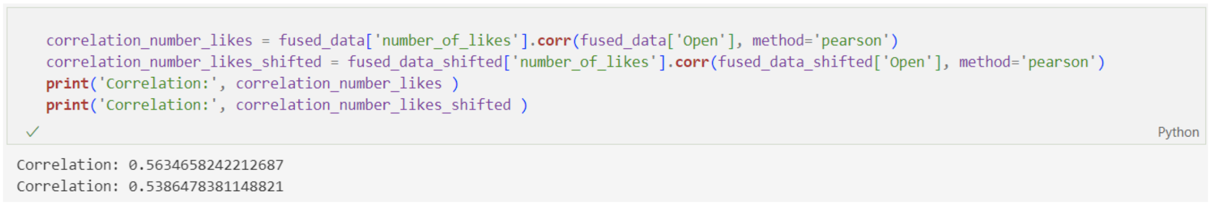
```
Correlation: 0.5634658242212687
Correlation: 0.5926481860926263
```

This gives a correlation of 0.593, which is greater than the unshifted correlation de 0.563

Next, we will examine the relationship between the like count on day D and the price of the OCEAN token on day D+1.



```python
shift_value = 1
fused_data_shifted = fused_data[['Open','date','tweet', 'number_of_likes', 'number_of_rt']]
fused_data_shifted['Open']=fused_data_shifted['Open'].shift(shift_value)
fused_data_shifted = fused_data_shifted.dropna()
```



```python
correlation_number_likes = fused_data['number_of_likes'].corr(fused_data['Open'], method='pearson')
correlation_number_likes_shifted = fused_data_shifted['number_of_likes'].corr(fused_data_shifted['Open'], method='pearson')
print('Correlation:', correlation_number_likes )
print('Correlation:', correlation_number_likes_shifted )
```

```
Correlation: 0.5634658242212687
Correlation: 0.5386478381148821
```

This gives a correlation of 0.539, which is lower than the unshifted correlation.

Comparing the correlation coefficients obtained, we observe that the correlation is higher between the number of likes on day D and Ocean's price on the previous day. We can deduce from this that the number of likes influences the price of OCEAN, and not vice versa.

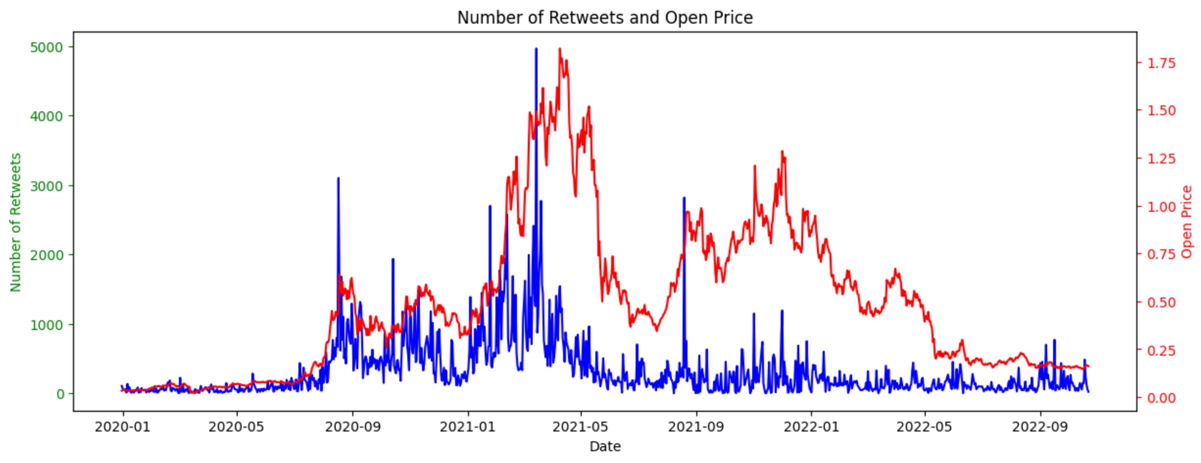
3. Correlation between the price of $OCEAN and the number of retweets generated by tweets containing "$OCEAN"

For this question, we will proceed in the same way as before, to remain consistent.

a) Correlation between $OCEAN and number of retweets during the whole period

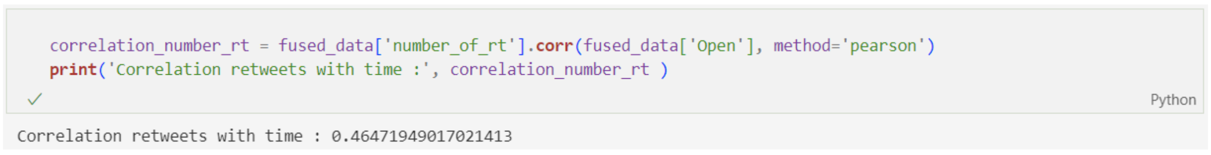
We start by plotting the number of OCEAN-related retweets against the price of the OCEAN token over time.

```python
fig, ax1 = plt.subplots(figsize=(14, 5))

ax1.plot(fused_data['date'], fused_data['number_of_rt'], color='b')
ax1.set_xlabel('Date')
ax1.set_ylabel('Number of Retweets', color='g')
ax1.tick_params('y', colors='g')

ax2 = ax1.twinx()
ax2.plot(fused_data['date'], fused_data['Open'], color='r')
ax2.set_ylabel('Open Price', color='r')
ax2.tick_params('y', colors='r')

plt.title('Number of Retweets and Open Price')
plt.show()
```
Python



The resulting graph shows us that the number of retweets (in blue) is somewhat similar to the price curve (in red), especially before 2021-09.

To quantify the relationship between these two variables, let's calculate Pearson's correlation coefficient.



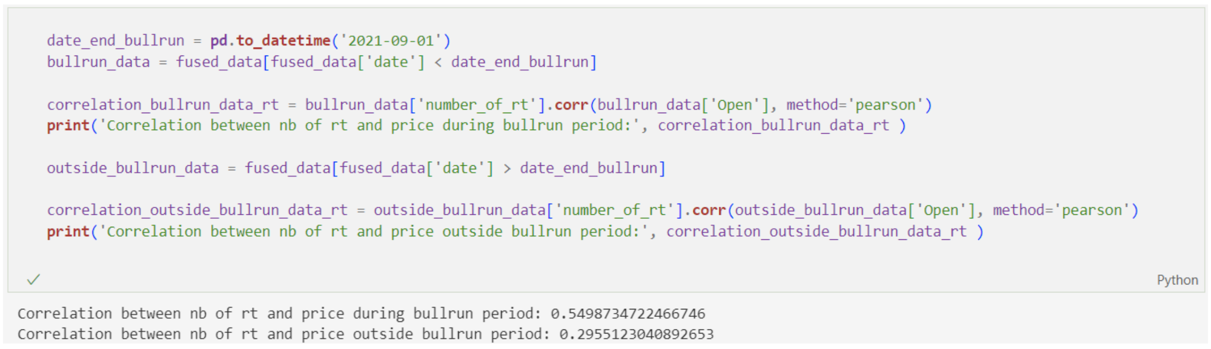
```python
correlation_number_rt = fused_data['number_of_rt'].corr(fused_data['Open'], method='pearson')
print('Correlation retweets with time :', correlation_number_rt )
```

```
Correlation retweets with time : 0.46471949017021413
```

We have a correlation relationship of 0.465, which corresponds to a moderate positive correlation between the two variables.

b) Correlation between $OCEAN and number of retweets during the "Bullrun"

As we saw in the previous section, the number of retweets seems to correspond more to the price from 2020-01 to 2021-09. We will therefore quantify the correlation between these two data during and outside this period.



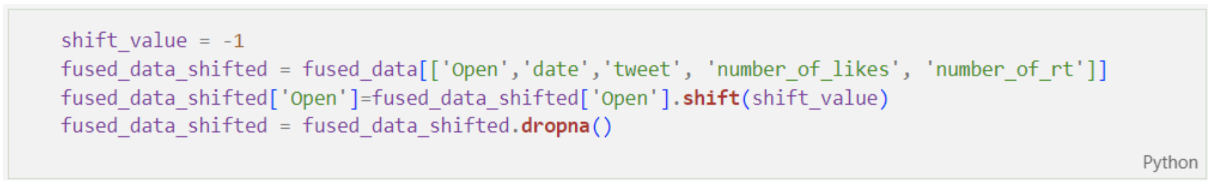
```python
date_end_bullrun = pd.to_datetime('2021-09-01')
bullrun_data = fused_data[fused_data['date'] < date_end_bullrun]

correlation_bullrun_data_rt = bullrun_data['number_of_rt'].corr(bullrun_data['Open'], method='pearson')
print('Correlation between nb of rt and price during bullrun period:', correlation_bullrun_data_rt )

outside_bullrun_data = fused_data[fused_data['date'] > date_end_bullrun]

correlation_outside_bullrun_data_rt = outside_bullrun_data['number_of_rt'].corr(outside_bullrun_data['Open'], method='pearson')
print('Correlation between nb of rt and price outside bullrun period:', correlation_outside_bullrun_data_rt )
```

```
Correlation between nb of rt and price during bullrun period: 0.5498734722466746
Correlation between nb of rt and price outside bullrun period: 0.2955123040892653
```

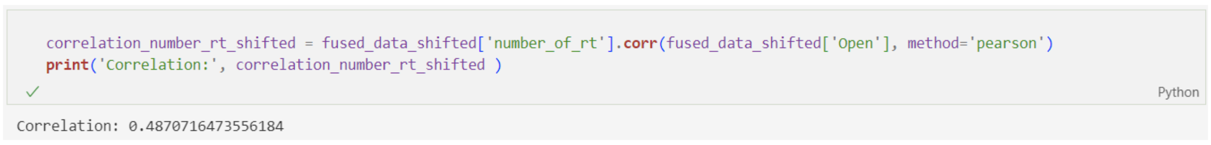
We have a correlation of 0.550 for the first period, against 0.295 for the second. Once again, we can see that the correlation is higher in the first part, corresponding to the Bullrun.

c) Correlation between $OCEAN and number of retweets shifted

To find out whether it's the number of retweets that impacts the price of the $OCEAN token, or vice versa, we study the correlation between these two variables with a one-day shift. Firstly, the correlation between the price on day D and the number of retweets on day D+1.



