Datasets for further Anaysis - Preprocessed

```
#Extrating the Preproceed Data for further Analysis
```

```
#Bitcoin Price
import pandas as pd

# URL to the raw CSV file
url = 'https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Sentimental%20Analysis/BitcoinPricePreprocessed.csv'

# Read the CSV file from the URL
crypto_usd = pd.read_csv(url)

# Display the first few rows of the data
print(crypto_usd.head())
```

```
                  time     close      high       low      open  volumefrom  \
0  2023-02-19 13:00:00  24682.03  24715.82  24682.03  24707.39      903.97
1  2023-02-19 14:00:00  24765.79  24792.85  24679.21  24682.03     1220.29
2  2023-02-19 15:00:00  24928.21  25022.49  24751.96  24765.79     5074.50
3  2023-02-19 16:00:00  24786.44  25175.28  24704.53  24928.21     7094.72
4  2023-02-19 17:00:00  24364.95  24806.64  24346.17  24786.44     6896.84

        volumeto        Date      Time        volume     marketcap  price_delta
0  2.233594e+07  2023-02-19  13:00:00  2.233504e+07  5.512964e+11          NaN
1  3.020300e+07  2023-02-19  14:00:00  3.020178e+07  7.480012e+11        83.76
2  1.263085e+08  2023-02-19  15:00:00  1.263034e+08  3.148644e+12       162.42
3  1.770671e+08  2023-02-19  16:00:00  1.770600e+08  4.388863e+12      -141.77
4  1.693379e+08  2023-02-19  17:00:00  1.693310e+08  4.125910e+12      -421.49
```

```
import pandas as pd

file_urls = [
    'https://github.com/Amarpreet3/CIND-820-CAPSTONE/raw/main/Sentimental%20Analysis/BitcoinTweetsPreprocessed_1.csv',
    'https://github.com/Amarpreet3/CIND-820-CAPSTONE/raw/main/Sentimental%20Analysis/BitcoinTweetsPreprocessed_2.csv',
    'https://github.com/Amarpreet3/CIND-820-CAPSTONE/raw/main/Sentimental%20Analysis/BitcoinTweetsPreprocessed_3.csv',
    'https://github.com/Amarpreet3/CIND-820-CAPSTONE/raw/main/Sentimental%20Analysis/BitcoinTweetsPreprocessed_4.csv',
    'https://github.com/Amarpreet3/CIND-820-CAPSTONE/raw/main/Sentimental%20Analysis/BitcoinTweetsPreprocessed_5.csv',
    'https://github.com/Amarpreet3/CIND-820-CAPSTONE/raw/main/Sentimental%20Analysis/BitcoinTweetsPreprocessed_6.csv'
]

dfs = []

for url in file_urls:
    # Read the CSV file
    df = pd.read_csv(url)

    # Append the DataFrame to the list
    dfs.append(df)

# Combine all DataFrames into a single DataFrame
combined_df = pd.concat(dfs)

# Display the first few rows of the combined DataFrame
print(combined_df.head())
```

```
             user_name  user_location  \
0                  Irk  Vancouver, WA
1         Xiang Zhang            NaN
2              Rhizoo            NaN
3       Hari Marquez  Las Vegas, NV
4  Bitcoin Candle Bot         Brazil

                                  user_description         user_created  \
0  Irk started investing in the stock market in 1...  2018-08-11 03:17:00
1  Professional Software Engineer ð»ðCrypto ...  2011-01-11 01:37:00
2  researcher. local maxima dunningâkruger spec...  2019-04-03 18:09:00
3  Donât trust, verify. #Bitcoin | El Salvador ...  2014-01-17 23:04:00
4  Robot that posts the closure of the bitcoin da...  2021-01-06 01:36:00

   user_followers  user_friends  user_favourites  user_verified  \
```

```
0          116.0          8.0          4580.0         False
1           42.0         22.0             5.0         False
2          778.0        627.0         32005.0         False
3          222.0        521.0         13052.0         False
4           40.0          4.0             1.0         False

                 date                                               text  \
0  2023-02-25 23:59:00  bitcoin btc rest crypto ye bitcoin cryptocurr ...
1  2023-02-25 23:59:00  retriev invest fund current ongo tidexcoin kic...
2  2023-02-25 23:59:00  bull save monthli thread today good shit bitco...
3  2023-02-25 23:59:00         el salvador shape futur bitcoin membvk32cn
4  2023-02-25 23:59:00  candl day 25022023 close open 2319406 high 232...

                                      hashtags             source  \
0           ['Bitcoin', 'crypto', 'NeedsMoreCrash']     Twitter Web App
1  ['Tidexcoin', 'Kicurrency', 'LMY', 'GMK', 'SYR...  Twitter for iPhone
2                                     ['bitcoin']     Twitter Web App
3                                     ['Bitcoin']     Twitter Web App
4              ['Bitcoin', 'Candle', 'BearMarket']  Bitcoin Candle Bot

   is_retweet  compound         score sentiment_level  polarity  subjectivity
0         0.0   -0.4019 -2.154092e+05        Negative  0.000000      0.000000
1         0.0    0.0000  0.000000e+00         Neutral  0.000000      0.400000
2         0.0    0.3612  9.005682e+06        Positive  0.250000      0.700000
3         0.0    0.0000  0.000000e+00         Neutral  0.000000      0.000000
4         0.0   -0.2732 -2.240240e+01        Negative  0.053333      0.446667
```

```
1 tweets = combined_df.copy()
```

```
1 tweets.head()
```

	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	user_verified	date	
0	Irk	Vancouver, WA	Irk started investing in the stock market in 1...	2018-08-11 03:17:00	116.0	8.0	4580.0	False	2023-02-25 23:59:00	b re y cryp
1	Xiang Zhang	NaN	Professional Software Engineer ð□□»ð□□□Crypto ...	2011-01-11 01:37:00	42.0	22.0	5.0	False	2023-02-25 23:59:00	retri fun
2	Rhizoo	NaN	researcher. local maxima dunningâ□□kruger spec...	2019-04-03 18:09:00	778.0	627.0	32005.0	False	2023-02-25 23:59:00	thre
3	Hari Marquez	Las Vegas, NV	Donâ□□t trust, verify. #Bitcoin | El Salvador ...	2014-01-17 23:04:00	222.0	521.0	13052.0	False	2023-02-25 23:59:00	el sh mem
4	Bitcoin Candle Bot	Brazil	Robot that posts the closure of the bitcoin da...	2021-01-06 01:36:00	40.0	4.0	1.0	False	2023-02-25 23:59:00	c 2 cl hi

```
1 print(tweets.columns)
```

```
Index(['user_name', 'user_location', 'user_description', 'user_created',
       'user_followers', 'user_friends', 'user_favourites', 'user_verified',
       'date', 'text', 'hashtags', 'source', 'is_retweet', 'compound', 'score',
       'sentiment_level', 'polarity', 'subjectivity'],
      dtype='object')
```

```
1 import pandas as pd
2
3
4 # Check the shape of the dataset
5 print("Shape of the dataset:", tweets.shape)
6
7 # Check the size of the dataset
```

```
print("Size of the dataset (number of elements):",tweets.size)
```

```
Shape of the dataset: (167652, 18)
Size of the dataset (number of elements): 3017736
```

```
import pandas as pd
import os


# Check the shape of the data
print("Shape of the data:", tweets.shape)
```

```
Shape of the data: (167652, 18)
```

```
label_counts = tweets['sentiment_level'].value_counts()
print(label_counts)
```

```
Neutral             93169
Positive            35921
Extreme Positive    17343
Negative            15903
Extreme Negative     5316
Name: sentiment_level, dtype: int64
```

```
#Crypto - Bitcoin
crypto_usd.head(2)

```

	time	close	high	low	open	volumefrom	volumeto	Date	Time	volume	marketcap	price_delta
0	2023-02-19 13:00:00	24682.03	24715.82	24682.03	24707.39	903.97	22335943.28	2023-02-19	13:00:00	22335039.31	5.512964e+11	NaN
1	2023-02-19	24765.70	24792.85	24679.21	24682.03	1220.20	30203001.55	2023-	14:00:00	30201781.26	7.480012e+11	83.76

```
#Tweets-Bitcoin

tweets.head(2)
```

	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	user_verified	date	
0	Irk	Vancouver, WA	Irk started investing in the stock market in 1...	2018-08-11 03:17:00	116.0	8.0	4580.0	False	2023-02-25 23:59:00	b bt cryp b crypl
1	Xiang Zhang	NaN	Professional Software Engineer ð□□»ð□□□Crypto ...	2011-01-11 01:37:00	42.0	22.0	5.0	False	2023-02-25 23:59:00	r i c tide

Exploratory Data Analysis (EDA)

Data Description

```
print(type(df))
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
df.head(2)
```

	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	user_verified	date	
0	Bitcoin Treasures ð□□□	Bitcoin City	I am a bot tracking the USD value of various b...	2020-08-23 16:20:00	2603.0	8.0	48.0	False	2023-03-04 20:00:00	worth
1	Bitcoin Price	Order Book	$BTC price updated every 4 hours\n\nPrices in ...	2022-03-25 08:31:00	16.0	0.0	2.0	False	2023-03-04 20:00:00	b sat