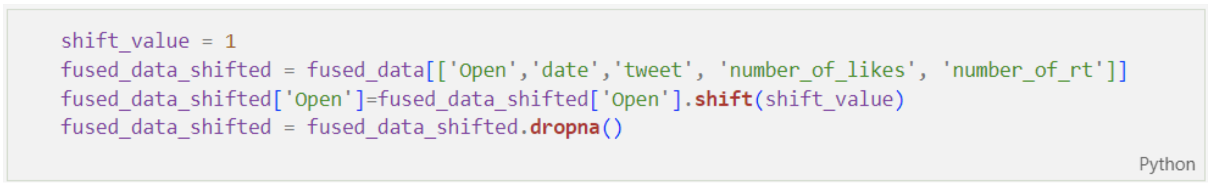
```python
shift_value = -1
fused_data_shifted = fused_data[['Open','date','tweet', 'number_of_likes', 'number_of_rt']]
fused_data_shifted['Open']=fused_data_shifted['Open'].shift(shift_value)
fused_data_shifted = fused_data_shifted.dropna()
```



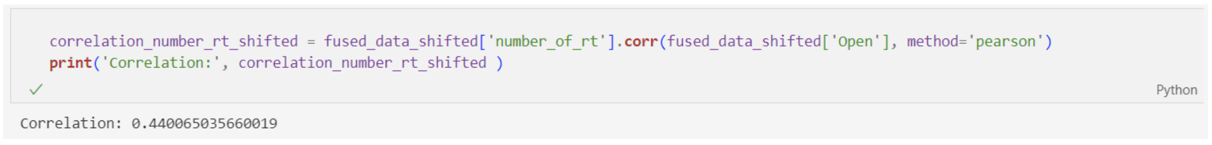
```python
correlation_number_rt_shifted = fused_data_shifted['number_of_rt'].corr(fused_data_shifted['Open'], method='pearson')
print('Correlation:', correlation_number_rt_shifted )
```

```
Correlation: 0.4870716473556184
```

Then, the correlation between the price on day D and the number of retweets on day D-1.



```python
shift_value = 1
fused_data_shifted = fused_data[['Open','date','tweet', 'number_of_likes', 'number_of_rt']]
fused_data_shifted['Open']=fused_data_shifted['Open'].shift(shift_value)
fused_data_shifted = fused_data_shifted.dropna()
```



```python
correlation_number_rt_shifted = fused_data_shifted['number_of_rt'].corr(fused_data_shifted['Open'], method='pearson')
print('Correlation:', correlation_number_rt_shifted )
```

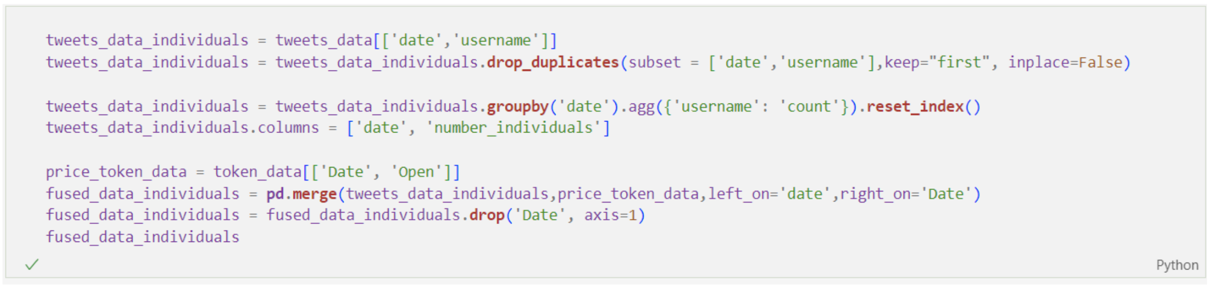
```
Correlation: 0.440065035660019
```

The coefficients obtained show that the correlation rate between number of retweets and price increases in the first case (0.487) and decreases in the second (0.440). It gives us the following idea: price is more likely to influence the number of retweets.

4. Correlation between the price of $OCEAN and the number of individuals tweeting with the cashtag "$OCEAN"

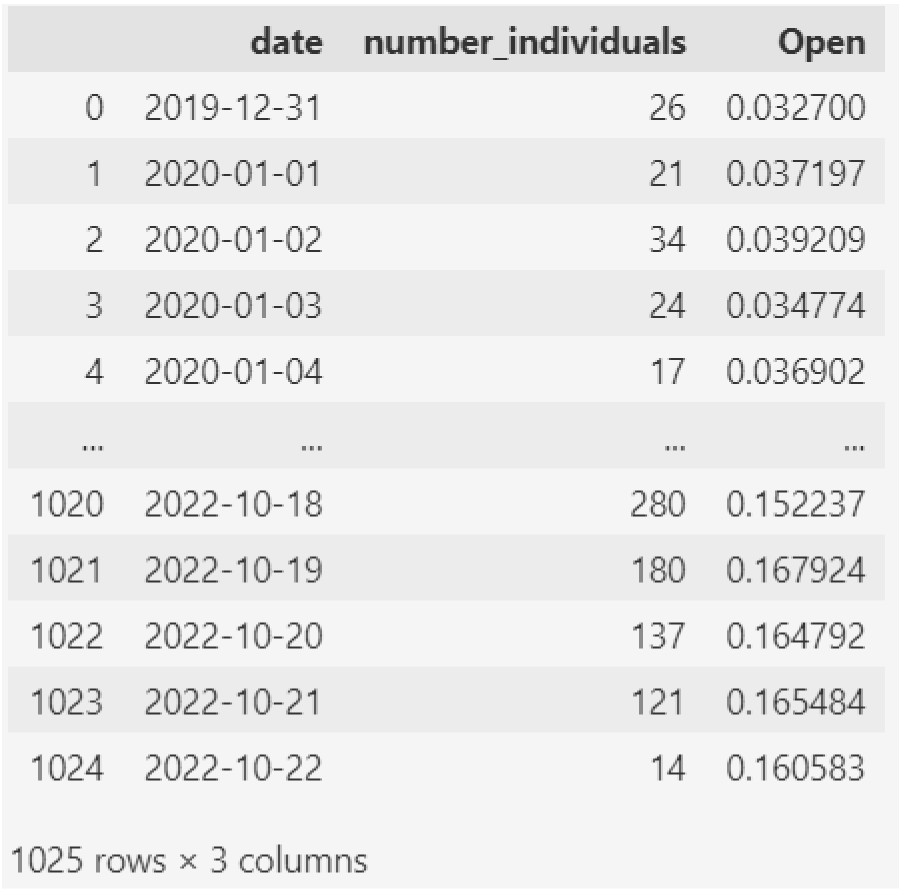
To analyze the $OCEAN price as a function of the number of individuals tweeting per day, we first kept the date and username columns. Then, since the username is unique on Tweeter, we remove duplicates from the username column. We add a number_individuals column that sums up the number of usernames per day.

Finally, we merge the date and number_individuals columns with an Open column for the token price, to obtain the following table.



```python
tweets_data_individuals = tweets_data[['date','username']]
tweets_data_individuals = tweets_data_individuals.drop_duplicates(subset = ['date','username'],keep="first", inplace=False)

tweets_data_individuals = tweets_data_individuals.groupby('date').agg({'username': 'count'}).reset_index()
tweets_data_individuals.columns = ['date', 'number_individuals']

price_token_data = token_data[['Date', 'Open']]
fused_data_individuals = pd.merge(tweets_data_individuals,price_token_data,left_on='date',right_on='Date')
fused_data_individuals = fused_data_individuals.drop('Date', axis=1)
fused_data_individuals
```

✓ Python



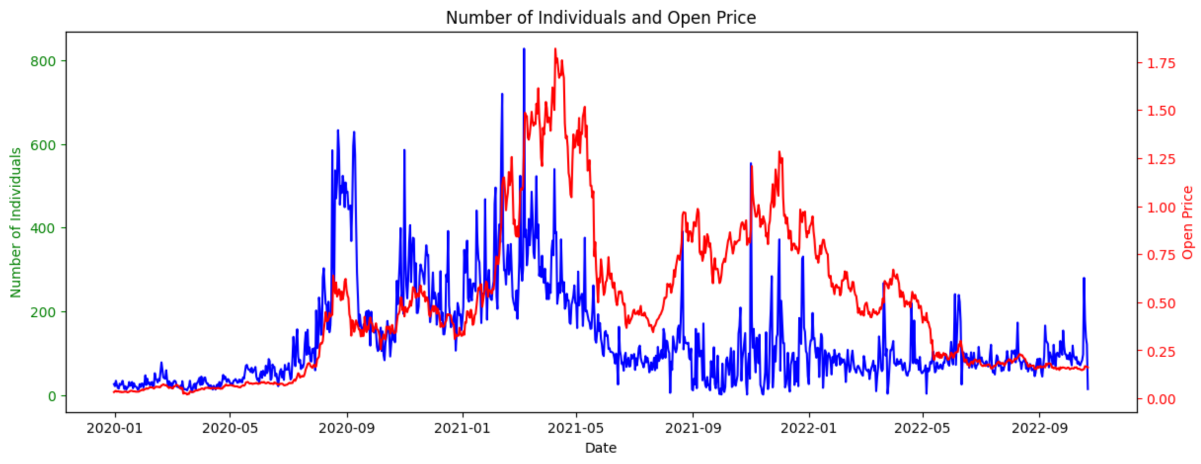
	date	number_individuals	Open
0	2019-12-31	26	0.032700
1	2020-01-01	21	0.037197
2	2020-01-02	34	0.039209
3	2020-01-03	24	0.034774
4	2020-01-04	17	0.036902
...	...	...	...
1020	2022-10-18	280	0.152237
1021	2022-10-19	180	0.167924
1022	2022-10-20	137	0.164792
1023	2022-10-21	121	0.165484
1024	2022-10-22	14	0.160583

1025 rows × 3 columns

a) Correlation between $OCEAN and number of individuals tweeting during the whole period

Now we display the number of individuals per day and the price of $OCEAN against time to get a visual representation.

```python
fig, ax1 = plt.subplots(figsize=(14, 5))

ax1.plot(fused_data_individuals['date'], fused_data_individuals['number_individuals'], color='b')
ax1.set_xlabel('Date')
ax1.set_ylabel('Number of Individuals', color='g')
ax1.tick_params('y', colors='g')

ax2 = ax1.twinx()
ax2.plot(fused_data_individuals['date'], fused_data_individuals['Open'], color='r')
ax2.set_ylabel('Open Price', color='r')
ax2.tick_params('y', colors='r')

plt.title('Number of Individuals and Open Price')
plt.show()
```



We can see that the two curves seem to follow a similar trend, and even more so before 2021-09. But we can also see that the individual curve (in blue) has many peaks.

Let's verify this observation by calculating the Pearson correlation coefficient.

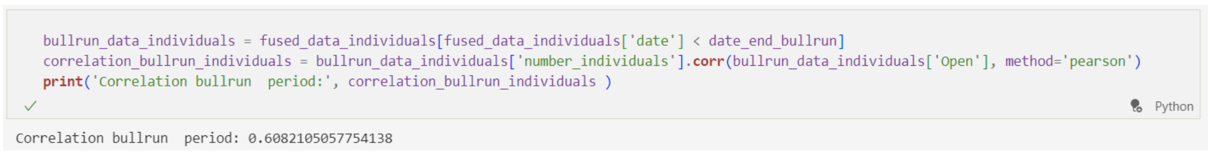
```python
correlation_number_individuals = fused_data_individuals['number_individuals'].corr(fused_data_individuals['Open'], method='pearson')
print('Correlation between number of individuals and price :', correlation_number_individuals )
```

```
Correlation between number of individuals and price : 0.479836770029259
```

We have a coefficient of 0.480, which corresponds to a moderate positive correlation.

b) Correlation between $OCEAN and number of individuals tweeting during the Bullrun

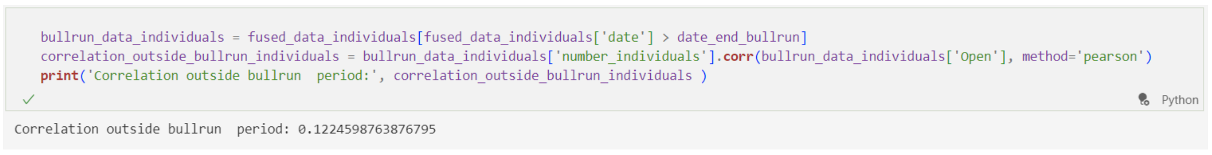
In the same way as before, we'll look at the difference in correlation between the period before 2021-09 and after.



```
bullrun_data_individuals = fused_data_individuals[fused_data_individuals['date'] < date_end_bullrun]
correlation_bullrun_individuals = bullrun_data_individuals['number_individuals'].corr(bullrun_data_individuals['Open'], method='pearson')
print('Correlation bullrun  period:', correlation_bullrun_individuals )
```

Python

```
Correlation bullrun  period: 0.6082105057754138
```



```
bullrun_data_individuals = fused_data_individuals[fused_data_individuals['date'] > date_end_bullrun]
correlation_outside_bullrun_individuals = bullrun_data_individuals['number_individuals'].corr(bullrun_data_individuals['Open'], method='pearson')
print('Correlation outside bullrun  period:', correlation_outside_bullrun_individuals )
```

Python

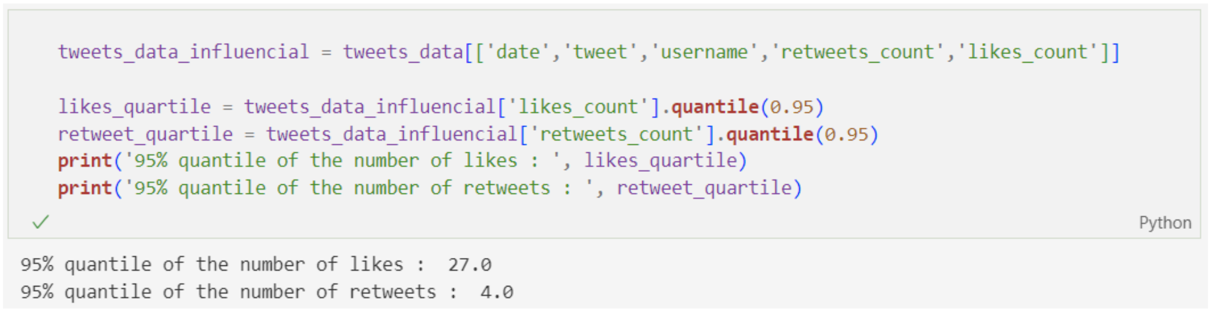
```
Correlation outside bullrun  period: 0.1224598763876795
```

For the bullrun period, we obtain a pearson correlation coefficient of 0.608, whereas after the bullrun period, we have a coefficient of 0.122. Therefore, there is a fairly significant difference of 0.486 between the two periods.

5. Impact of influential tweets on the price of the OCEAN token

To answer this question, we will define an influential tweet as one that has received more likes and retweets than the 95% quantile of likes and retweets of all tweets.

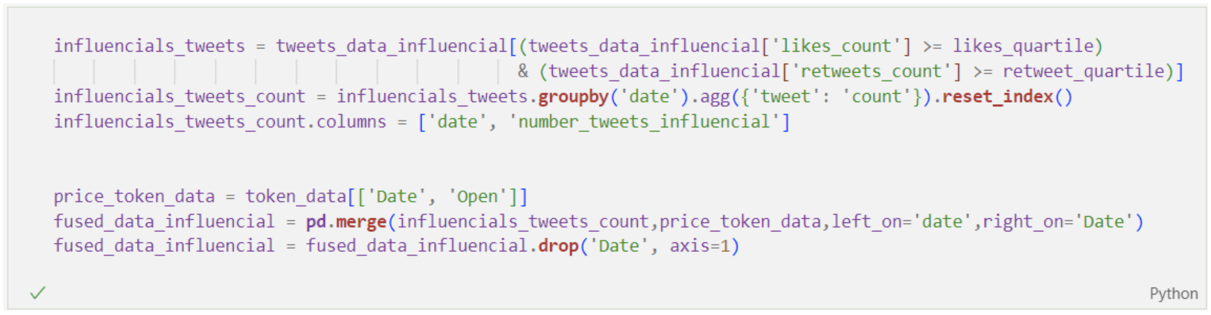
To start with, we create a table with the date, tweets, username, retweets count and likes count. Then we calculate the 95% quantile of the number of likes and retweets.



```python
tweets_data_influencial = tweets_data[['date','tweet','username','retweets_count','likes_count']]

likes_quartile = tweets_data_influencial['likes_count'].quantile(0.95)
retweet_quartile = tweets_data_influencial['retweets_count'].quantile(0.95)
print('95% quantile of the number of likes : ', likes_quartile)
print('95% quantile of the number of retweets : ', retweet_quartile)
```

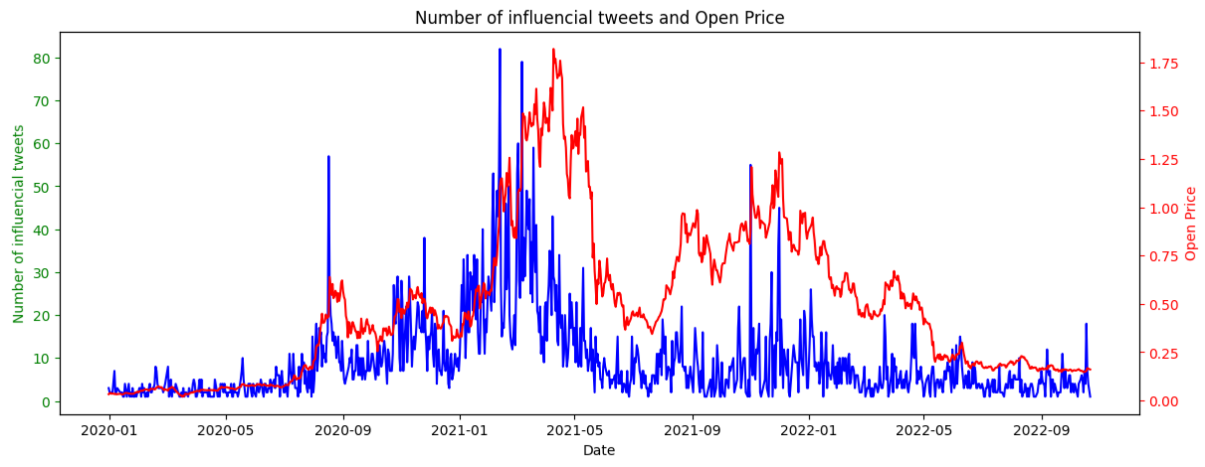
```
95% quantile of the number of likes :  27.0
95% quantile of the number of retweets :  4.0
```

An influential tweet is a tweet that has received more than 27 likes and more than 4 retweets.
We sort to keep only the influential tweets, then add a column that counts the number of influential tweets per day. We sort to keep only the influential tweets, then add a column that counts the number of influential tweets per day. Finally, we create the fused_data_influencials table, adding a column with the price of the OCEAN token.

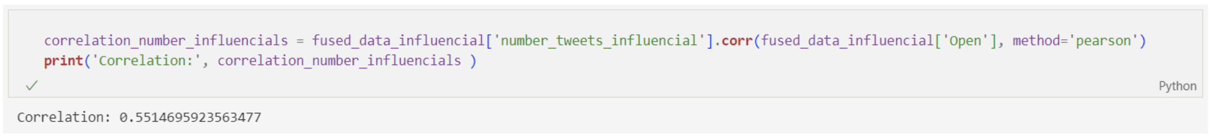


```python
influencials_tweets = tweets_data_influencial[(tweets_data_influencial['likes_count'] >= likes_quartile)
                                               & (tweets_data_influencial['retweets_count'] >= retweet_quartile)]
influencials_tweets_count = influencials_tweets.groupby('date').agg({'tweet': 'count'}).reset_index()
influencials_tweets_count.columns = ['date', 'number_tweets_influencial']


price_token_data = token_data[['Date', 'Open']]
fused_data_influencial = pd.merge(influencials_tweets_count,price_token_data,left_on='date',right_on='Date')
fused_data_influencial = fused_data_influencial.drop('Date', axis=1)
```

We then produce a graph showing the number of tweets and the price over time.