2 rows × 21 columns

```
1
```

```
1 import pandas as pd
2
3 # Iterate through each attribute
4 for column in df.columns:
5     unique_count = df[column].nunique()
6     print(f"Unique count for attribute '{column}': {unique_count}")
7
```

```
Unique count for attribute 'user_name': 36053
Unique count for attribute 'user_location': 9432
Unique count for attribute 'user_description': 33094
Unique count for attribute 'user_created': 36531
Unique count for attribute 'user_followers': 11793
Unique count for attribute 'user_friends': 5190
Unique count for attribute 'user_favourites': 17427
Unique count for attribute 'user_verified': 2
Unique count for attribute 'date': 10215
Unique count for attribute 'text': 162425
Unique count for attribute 'hashtags': 40623
Unique count for attribute 'source': 719
Unique count for attribute 'is_retweet': 1
Unique count for attribute 'compound': 1716
Unique count for attribute 'score': 56006
Unique count for attribute 'sentiment_level': 5
Unique count for attribute 'polarity': 2584
Unique count for attribute 'subjectivity': 2594
```

```
1 df.columns
```

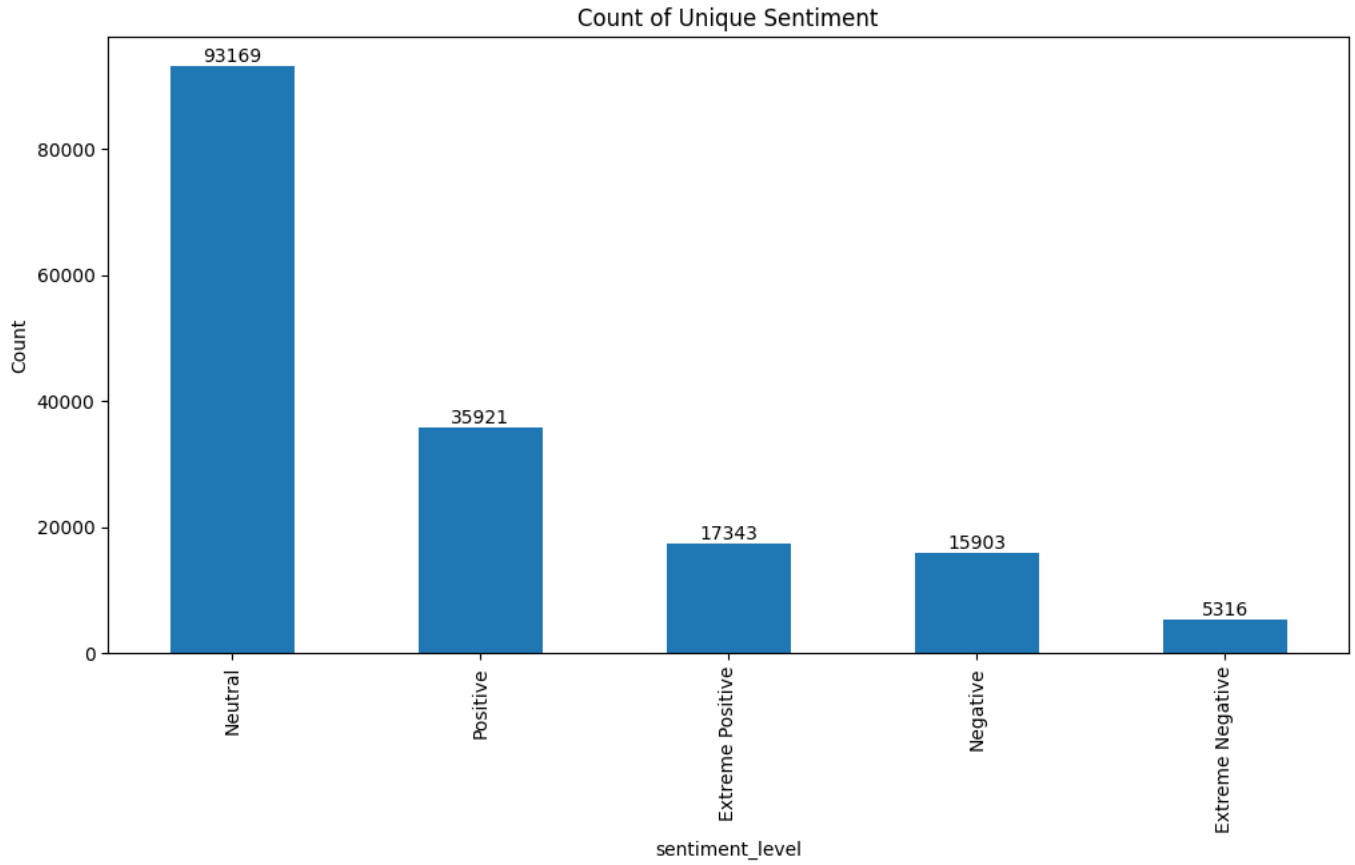
```
Index(['user_name', 'user_location', 'user_description', 'user_created',
       'user_followers', 'user_friends', 'user_favourites', 'user_verified',
       'date', 'text', 'hashtags', 'source', 'is_retweet', 'compound', 'score',
       'sentiment_level', 'polarity', 'subjectivity'],
      dtype='object')
```

```
1 df['sentiment_level'].value_counts()
```

```
Neutral             93169
Positive            35921
Extreme Positive    17343
Negative            15903
Extreme Negative     5316
Name: sentiment_level, dtype: int64
```

```
1 import matplotlib.pyplot as plt
2
3 counts = df['sentiment_level'].value_counts()
4
5 # Plotting the bar chart
6 plt.figure(figsize=(12, 6))
7 ax = counts.plot(kind='bar')
8 plt.xlabel('sentiment_level')
9 plt.ylabel('Count')
10 plt.title('Count of Unique Sentiment')
11 plt.xticks(rotation=90)
12
```

```
# Labeling the height on top of each bar
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() + p.get_width() / 2, p.get_height()), ha='center', va='bottom')

plt.show()

```




```

print(df.dtypes)
```

```
user_name            object
user_location        object
user_description     object
user_created         object
user_followers      float64
user_friends        float64
user_favourites     float64
user_verified          bool
date                 object
text                 object
hashtags             object
source               object
is_retweet          float64
compound            float64
score               float64
sentiment_level      object
polarity            float64
subjectivity        float64
dtype: object
```

```

# Get the shape of the merged dataset
rows, columns = df.shape
print("Number of rows:", rows)
print("Number of columns:", columns)
```

```
Number of rows: 167652
Number of columns: 18
```

```
# Displaying the summary information
summary = df.info()
```

```
3 print(summary)
4
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 167652 entries, 0 to 27941
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   user_name         167642 non-null  object 
 1   user_location     83349 non-null   object 
 2   user_description  157323 non-null  object 
 3   user_created      167652 non-null  object 
 4   user_followers    167652 non-null  float64
 5   user_friends      167652 non-null  float64
 6   user_favourites   167652 non-null  float64
 7   user_verified     167652 non-null  bool   
 8   date              167652 non-null  object 
 9   text              167652 non-null  object 
 10  hashtags          167142 non-null  object 
 11  source            167142 non-null  object 
 12  is_retweet        167142 non-null  float64
 13  compound          167652 non-null  float64
 14  score             167652 non-null  float64
 15  sentiment_level   167652 non-null  object 
 16  polarity          167652 non-null  float64
 17  subjectivity      167652 non-null  float64
dtypes: bool(1), float64(8), object(9)
memory usage: 23.2+ MB
None
```

```
1 # Let's see meta information about numeric data, we can also see if there any extreme values
2 df.describe()
```

	user_followers	user_friends	user_favourites	is_retweet	compound	score	polarity	subjectivity
count	1.676520e+05	167652.000000	1.676520e+05	167142.0	167652.000000	1.676520e+05	167652.000000	167652.000000
mean	1.069845e+04	774.246069	6.267079e+03	0.0	0.105530	3.260548e+07	0.069989	0.246614
std	1.317029e+05	2691.620939	2.192673e+04	0.0	0.339531	1.440558e+09	0.223690	0.271300
min	0.000000e+00	0.000000	0.000000e+00	0.0	-0.991300	-9.459832e+10	-1.000000	0.000000
25%	1.190000e+02	9.000000	1.100000e+01	0.0	0.000000	0.000000e+00	0.000000	0.000000
50%	5.560000e+02	123.000000	2.810000e+02	0.0	0.000000	0.000000e+00	0.000000	0.223016
75%	1.956000e+03	606.000000	3.300000e+03	0.0	0.361200	7.648344e+03	0.100000	0.418182
max	1.878937e+07	254276.000000	1.083014e+06	0.0	0.984000	1.350093e+11	1.000000	1.000000

```
1 df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
user_followers	167652.0	1.069845e+04	1.317029e+05	0.000000e+00	119.0	556.000000	1956.000000	1.878937e+07
user_friends	167652.0	7.742461e+02	2.691621e+03	0.000000e+00	9.0	123.000000	606.000000	2.542760e+05
user_favourites	167652.0	6.267079e+03	2.192673e+04	0.000000e+00	11.0	281.000000	3300.000000	1.083014e+06
is_retweet	167142.0	0.000000e+00	0.000000e+00	0.000000e+00	0.0	0.000000	0.000000	0.000000e+00
compound	167652.0	1.055299e-01	3.395308e-01	-9.913000e-01	0.0	0.000000	0.361200	9.840000e-01
score	167652.0	3.260548e+07	1.440558e+09	-9.459832e+10	0.0	0.000000	7648.344025	1.350093e+11
polarity	167652.0	6.998889e-02	2.236899e-01	-1.000000e+00	0.0	0.000000	0.100000	1.000000e+00
subjectivity	167652.0	2.466140e-01	2.712997e-01	0.000000e+00	0.0	0.223016	0.418182	1.000000e+00

Analysing Null/NAN values