The two curves follow a similar trend as a function of time. Let's calculate the correlation coefficient.



```
correlation_number_influencials = fused_data_influencial['number_tweets_influencial'].corr(fused_data_influencial['Open'], method='pearson')
print('Correlation:', correlation_number_influencials )
```

```
Correlation: 0.5514695923563477
```

We obtain a correlation of 0.551 for the number of influential tweets and the price of $OCEAN, which corresponds to a fairly high positive correlation.

From our analysis, it is evident that influential tweets have one of the highest correlations with the price of the OCEAN token compared to other factors we examined. The correlation coefficient of 0.551 indicates a fairly strong positive correlation. This suggests that influential tweets play a significant role in impacting the price of the OCEAN token.

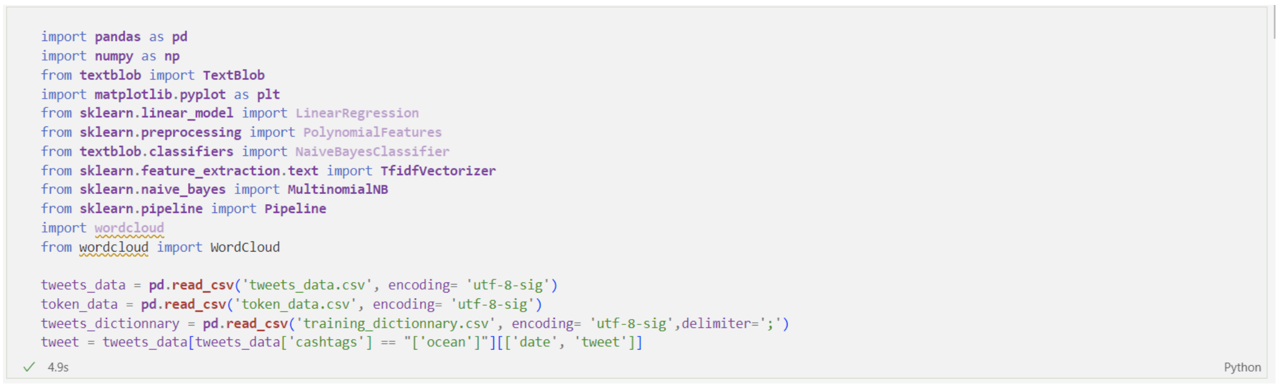
These findings imply that when influential individuals or news events generate tweets related to OCEAN, there is a notable effect on the token's price. It highlights the importance of monitoring and analyzing influential social media activity within the OCEAN community, as it can potentially influence market trends and investor sentiment.

By paying attention to these influential tweets, market participants can gain valuable insights and potentially make more informed decisions regarding their OCEAN holdings.

Machine learning

1. Main model

This code implements a machine learning model for sentiment analysis of tweets, classifying them as bullish, bearish, or neutral to see the repartition of those tweets and what they usually contain. The model utilizes various libraries and techniques to perform this task.

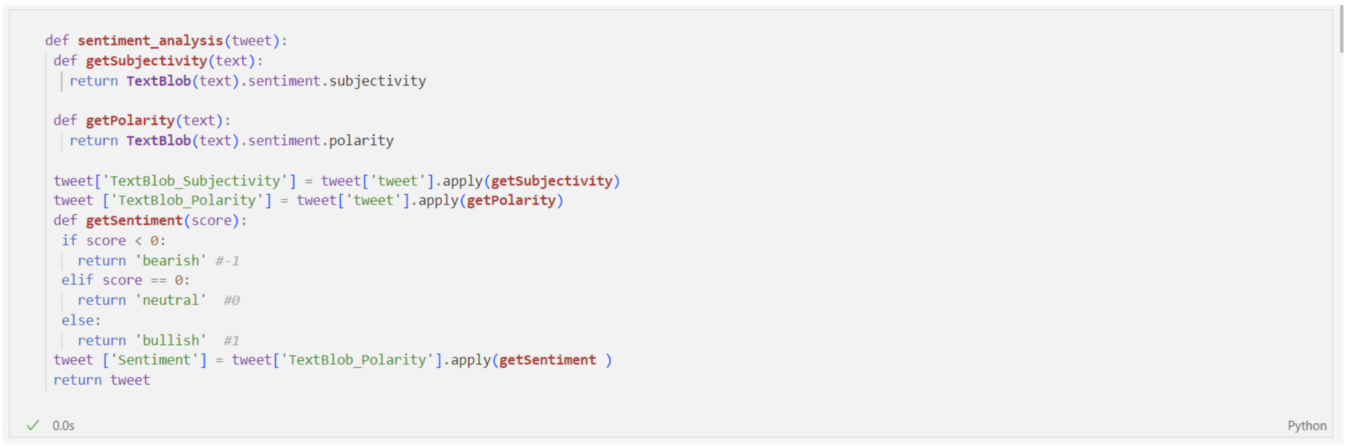


```python
import pandas as pd
import numpy as np
from textblob import TextBlob
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from textblob.classifiers import NaiveBayesClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
import wordcloud
from wordcloud import WordCloud

tweets_data = pd.read_csv('tweets_data.csv', encoding= 'utf-8-sig')
token_data = pd.read_csv('token_data.csv', encoding= 'utf-8-sig')
tweets_dictionnary = pd.read_csv('training_dictionnary.csv', encoding= 'utf-8-sig',delimiter=';')
tweet = tweets_data[tweets_data['cashtags'] == "['ocean']"][['date', 'tweet']]
```

✓ 4.9s Python

The first step is to import the necessary libraries, including pandas for data manipulation, numpy for numerical operations, TextBlob for sentiment analysis, matplotlib for data visualization, and scikit-learn for machine learning algorithms. Additionally, the code imports the WordCloud library for generating word clouds.

Next, the code loads the tweet data from a CSV file using pandas. It also loads a token data file and a training dictionary file, which may contain additional information used in the sentiment analysis process.

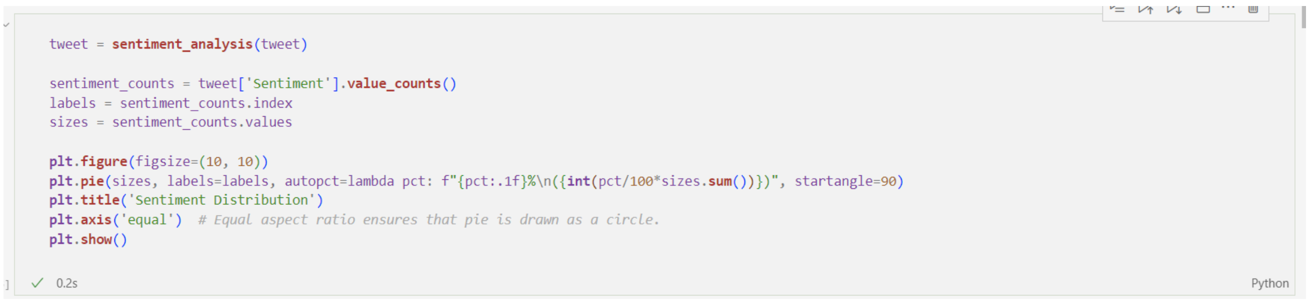


```python
def sentiment_analysis(tweet):
    def getSubjectivity(text):
        return TextBlob(text).sentiment.subjectivity

    def getPolarity(text):
        return TextBlob(text).sentiment.polarity

    tweet['TextBlob_Subjectivity'] = tweet['tweet'].apply(getSubjectivity)
    tweet ['TextBlob_Polarity'] = tweet['tweet'].apply(getPolarity)
    def getSentiment(score):
     if score < 0:
       return 'bearish' #-1
     elif score == 0:
       return 'neutral'  #0
     else:
       return 'bullish'  #1
    tweet ['Sentiment'] = tweet['TextBlob_Polarity'].apply(getSentiment )
    return tweet
```

✓ 0.0s Python

The sentiment analysis function, sentiment_analysis(tweet), is defined to calculate the sentiment polarity and subjectivity of each tweet using the TextBlob library. It assigns sentiment labels (bullish, bearish, or neutral) based on the polarity score. The function returns the modified tweet data.

The code then calls the sentiment_analysis() function on a specific subset of tweets, which are filtered based on the presence of the cashtag 'ocean'. The goal of this is to remove tweets that are talking about multiple cryptocurrencies and just use the $OCEAN cashtag as a way to get visibility.

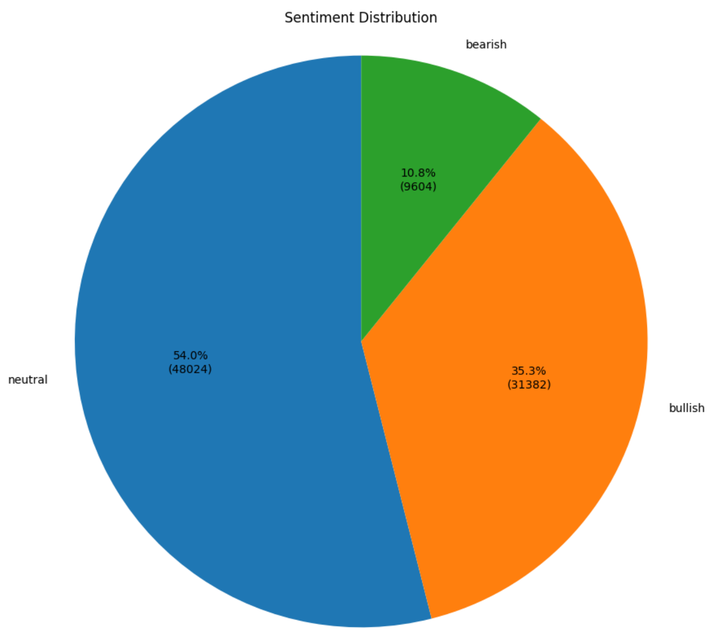


```python
tweet = sentiment_analysis(tweet)

sentiment_counts = tweet['Sentiment'].value_counts()
labels = sentiment_counts.index
sizes = sentiment_counts.values

plt.figure(figsize=(10, 10))
plt.pie(sizes, labels=labels, autopct=lambda pct: f"{pct:.1f}%\n({int(pct/100*sizes.sum())})", startangle=90)
plt.title('Sentiment Distribution')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

✓ 0.2s

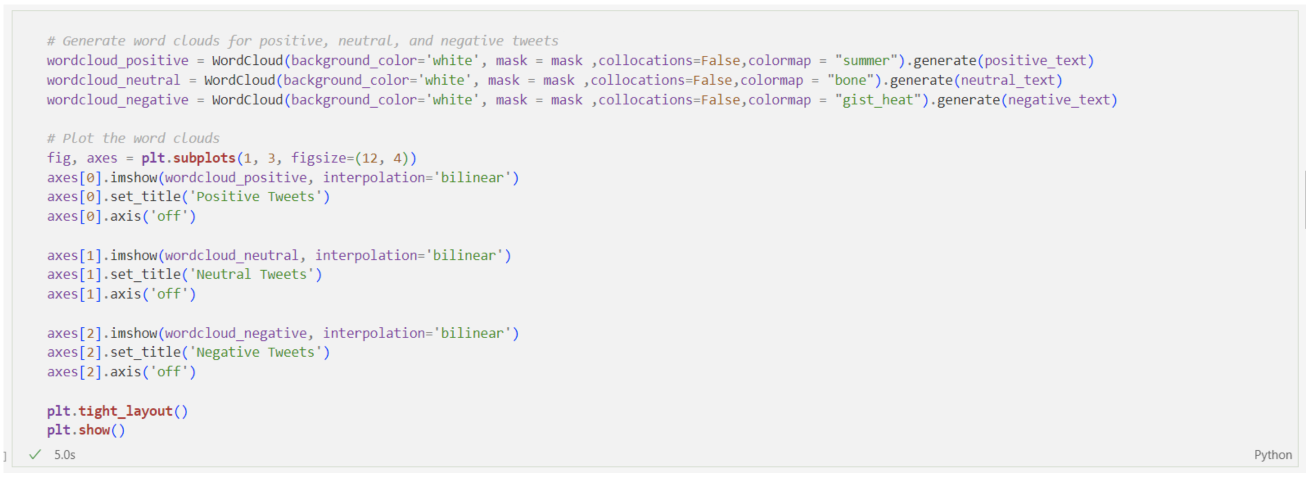
Python

To visualize the sentiment distribution, the code generates a pie chart using matplotlib. The chart displays the percentage and count of bullish, bearish, and neutral tweets.



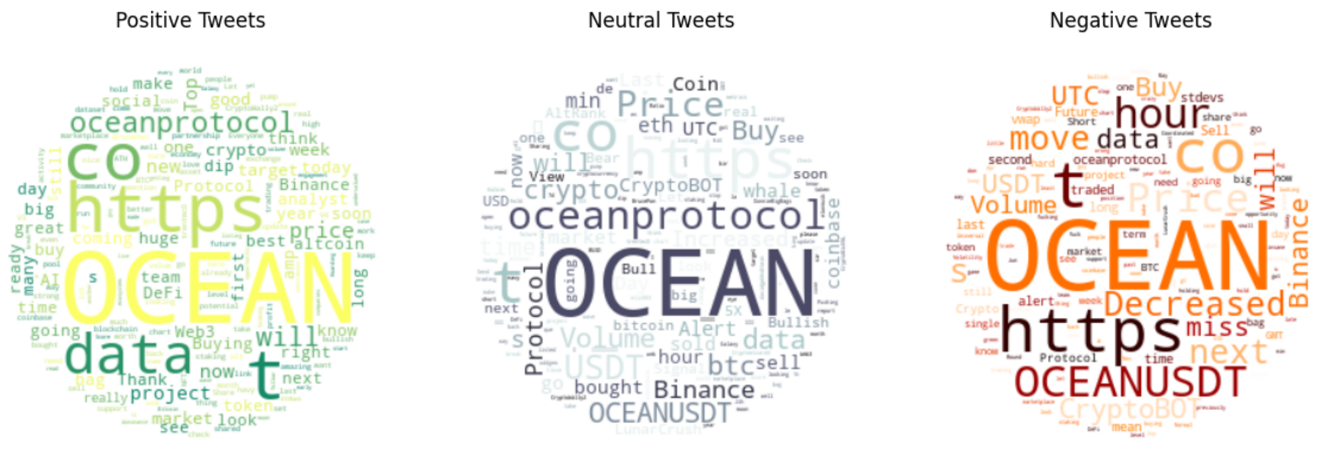
2. Visual representation of the words

The next section of the code generates word clouds for positive, neutral, and negative tweets. Word clouds provide a visual representation of the most frequently occurring words in each sentiment category.



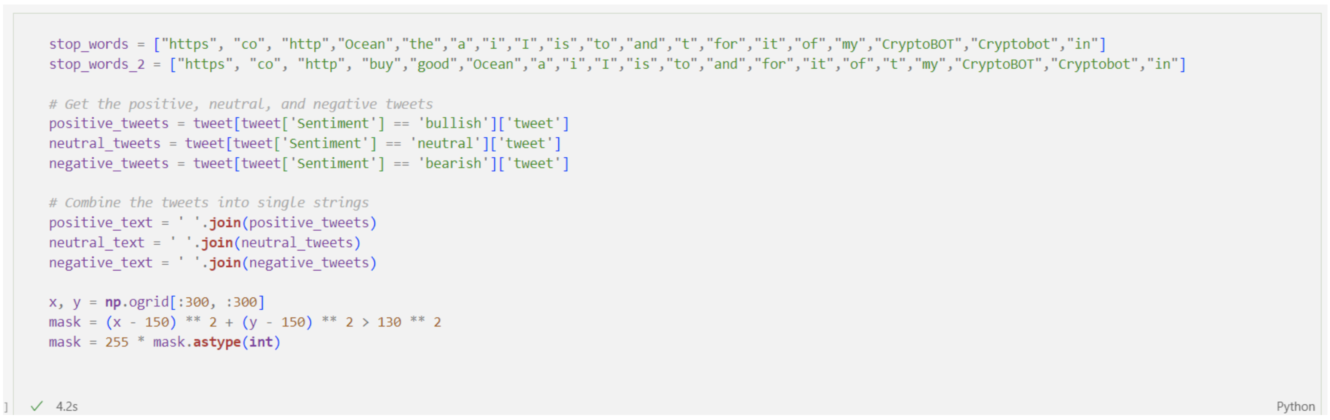
```python
# Generate word clouds for positive, neutral, and negative tweets
wordcloud_positive = WordCloud(background_color='white', mask = mask ,collocations=False,colormap = "summer").generate(positive_text)
wordcloud_neutral = WordCloud(background_color='white', mask = mask ,collocations=False,colormap = "bone").generate(neutral_text)
wordcloud_negative = WordCloud(background_color='white', mask = mask ,collocations=False,colormap = "gist_heat").generate(negative_text)

# Plot the word clouds
fig, axes = plt.subplots(1, 3, figsize=(12, 4))
axes[0].imshow(wordcloud_positive, interpolation='bilinear')
axes[0].set_title('Positive Tweets')
axes[0].axis('off')

axes[1].imshow(wordcloud_neutral, interpolation='bilinear')
axes[1].set_title('Neutral Tweets')
axes[1].axis('off')

axes[2].imshow(wordcloud_negative, interpolation='bilinear')
axes[2].set_title('Negative Tweets')
axes[2].axis('off')

plt.tight_layout()
plt.show()
```

✓ 5.0s Python



First we can see there are a lot of words that may not carry significant sentiment information.

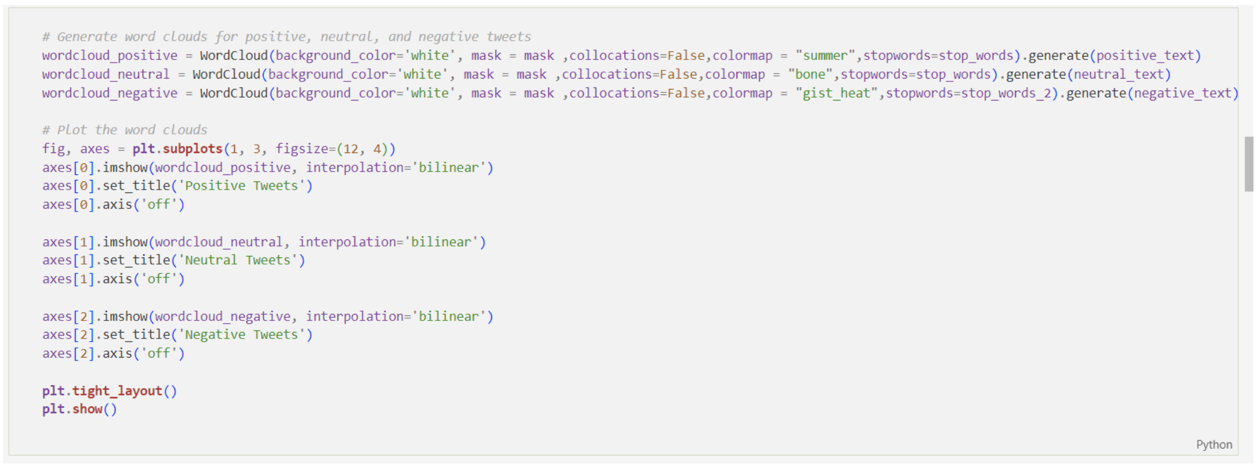
The code defines stop word lists to filter out those common words. These stop words include URLs, common English words, and some domain-specific terms.