```
1 df.isnull().sum()
```

```
user_name             10
user_location      84303
user_description   10329
user_created           0
user_followers         0
```

```
user_friends           0
user_favourites        0
user_verified          0
date                   0
text                   0
hashtags             510
source               510
is_retweet           510
compound               0
score                  0
sentiment_level        0
polarity               0
subjectivity           0
dtype: int64
```

```
#Null values for attributes which are not significant, so keepin the dataset unaltered
```

```
df.isnull().count()
```

```
user_name           27942
user_location       27942
user_description    27942
user_created        27942
user_followers      27942
user_friends        27942
user_favourites     27942
user_verified       27942
date                27942
text                27942
hashtags            27942
source              27942
is_retweet          27942
compound            27942
score               27942
sentiment_level     27942
polarity            27942
subjectivity        27942
dtype: int64
```

Statistics Summary

```
# Display summary statistics of numeric columns
numeric_columns = df.select_dtypes(include=[int, float])
summary_statistics = numeric_columns.describe()
print("Summary Statistics:")
print(summary_statistics)
```

```
Summary Statistics:
       user_followers   user_friends  user_favourites  is_retweet  \
count    1.676520e+05  167652.000000     1.676520e+05    167142.0
mean     1.069845e+04     774.246069     6.267079e+03         0.0
std      1.317029e+05    2691.620939     2.192673e+04         0.0
min      0.000000e+00       0.000000     0.000000e+00         0.0
25%      1.190000e+02       9.000000     1.100000e+01         0.0
50%      5.560000e+02     123.000000     2.810000e+02         0.0
75%      1.956000e+03     606.000000     3.300000e+03         0.0
max      1.878937e+07  254276.000000     1.083014e+06         0.0

            compound         score       polarity   subjectivity
count  167652.000000  1.676520e+05  167652.000000  167652.000000
mean        0.105530  3.260548e+07       0.069989       0.246614
std         0.339531  1.440558e+09       0.223690       0.271300
min        -0.991300 -9.459832e+10      -1.000000       0.000000
25%         0.000000  0.000000e+00       0.000000       0.000000
50%         0.000000  0.000000e+00       0.000000       0.223016
75%         0.361200  7.648344e+03       0.100000       0.418182
max         0.984000  1.350093e+11       1.000000       1.000000
```

Categorical to One-Hot (numeric) Encoding

```
#Let's create a list for our categorical columns
#cat_cols=["Name", "Symbol", "Date"]
```

```
# Create a copy of the data frame in memory with a different name
#df_onehot=df.copy()
```

```
#convert only categorical variables/features to dummy/one-hot features
#df_onehot = pd.get_dummies(df, columns=cat_cols, prefix = cat_cols)
#print the dataset
#df_onehot
```

```
#Will do Unigram and Bigram for 'text'
```

Pandas Profiling

```
#!pip install pandas-profiling

```

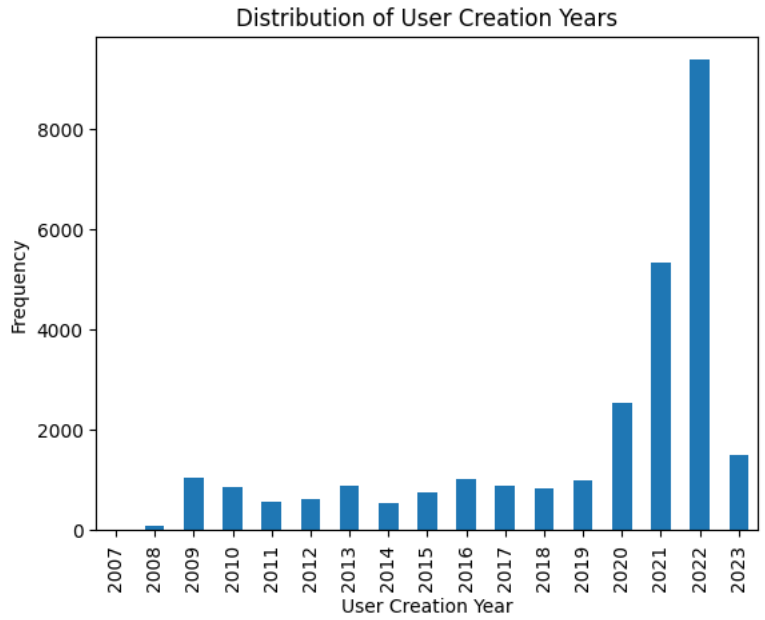
```
#from pandas_profiling import ProfileReport
#profile = ProfileReport(df)
```

```
#profile.to_notebook_iframe()
```

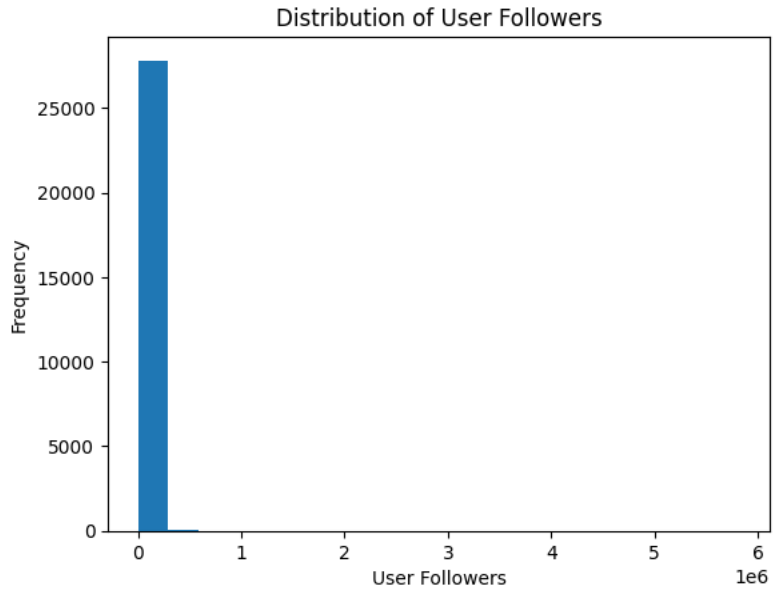
Univariate Analysis

```
import matplotlib.pyplot as plt
# Univariate analysis for user_created
df['user_created'] = pd.to_datetime(df['user_created'])
df['user_created_year'] = df['user_created'].dt.year
user_created_counts = df['user_created_year'].value_counts().sort_index()
user_created_counts.plot(kind='bar')
plt.xlabel('User Creation Year')
plt.ylabel('Frequency')
plt.title('Distribution of User Creation Years')
plt.show()

# Univariate analysis for user_followers
user_followers_stats = df['user_followers'].describe()
print(user_followers_stats)

plt.hist(df['user_followers'], bins=20)
plt.xlabel('User Followers')
plt.ylabel('Frequency')
plt.title('Distribution of User Followers')
plt.show()

# Univariate analysis for user_friends
user_friends_stats = df['user_friends'].describe()
print(user_friends_stats)

plt.hist(df['user_friends'], bins=20)
plt.xlabel('User Friends')
plt.ylabel('Frequency')
plt.title('Distribution of User Friends')
plt.show()

# Univariate analysis for user_favourites
user_favourites_stats = df['user_favourites'].describe()
print(user_favourites_stats)

plt.hist(df['user_favourites'], bins=20)
plt.xlabel('User Favorites')
plt.ylabel('Frequency')
plt.title('Distribution of User Favorites')
plt.show()

# Univariate analysis for user_verified
user_verified_counts = df['user_verified'].value_counts()
user_verified_counts.plot(kind='bar')
plt.xlabel('User Verified')
plt.ylabel('Frequency')
plt.title('Distribution of User Verification')
plt.show()

# Univariate analysis for date
```

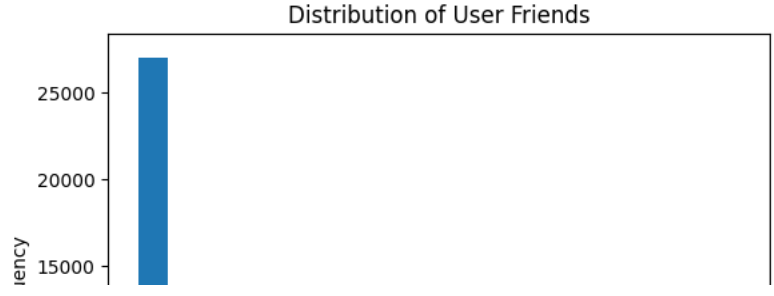
```
df['date'] = pd.to_datetime(df['date'])
df['date_year'] = df['date'].dt.year
date_counts = df['date_year'].value_counts().sort_index()
date_counts.plot(kind='bar')
plt.xlabel('Year')
plt.ylabel('Frequency')
plt.title('Distribution of Tweets over Time')
plt.show()

# Univariate analysis for text
df['text_length'] = df['text'].apply(lambda x: len(x))
text_length_stats = df['text_length'].describe()
print(text_length_stats)

plt.hist(df['text_length'], bins=20)
plt.xlabel('Text Length')
plt.ylabel('Frequency')
plt.title('Distribution of Text Lengths')
plt.show()

# Univariate analysis for hashtags - Will Perform word cloud analysis
```

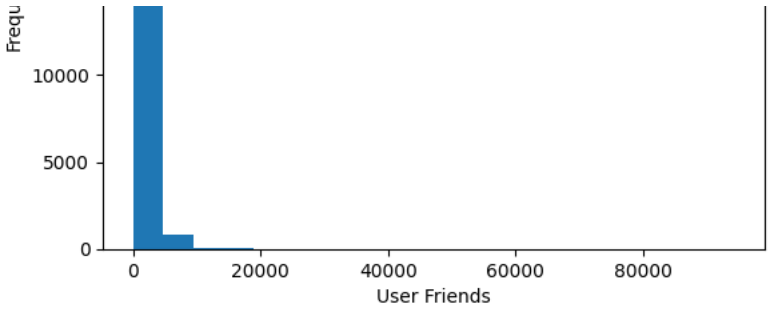



```
count    2.794200e+04
mean     8.843726e+03
std      9.777185e+04
min      0.000000e+00
25%      1.130000e+02
50%      5.430000e+02
75%      1.852000e+03
max      5.825939e+06
Name: user_followers, dtype: float64
```

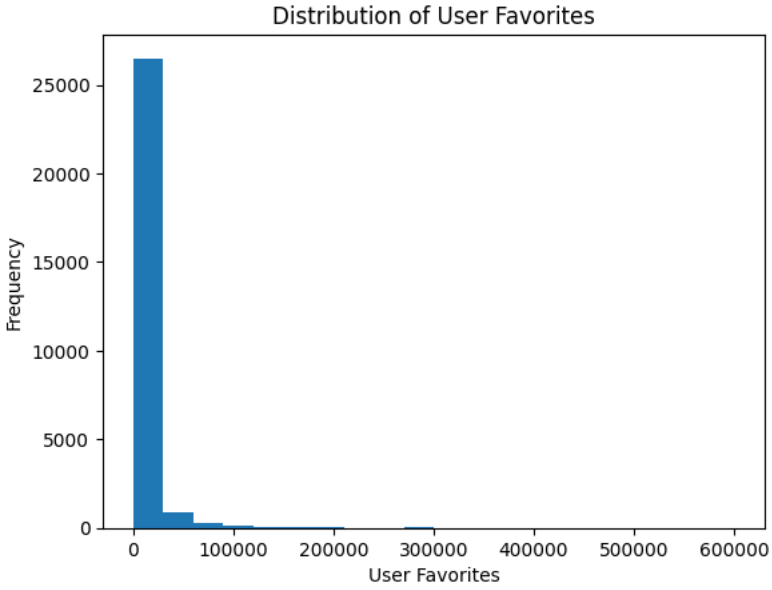


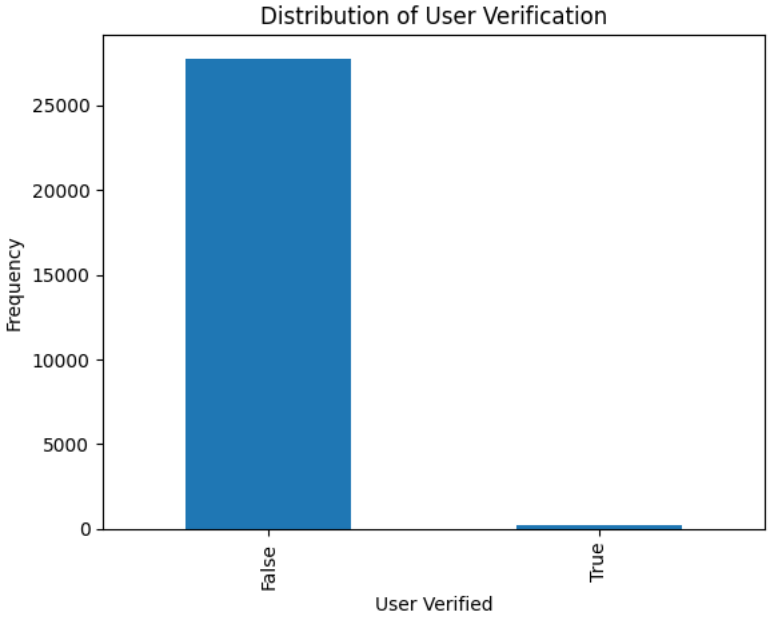

```
count    27942.000000
mean       722.446353
std       2094.308428
min          0.000000
25%          9.000000
50%        121.000000
75%        606.000000
max      94353.000000
Name: user_friends, dtype: float64
```

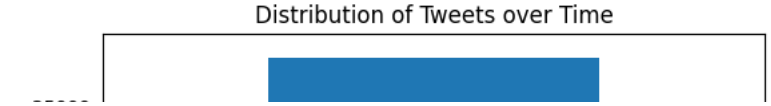





```
count     27942.000000
mean       6427.776358
std       21926.397398
min           0.000000
25%          11.000000
50%         281.000000
75%        3561.750000
max      600534.000000
Name: user_favourites, dtype: float64
```





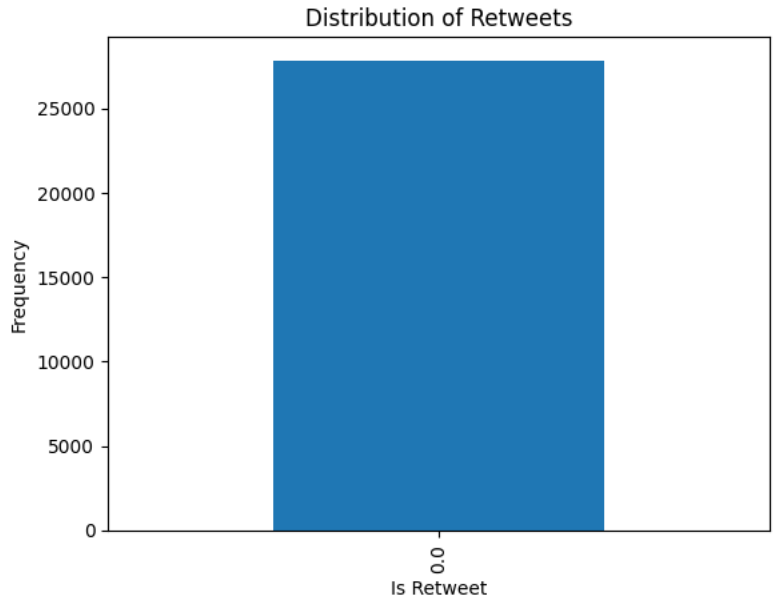



```
1 # Univariate analysis for source
2 #source_counts = df['source'].value_counts()
```

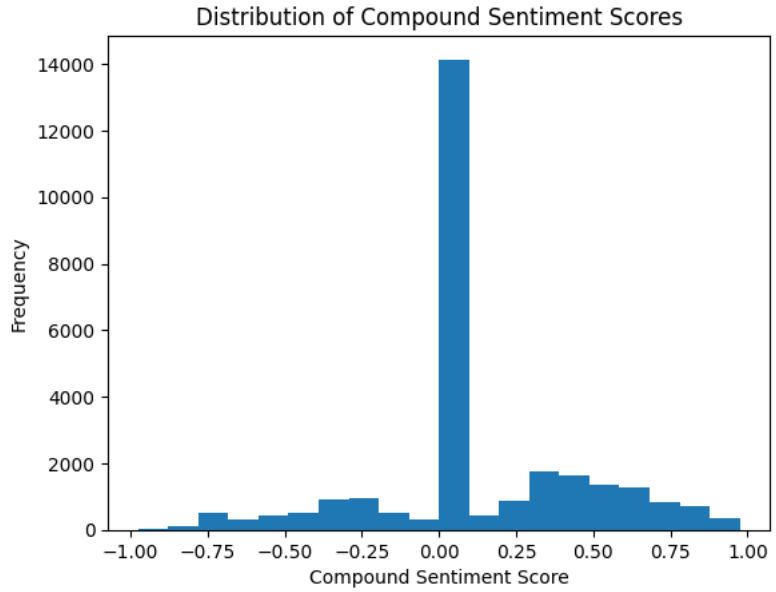
```
#source_counts.plot(kind='bar')
#plt.xlabel('Tweet Source')
#plt.ylabel('Frequency')
#plt.title('Distribution of Tweet Sources')
#plt.show()
```



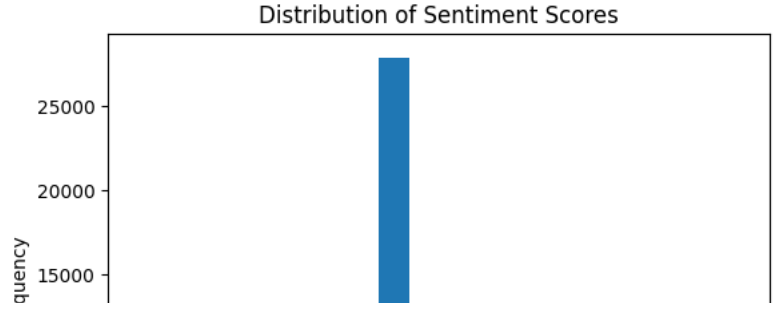
```

# Univariate analysis for is_retweet
is_retweet_counts = df['is_retweet'].value_counts()
is_retweet_counts.plot(kind='bar')
plt.xlabel('Is Retweet')
plt.ylabel('Frequency')
plt.title('Distribution of Retweets')
plt.show()

# Univariate analysis for compound
compound_stats = df['compound'].describe()
print(compound_stats)

plt.hist(df['compound'], bins=20)
plt.xlabel('Compound Sentiment Score')
plt.ylabel('Frequency')
plt.title('Distribution of Compound Sentiment Scores')
plt.show()

# Univariate analysis for score
score_stats = df['score'].describe()
print(score_stats)

plt.hist(df['score'], bins=20)
plt.xlabel('Sentiment Score')
plt.ylabel('Frequency')
plt.title('Distribution of Sentiment Scores')
plt.show()