```python
stop_words = ["https", "co", "http","Ocean","the","a","i","I","is","to","and","t","for","it","of","my","CryptoBOT","Cryptobot","in"]
stop_words_2 = ["https", "co", "http", "buy","good","Ocean","a","i","I","is","to","and","for","it","of","t","my","CryptoBOT","Cryptobot","in"]

# Get the positive, neutral, and negative tweets
positive_tweets = tweet[tweet['Sentiment'] == 'bullish']['tweet']
neutral_tweets = tweet[tweet['Sentiment'] == 'neutral']['tweet']
negative_tweets = tweet[tweet['Sentiment'] == 'bearish']['tweet']

# Combine the tweets into single strings
positive_text = ' '.join(positive_tweets)
neutral_text = ' '.join(neutral_tweets)
negative_text = ' '.join(negative_tweets)

x, y = np.ogrid[:300, :300]
mask = (x - 150) ** 2 + (y - 150) ** 2 > 130 ** 2
mask = 255 * mask.astype(int)
```

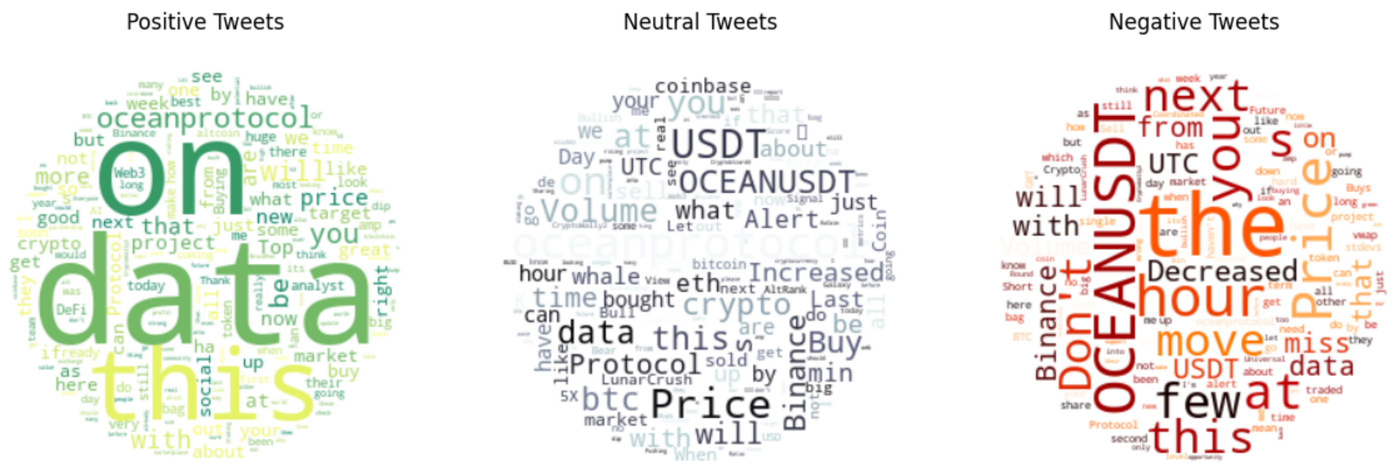
✓ 4.2s

Python



```python
# Generate word clouds for positive, neutral, and negative tweets
wordcloud_positive = WordCloud(background_color='white', mask = mask ,collocations=False,colormap = "summer",stopwords=stop_words).generate(positive_text)
wordcloud_neutral = WordCloud(background_color='white', mask = mask ,collocations=False,colormap = "bone",stopwords=stop_words).generate(neutral_text)
wordcloud_negative = WordCloud(background_color='white', mask = mask ,collocations=False,colormap = "gist_heat",stopwords=stop_words_2).generate(negative_text)

# Plot the word clouds
fig, axes = plt.subplots(1, 3, figsize=(12, 4))
axes[0].imshow(wordcloud_positive, interpolation='bilinear')
axes[0].set_title('Positive Tweets')
axes[0].axis('off')

axes[1].imshow(wordcloud_neutral, interpolation='bilinear')
axes[1].set_title('Neutral Tweets')
axes[1].axis('off')

axes[2].imshow(wordcloud_negative, interpolation='bilinear')
axes[2].set_title('Negative Tweets')
axes[2].axis('off')

plt.tight_layout()
plt.show()
```

Python

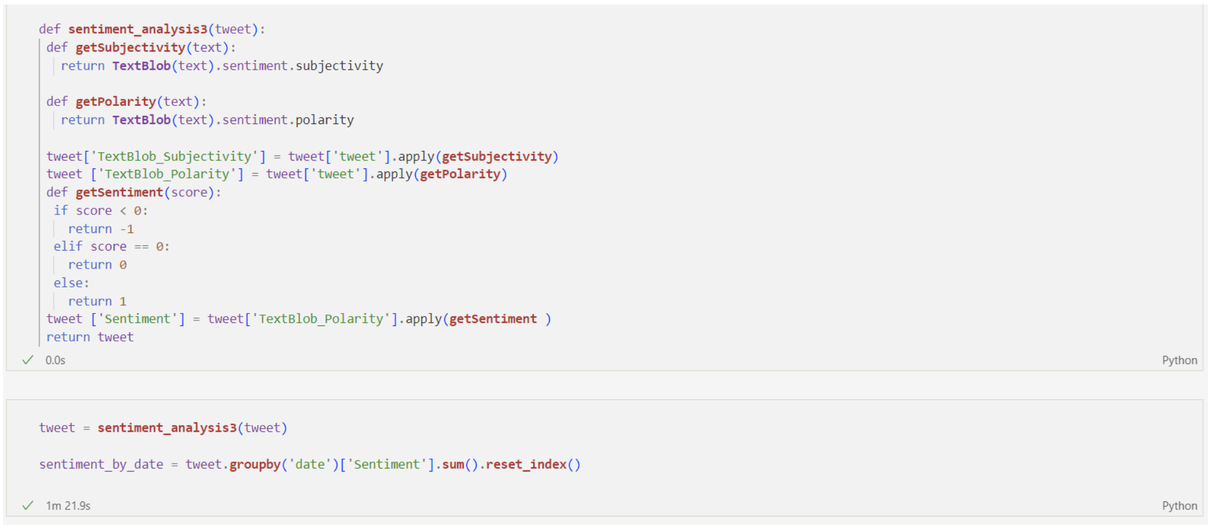


After filtering the words in the word clouds, interesting patterns emerge in terms of the words associated with different sentiment categories.

In the positive word cloud, we observe words related to the project, such as "data," "community," and even the name of the founder, "BrucePon." This suggests that positive sentiment in the tweets is often associated with discussions about the project's development, community engagement, and possibly positive news or updates related to the project. These words indicate a generally optimistic outlook and enthusiasm surrounding the project.

The neutral word cloud, on the other hand, contains words that are more factual and related to the financial aspect of the topic. Words like "price,", USDT, "ETH", "BTC", and "financial movements" indicate that neutral sentiment tweets tend to focus on analyzing and discussing the price movements of cryptocurrencies, particularly in relation to the project. These tweets may provide information or updates about the financial aspects of the project without expressing a clear positive or negative sentiment.

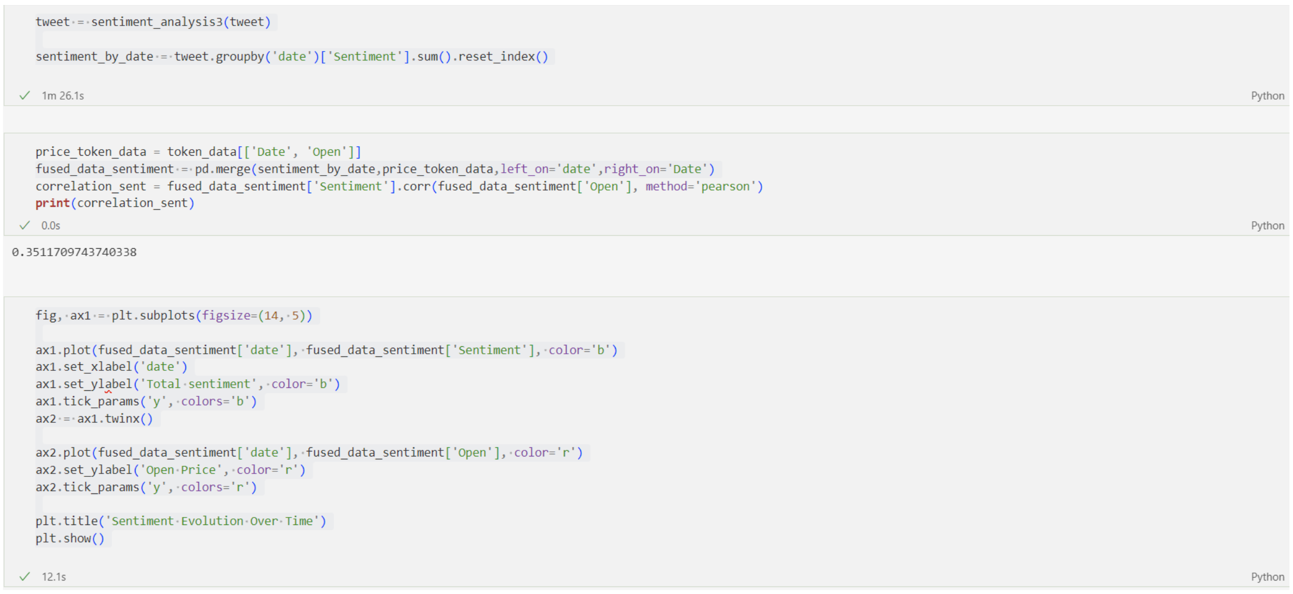
In the negative word cloud, we observe words that reflect fear or concern about unfavorable financial movements. Negative sentiment tweets may contain words related to the fear of losing money or experiencing a decline in the price of "ocean." This suggests that negative sentiment is often associated with a pessimistic outlook or worry about potential negative impacts on financial investments in relation to the project. These tweets may express concerns or caution about the project's performance or future prospects.



```python
def sentiment_analysis3(tweet):
 def getSubjectivity(text):
   return TextBlob(text).sentiment.subjectivity

 def getPolarity(text):
   return TextBlob(text).sentiment.polarity

 tweet['TextBlob_Subjectivity'] = tweet['tweet'].apply(getSubjectivity)
 tweet ['TextBlob_Polarity'] = tweet['tweet'].apply(getPolarity)
 def getSentiment(score):
  if score < 0:
    return -1
  elif score == 0:
    return 0
  else:
    return 1
 tweet ['Sentiment'] = tweet['TextBlob_Polarity'].apply(getSentiment )
 return tweet
```

```python
tweet = sentiment_analysis3(tweet)

sentiment_by_date = tweet.groupby('date')['Sentiment'].sum().reset_index()
```

Modifying a bit our function, we change the output : instead of giving “neutral” “bullish” or “bearish” we attribute 0, 1 or -1 as a value.