# Univariate analysis for sentiment_level
sentiment_level_counts = df['sentiment_level'].value_counts()
sentiment_level_counts.plot(kind='bar')
plt.xlabel('Sentiment Level')
plt.ylabel('Frequency')
plt.title('Distribution of Sentiment Levels')
plt.show()

# Univariate analysis for polarity
polarity_stats = df['polarity'].describe()
print(polarity_stats)

plt.hist(df['polarity'], bins=20)
plt.xlabel('Polarity')
plt.ylabel('Frequency')
plt.title('Distribution of Polarity')
plt.show()

```

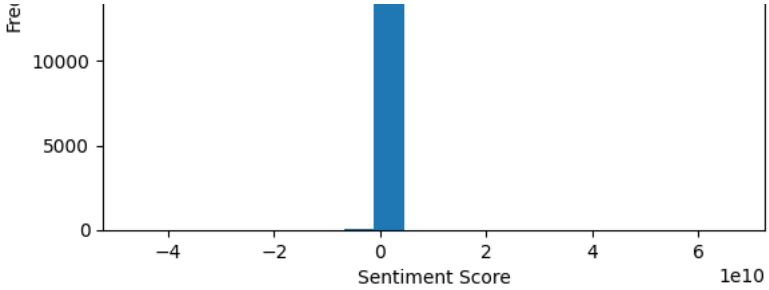


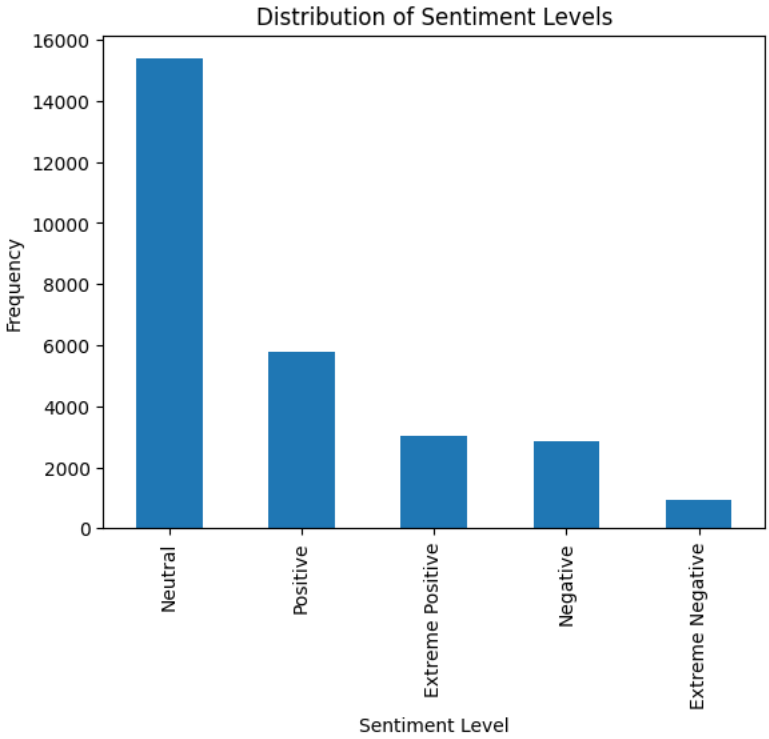

```
count    27942.000000
mean         0.102366
std          0.343487
min         -0.976100
25%          0.000000
50%          0.000000
75%          0.340000
max          0.975300
Name: compound, dtype: float64
```




```
count    2.794200e+04
mean     2.813255e+07
std      1.346742e+09
min     -4.657105e+10
25%      0.000000e+00
50%      0.000000e+00
75%      8.164702e+03
max      6.683636e+10
Name: score, dtype: float64
```







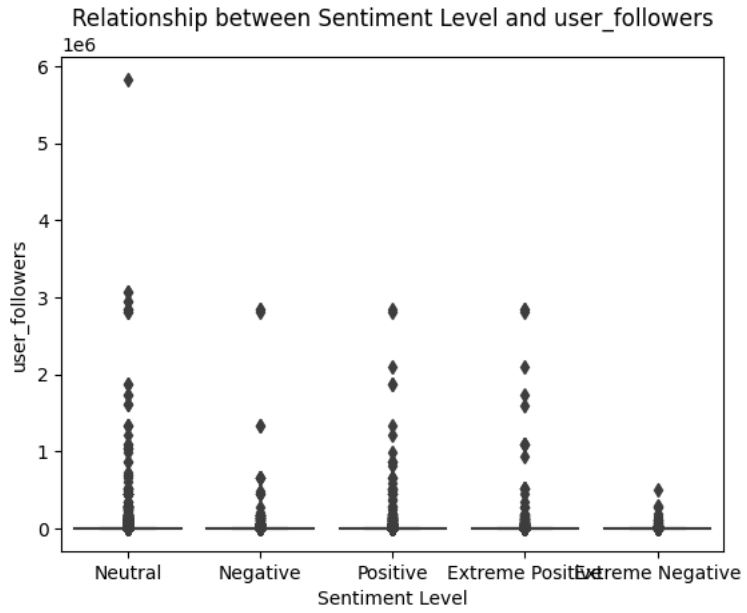

```
count    27942.000000
mean         0.069336
std          0.221637
min         -1.000000
25%          0.000000
50%          0.000000
75%          0.095238
max          1.000000
Name: polarity, dtype: float64
```

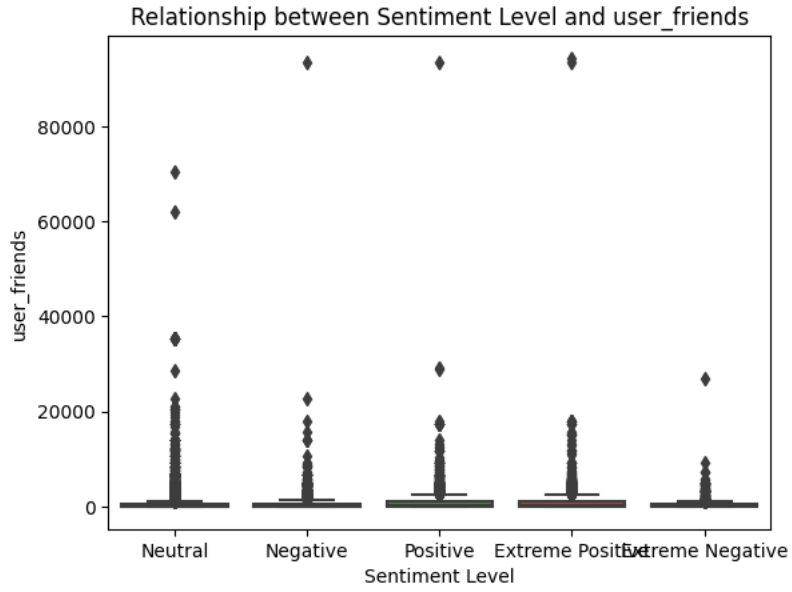
Bivariate Analysis

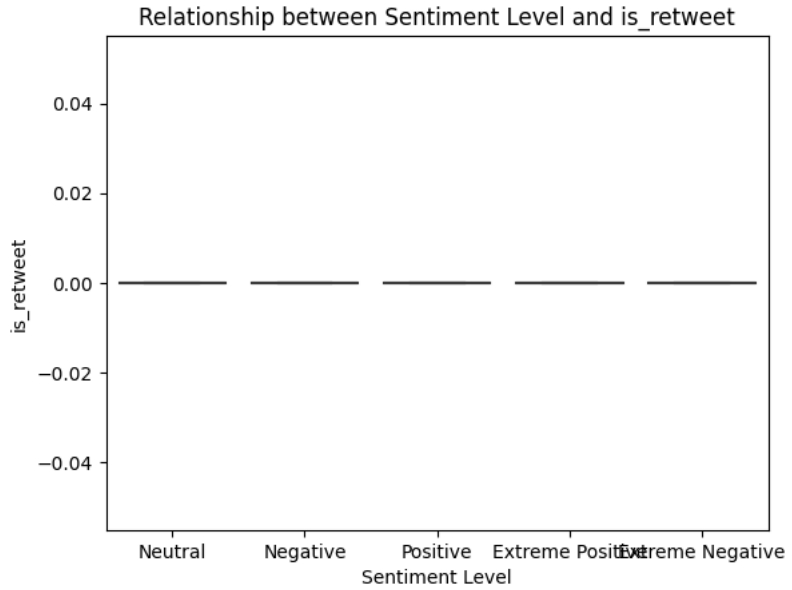
```
#BBariate Analysis of sentiment_level with other attaributes
```

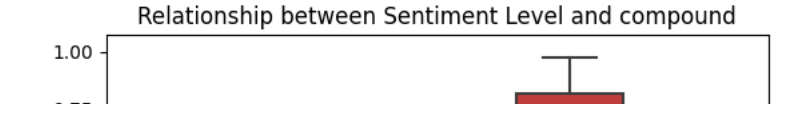
```
import seaborn as sns

# Select pairs of attributes for bivariate analysis
attribute_pairs = [
    ('sentiment_level', 'user_followers'),
    ('sentiment_level', 'user_friends'),
    ('sentiment_level', 'is_retweet'),
    ('sentiment_level', 'compound'),
    ('sentiment_level', 'score'),
    ('sentiment_level', 'polarity'),
    ('sentiment_level', 'subjectivity'),

]