```python
tweet = sentiment_analysis3(tweet)

sentiment_by_date = tweet.groupby('date')['Sentiment'].sum().reset_index()
```

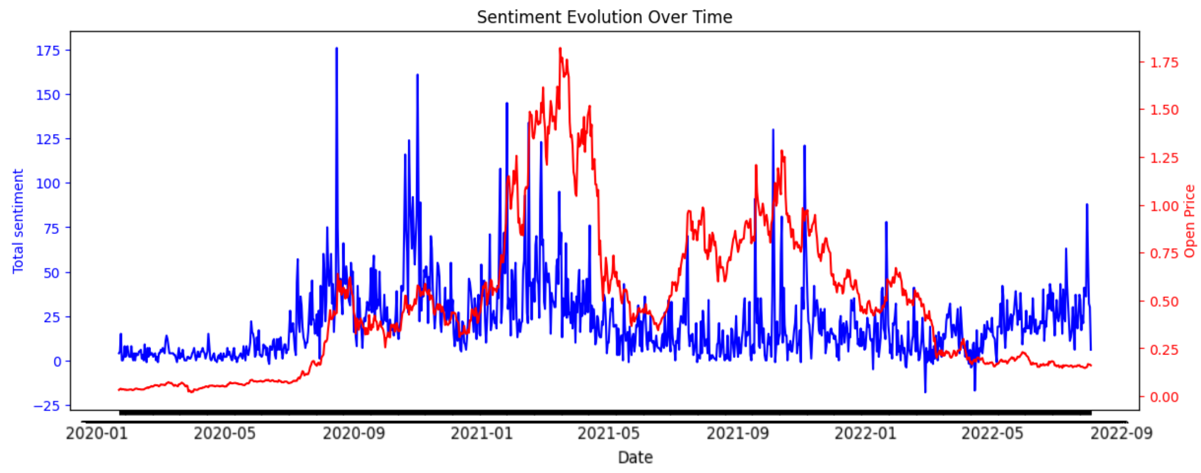
```python
price_token_data = token_data[['Date', 'Open']]
fused_data_sentiment = pd.merge(sentiment_by_date,price_token_data,left_on='date',right_on='Date')
correlation_sent = fused_data_sentiment['Sentiment'].corr(fused_data_sentiment['Open'], method='pearson')
print(correlation_sent)
```

```
0.3511709743740338
```

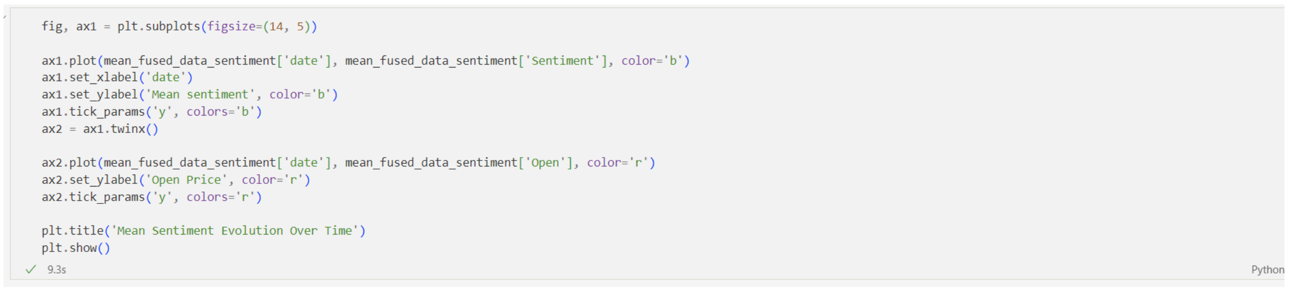
```python
fig, ax1 = plt.subplots(figsize=(14, 5))

ax1.plot(fused_data_sentiment['date'], fused_data_sentiment['Sentiment'], color='b')
ax1.set_xlabel('date')
ax1.set_ylabel('Total sentiment', color='b')
ax1.tick_params('y', colors='b')
ax2 = ax1.twinx()

ax2.plot(fused_data_sentiment['date'], fused_data_sentiment['Open'], color='r')
ax2.set_ylabel('Open Price', color='r')
ax2.tick_params('y', colors='r')

plt.title('Sentiment Evolution Over Time')
plt.show()
```

This allows us to sum the outputs for the tweets each day and get the general polarity for this day. We then calculate the correlation with the price and we end up with a 0.35 Pearson coefficient, which is a low but still not negligible positive correlation.

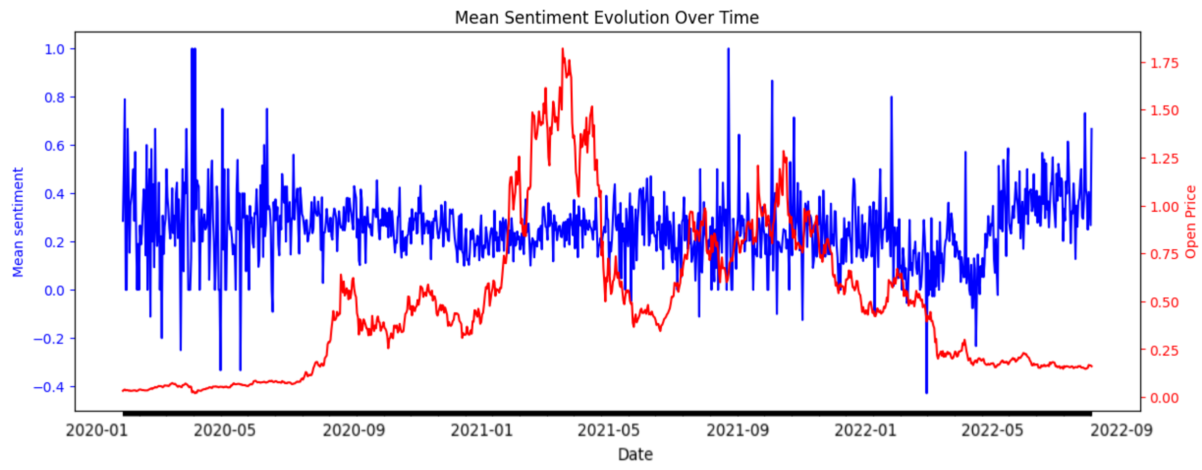
We can make the graph of Sentiment evolution over time as found below. As there are more neutral and positive tweets than negative, this will be very linked to the number of tweets.



As this will be very dependent on the number of tweets, we can also make the average sentiment per day over the time period.

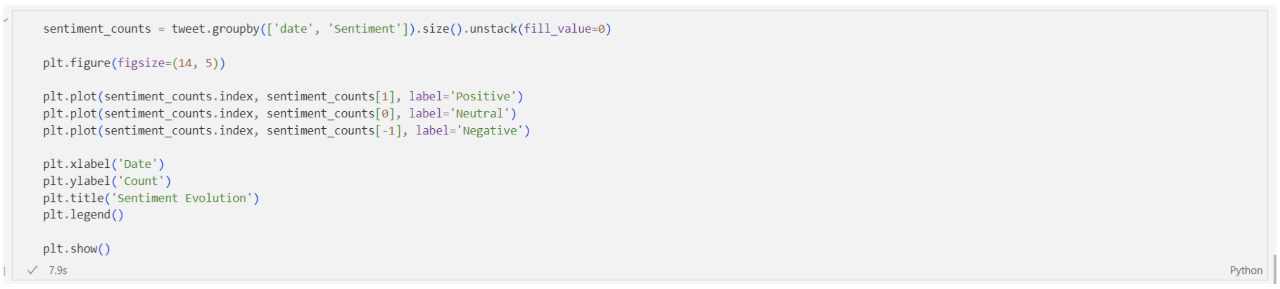


```
fig, ax1 = plt.subplots(figsize=(14, 5))

ax1.plot(mean_fused_data_sentiment['date'], mean_fused_data_sentiment['Sentiment'], color='b')
ax1.set_xlabel('date')
ax1.set_ylabel('Mean sentiment', color='b')
ax1.tick_params('y', colors='b')
ax2 = ax1.twinx()

ax2.plot(mean_fused_data_sentiment['date'], mean_fused_data_sentiment['Open'], color='r')
ax2.set_ylabel('Open Price', color='r')
ax2.tick_params('y', colors='r')

plt.title('Mean Sentiment Evolution Over Time')
plt.show()
```

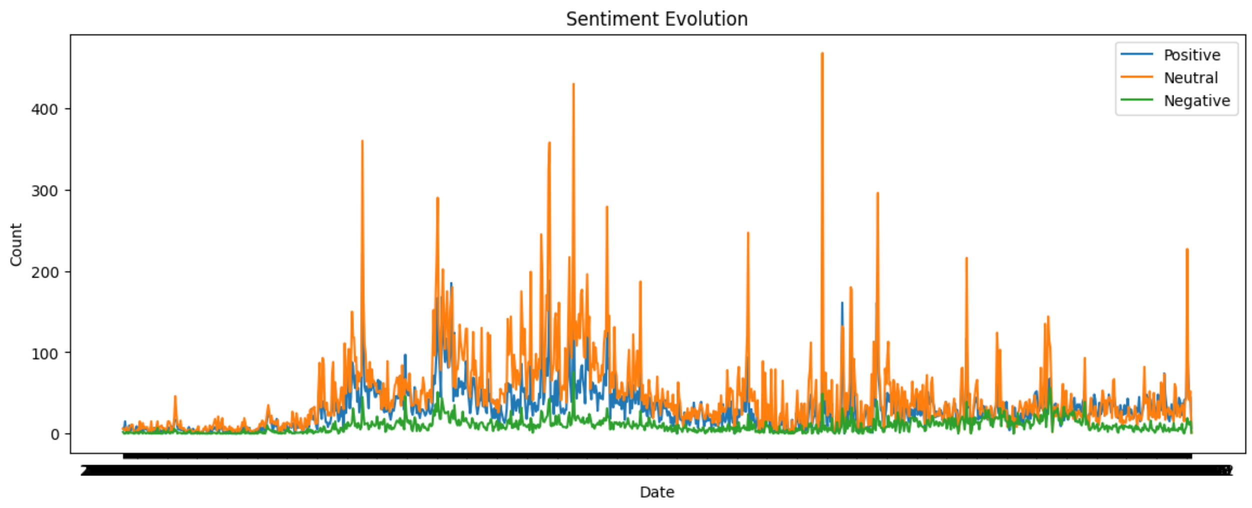


This allows us to really see for each day, independent of the number of tweets and the period (Bullrun for example), what people think of Ocean on average. We can the the average sentiment dipped around 2022-05 but is now up and by quite a lot.

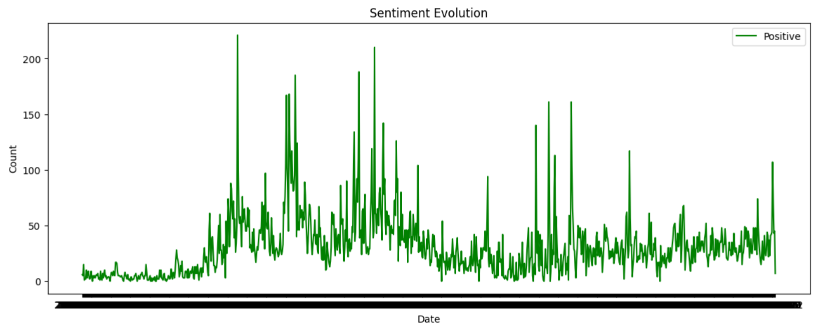
To have another graphical analysis, we plot the number of each positive, neutral and negative tweets over time.

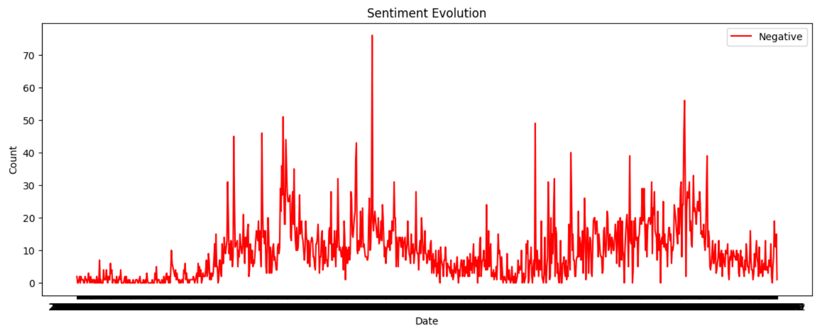