# Perform bivariate analysis for each attribute pair
for attr_x, attr_y in attribute_pairs:
    df['sentiment_level'] = df['sentiment_level'].astype(str)  # Convert sentiment_level to string type
```

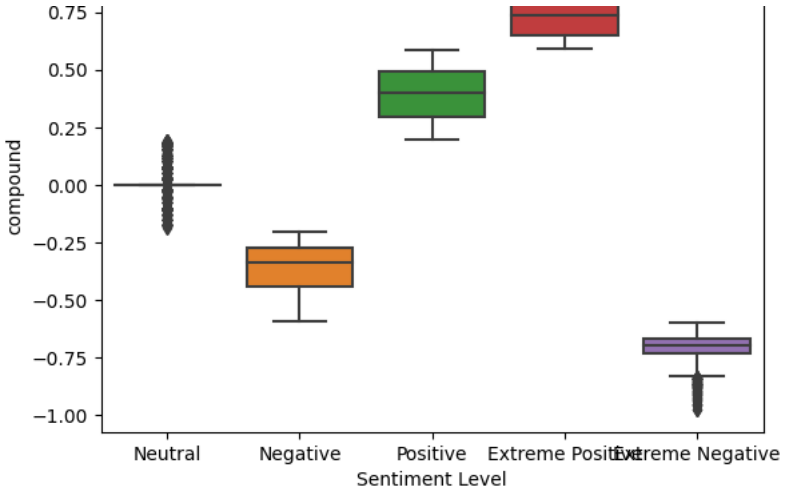
```
    sns.boxplot(x=attr_x, y=attr_y, data=df)
    plt.xlabel('Sentiment Level')
    plt.ylabel(attr_y)
    plt.title(f'Relationship between Sentiment Level and {attr_y}')
    plt.show()

```

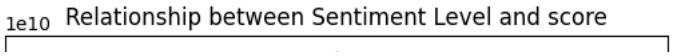

Relationship between Sentiment Level and user_followers
1e6
6
5
4
3
2
1
0
user_followers
Neutral
Negative
Positive
Extreme Positive
Extreme Negative
Sentiment Level


Relationship between Sentiment Level and user_friends
80000
60000
40000
20000
0
user_friends
Neutral
Negative
Positive
Extreme Positive
Extreme Negative
Sentiment Level


Relationship between Sentiment Level and is_retweet
0.04
0.02
0.00
−0.02
−0.04
is_retweet
Neutral
Negative
Positive
Extreme Positive
Extreme Negative
Sentiment Level


Relationship between Sentiment Level and compound
1.00





Data visualization

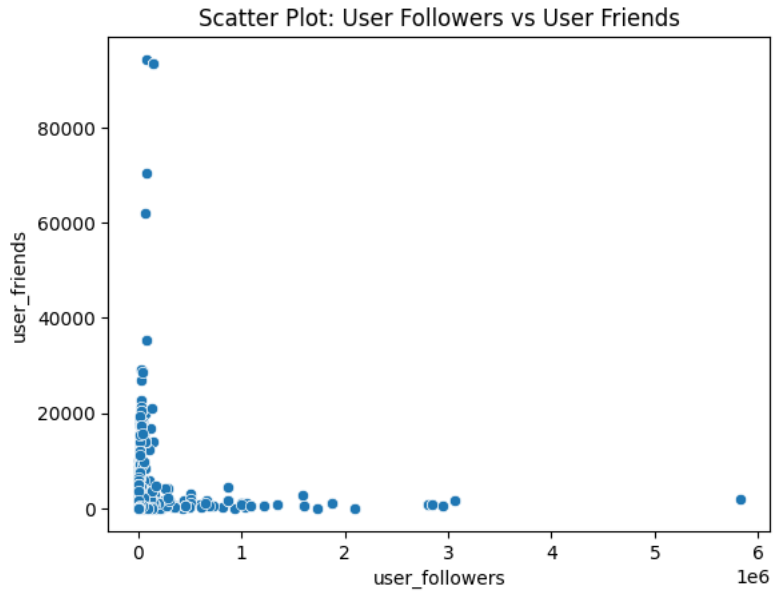
```
!pip install seaborn
!pip install matplotlib

```

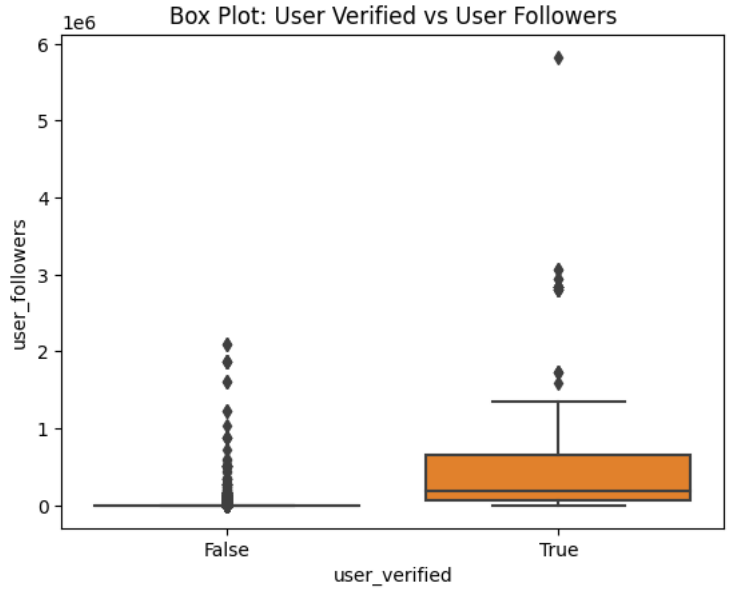
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.22.4)
Requirement already satisfied: pandas>=0.25 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.1
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (8.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.25->seaborn) (2022.7.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1-
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.40.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.22.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (8.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

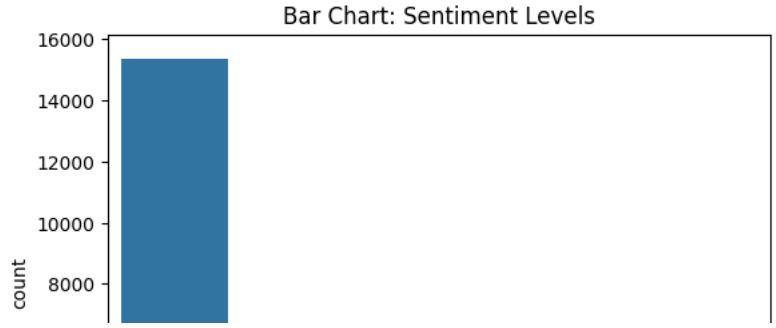
```
#Some Visualtizations of attributes for analysis
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Scatter plot
sns.scatterplot(data=df, x='user_followers', y='user_friends')
plt.title('Scatter Plot: User Followers vs User Friends')
plt.show()

# Box plot
sns.boxplot(data=df, x='user_verified', y='user_followers')
plt.title('Box Plot: User Verified vs User Followers')
plt.show()

# Bar chart
sns.countplot(data=df, x='sentiment_level')
plt.title('Bar Chart: Sentiment Levels')
```

```
plt.show()