```
sentiment_counts = tweet.groupby(['date', 'Sentiment']).size().unstack(fill_value=0)

plt.figure(figsize=(14, 5))

plt.plot(sentiment_counts.index, sentiment_counts[1], label='Positive')
plt.plot(sentiment_counts.index, sentiment_counts[0], label='Neutral')
plt.plot(sentiment_counts.index, sentiment_counts[-1], label='Negative')

plt.xlabel('Date')
plt.ylabel('Count')
plt.title('Sentiment Evolution')
plt.legend()

plt.show()
```



Of course this is very dependent on the number of tweets again, but can allow us to see some periods where the number of positive/negative tweets spike.





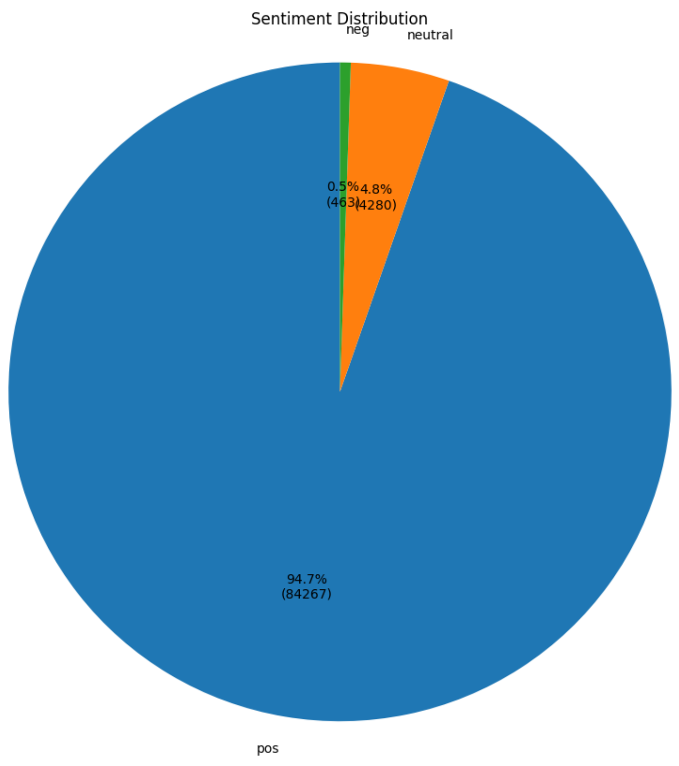
Graphs of only the positive tweets count and negative tweets counts over time.

3. Other model tested

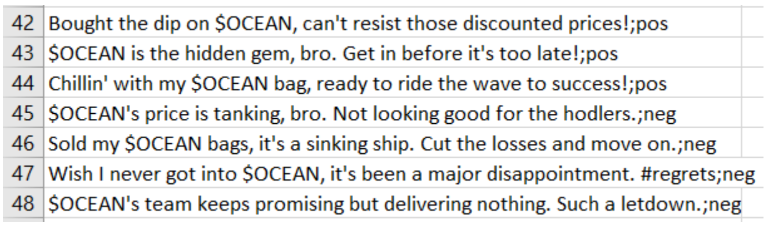
In addition to the previous model, we also attempted to develop another model that was trained on a part of the dataset of tweets labeled by hand as positive, neutral, or negative. The aim was to create a model that is more aligned with the language commonly used in the crypto and Twitter communities. To augment the dataset, we even used a large language model to generate some of the labeled tweets.

However, this alternative model didn't perform as well as expected. One of the main challenges was the limited amount of labeled data available for training. Creating a large and accurately labeled dataset for sentiment analysis can be a time-consuming and resource-intensive task. Consequently, the model's performance was affected by the lack of sufficient training data.
Improving this model would require a significant effort, including gathering a larger labeled dataset and involving the community to help with the labeling process. By harnessing the collective knowledge and expertise of the community, it becomes possible to create a more comprehensive and accurate sentiment analysis model specific to the crypto and Twitter language.

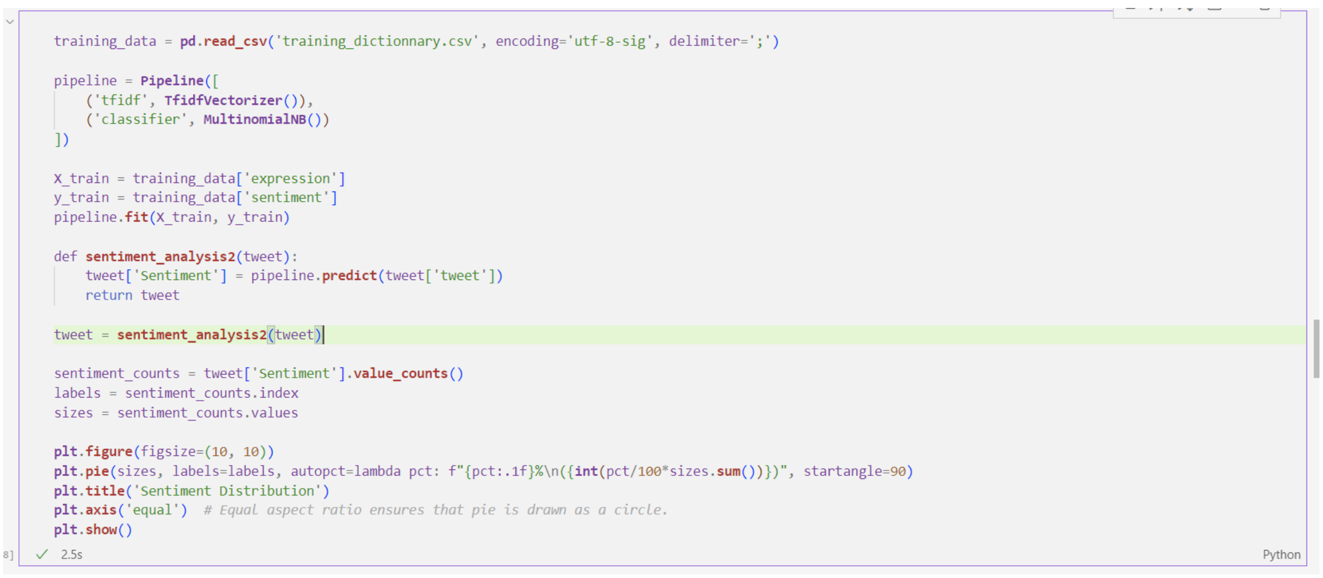
Despite its limitations, we applied the alternative model to the tweet data. The sentiment labels predicted by the model were used to visualize the sentiment distribution in a pie chart. This chart displays the percentage and count of bullish, bearish, and neutral tweets, providing an overview of the sentiment distribution based on the alternative model's predictions.



Here is the code and a snippet of the dataset used for the other model :



42	Bought the dip on $OCEAN, can't resist those discounted prices!;pos
43	$OCEAN is the hidden gem, bro. Get in before it's too late!;pos
44	Chillin' with my $OCEAN bag, ready to ride the wave to success!;pos
45	$OCEAN's price is tanking, bro. Not looking good for the hodlers.;neg
46	Sold my $OCEAN bags, it's a sinking ship. Cut the losses and move on.;neg
47	Wish I never got into $OCEAN, it's been a major disappointment. #regrets;neg
48	$OCEAN's team keeps promising but delivering nothing. Such a letdown.;neg



```python
training_data = pd.read_csv('training_dictionnary.csv', encoding='utf-8-sig', delimiter=';')

pipeline = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('classifier', MultinomialNB())
])

X_train = training_data['expression']
y_train = training_data['sentiment']
pipeline.fit(X_train, y_train)

def sentiment_analysis2(tweet):
    tweet['Sentiment'] = pipeline.predict(tweet['tweet'])
    return tweet

tweet = sentiment_analysis2(tweet)

sentiment_counts = tweet['Sentiment'].value_counts()
labels = sentiment_counts.index
sizes = sentiment_counts.values

plt.figure(figsize=(10, 10))
plt.pie(sizes, labels=labels, autopct=lambda pct: f"{pct:.1f}%\n({int(pct/100*sizes.sum())})", startangle=90)
plt.title('Sentiment Distribution')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

Conclusion

In conclusion, I absolutely loved working on this project.

By calculating the correlation between the price of $OCEAN and the number of tweets, likes, retweets, and individuals using the cashtag, we gained valuable insights.
We also delved into the impact of influential tweets on the price of the OCEAN token. This analysis helped us identify the power of influential individuals or news events in driving price movements.

Moreover, our machine learning model, utilizing TextBlob for sentiment analysis, proved to be an invaluable tool. It enabled us to classify tweets and extract meaningful insights, including the identification of main themes within each sentiment category.

We extend our gratitude to the Ocean team for providing challenges like this, which allowed us to explore the dynamics of crypto tokens and gain a deeper understanding of its market behavior.