# Histogram
sns.histplot(data=df, x='user_favourites', bins=20)
plt.title('Histogram: User Favorites')
plt.show()

# Heatmap
heatmap_data = df[['user_followers', 'user_friends', 'user_favourites', 'compound', 'polarity', 'subjectivity']].corr()
sns.heatmap(heatmap_data, annot=True, cmap='coolwarm')
plt.title('Heatmap: Correlation Matrix')
plt.show()

```

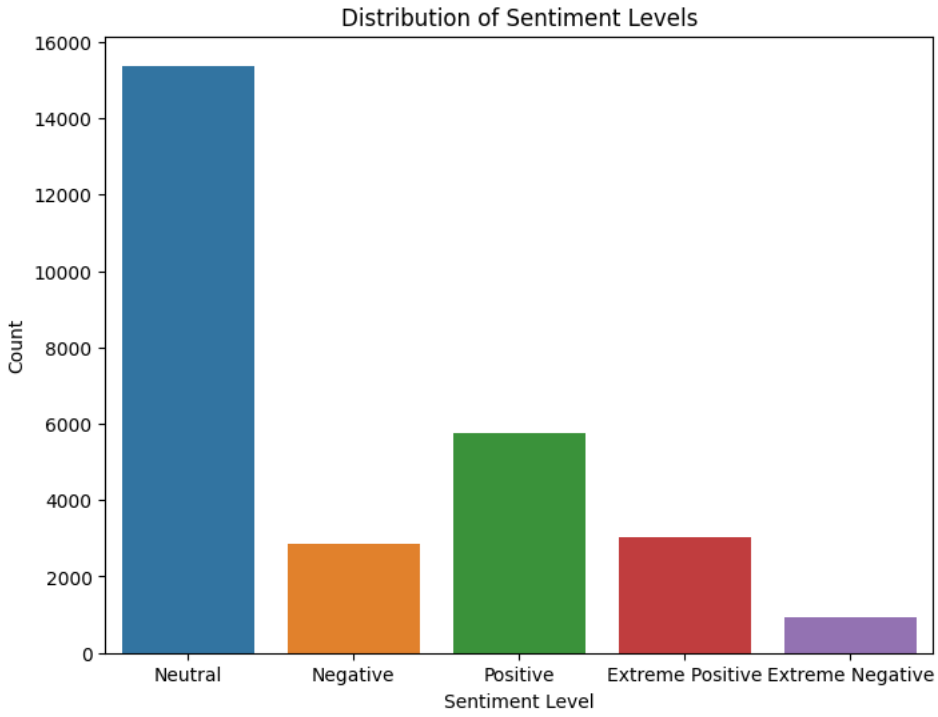


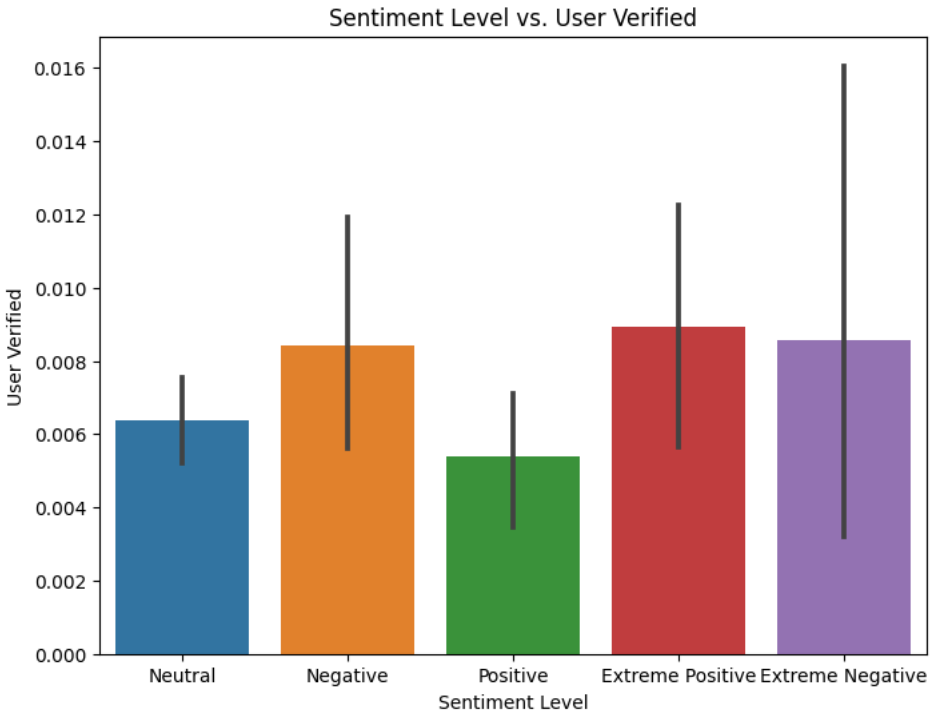


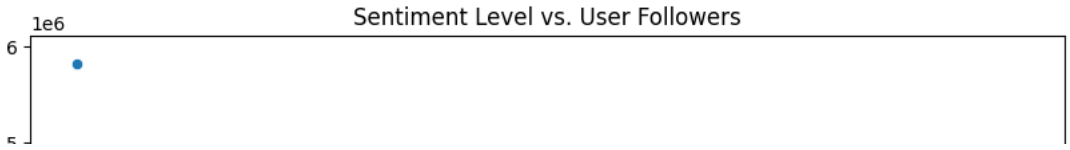



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Convert 'user_verified' column to numeric representation
df['user_verified_numeric'] = df['user_verified'].astype(int)

# EDA for sentiment_level

# Countplot of sentiment_level
plt.figure(figsize=(8, 6))
sns.countplot(x='sentiment_level', data=df)
plt.title('Distribution of Sentiment Levels')
plt.xlabel('Sentiment Level')
plt.ylabel('Count')
plt.show()
```

```
plt.show()

# Relationship with other attributes

# Bar plot of sentiment_level vs. user_verified
plt.figure(figsize=(8, 6))
sns.barplot(x='sentiment_level', y='user_verified_numeric', data=df)
plt.title('Sentiment Level vs. User Verified')
plt.xlabel('Sentiment Level')
plt.ylabel('User Verified')
plt.show()

# Bar plot of sentiment_level vs. user_location
#plt.figure(figsize=(12, 6))
#sns.countplot(x='sentiment_level', hue='user_location', data=df)
#plt.title('Sentiment Level vs. User Location')
#plt.xlabel('Sentiment Level')
#plt.ylabel('Count')
#plt.legend(title='User Location')
#plt.show()

# Scatter plot of sentiment_level vs. user_followers
plt.figure(figsize=(10, 6))
sns.scatterplot(x='sentiment_level', y='user_followers', data=df)
plt.title('Sentiment Level vs. User Followers')
plt.xlabel('Sentiment Level')
plt.ylabel('User Followers')
plt.show()

# Summary statistics of sentiment_level
sentiment_stats = df.groupby('sentiment_level').describe()['user_followers']
print("\nSummary Statistics of User Followers by Sentiment Level:")
print(sentiment_stats)

# Remove the temporary column
df.drop('user_verified_numeric', axis=1, inplace=True)

```







Feature engineering

Scaling and Encoding

```
1 import pandas as pd
2 from sklearn.preprocessing import MinMaxScaler, LabelEncoder
3
4 # Creating interaction variables
5 df['interaction_var'] = df['user_followers'] * df['user_friends']
```

```
# Scaling or normalizing variables
scaler = MinMaxScaler()
df['user_followers_scaled'] = scaler.fit_transform(df[['user_followers']])
df['user_friends_scaled'] = scaler.fit_transform(df[['user_friends']])

# Binning variables
df['user_favourites_bins'] = pd.cut(df['user_favourites'], bins=[0, 100, 1000, float('inf')], labels=['low', 'medium', 'high'])

# Encoding categorical variables
label_encoder = LabelEncoder()
df['sentiment_level_encoded'] = label_encoder.fit_transform(df['sentiment_level'])

# Display the updated DataFrame
print(df.head())

```

```
                user_name user_location  \
0  Bitcoin Treasures ð□□□  Bitcoin City
1          Bitcoin Price    Order Book
2              OPQRSTEE         Miami
3       BNB Price Tracker           NaN
4             Frases_k3_           NaN

                                  user_description         user_created  \
0  I am a bot tracking the USD value of various b...  2020-08-23 16:20:00
1  $BTC price updated every 4 hours\n\nPrices in ...  2022-03-25 08:31:00
2                                              NaN  2019-01-22 07:36:00
3            #BNB #BNBTracker #BinanceCoin #Binance  2022-07-11 05:24:00
4                                              NaN  2011-02-25 15:56:00

   user_followers  user_friends  user_favourites  user_verified  \
0          2603.0           8.0             48.0          False
1            16.0           0.0              2.0          False
2            47.0         169.0             65.0          False
3           483.0           5.0              0.0          False
4           267.0         754.0            727.0          False

                  date                                               text  \
0  2023-03-04 20:00:00  cynthia lummi bitcoin treasur worth36279 usd 5...
1  2023-03-04 20:00:00  btc price sat 04 mar 2023 200042 gmt 222579 ð□...
2  2023-03-04 20:00:00  check product ð□□□ hulk logo men premium tshir...
3  2023-03-04 20:00:00  binancecoin price updat bnb bnb 28882 usd bitc...
4  2023-03-04 20:00:00                            know peopl say bitcoin

   ... compound    score  sentiment_level  polarity  subjectivity  \
0  ...   0.0000   0.0000          Neutral      0.00          0.00
1  ...  -0.2732 -13.9332         Negative      0.08          0.42
2  ...   0.0000   0.0000          Neutral     -0.40          0.40
3  ...   0.0000   0.0000          Neutral      0.00          0.25
4  ...   0.0000   0.0000          Neutral      0.00          0.00

  interaction_var  user_followers_scaled  user_friends_scaled  \
0         20824.0               0.000447             0.000085
1             0.0               0.000003             0.000000
2          7943.0               0.000008             0.001791
3          2415.0               0.000083             0.000053
4        201318.0               0.000046             0.007991

   user_favourites_bins  sentiment_level_encoded
0                   low                        3
1                   low                        2
2                   low                        3
3                   NaN                        3
4                medium                        3

[5 rows x 23 columns]
```

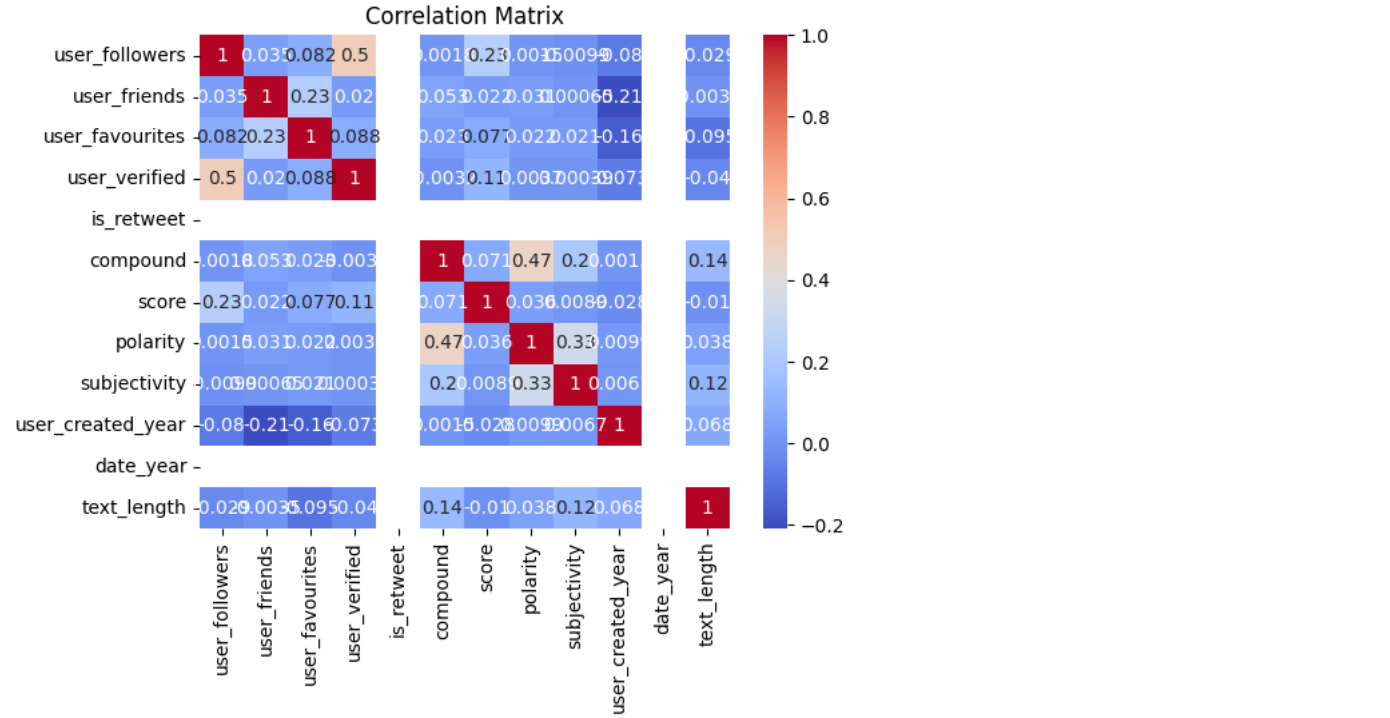
Correlation analysis

```
#Correlation Matrix for all attributes
```

```
import seaborn as sns

# Generate correlation matrix
correlation_matrix = df.corr()

# Create heatmap of correlation matrix
```

```
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")

# Display the plot
plt.title("Correlation Matrix")
plt.show()

```

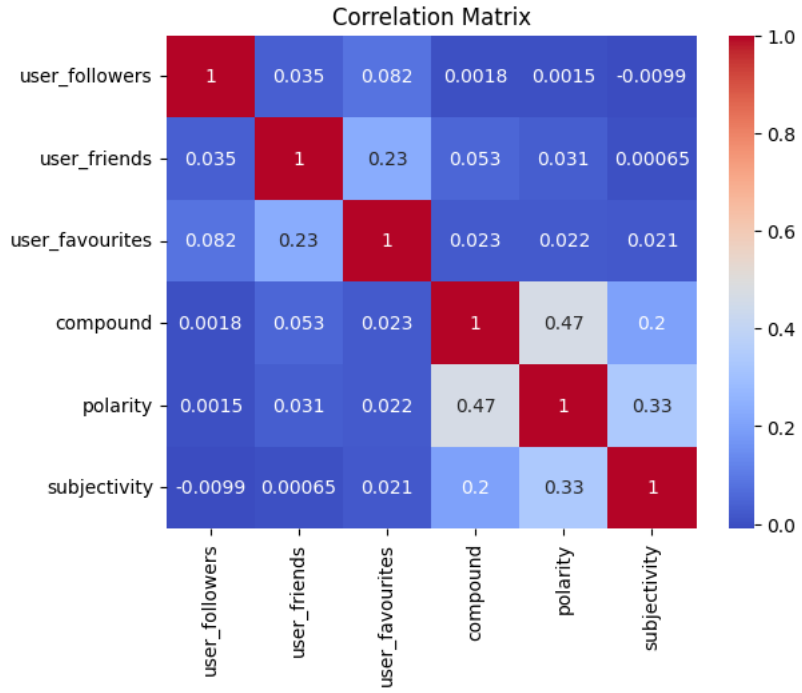
```
<ipython-input-22-dae9ae9f4ad1>:4: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future versio
  correlation_matrix = df.corr()
```

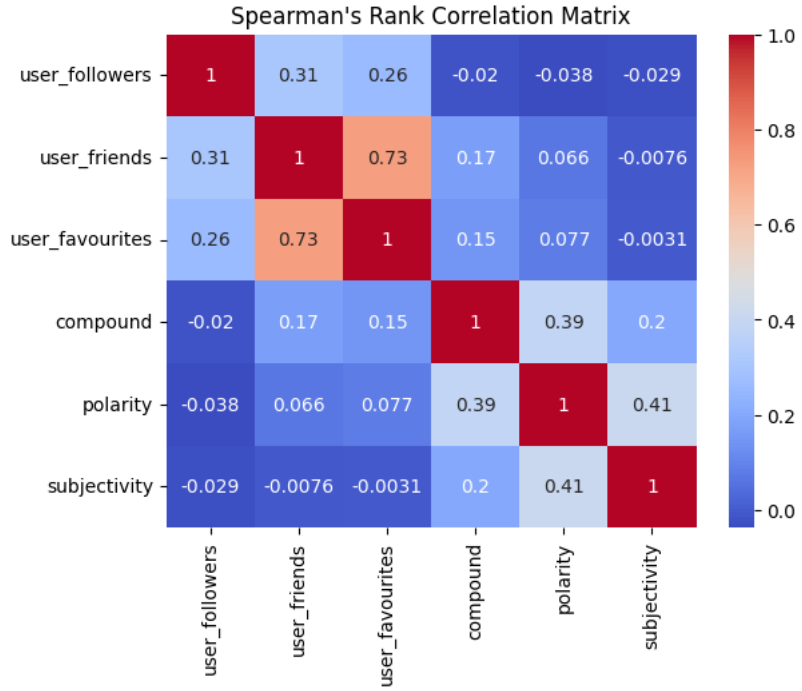



```
#Correlation Matrix for Selected features
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Select the relevant columns for correlation analysis
selected_columns = ['user_followers', 'user_friends', 'user_favourites', 'compound', 'polarity', 'subjectivity']
correlation_data = df[selected_columns]

# Calculate correlation matrix
correlation_matrix = correlation_data.corr()

# Visualize correlation matrix as a heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

# Calculate Spearman's rank correlation matrix
spearman_corr_matrix = correlation_data.corr(method='spearman')

# Visualize Spearman's rank correlation matrix as a heatmap
sns.heatmap(spearman_corr_matrix, annot=True, cmap='coolwarm')
plt.title("Spearman's Rank Correlation Matrix")
plt.show()

# Identify highly correlated variables
threshold = 0.7  # Set the correlation threshold as per your requirement
highly_correlated_pairs = set()
for i in range(len(correlation_matrix.columns)):
    for j in range(i):
        if abs(correlation_matrix.iloc[i, j]) >= threshold:
            col_i = correlation_matrix.columns[i]
            col_j = correlation_matrix.columns[j]
            highly_correlated_pairs.add((col_i, col_j))

```

```
print("Highly Correlated Variable Pairs:")
for pair in highly_correlated_pairs:
    print(pair)

```






```
Highly Correlated Variable Pairs:
```

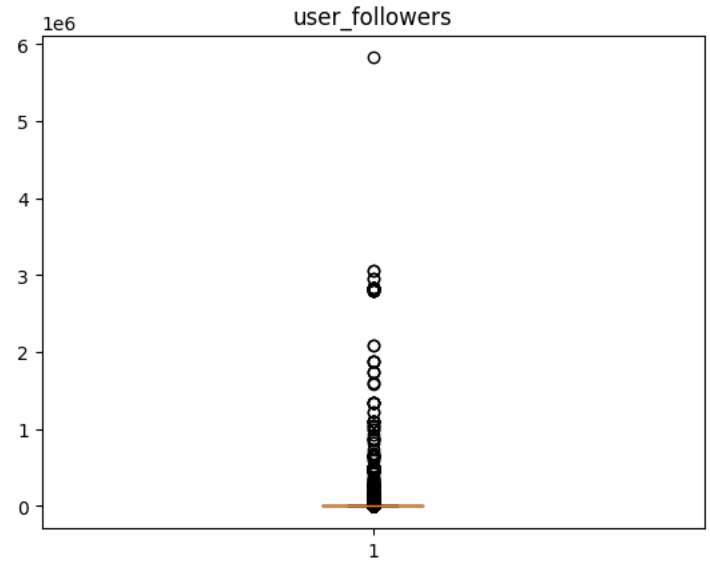
Outlier and anomaly detection

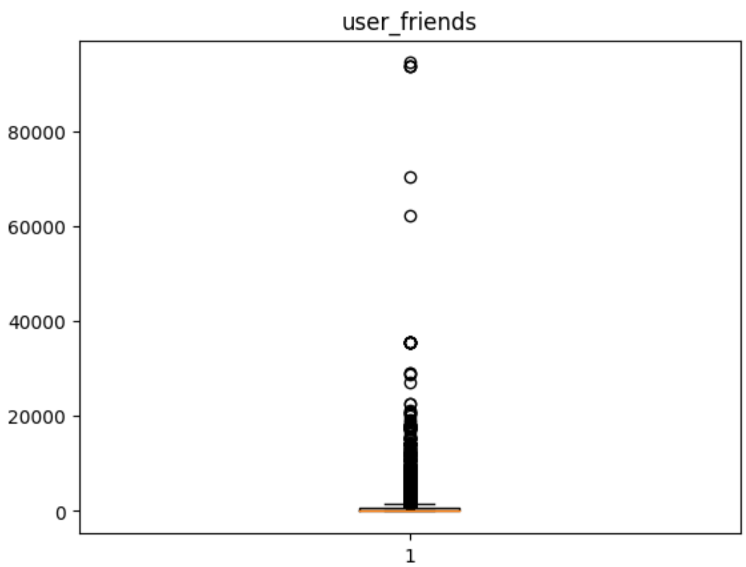
```
#Outliers in categorical or non-numeric columns might not be meaningful or applicable in the same way as with numerical data
```

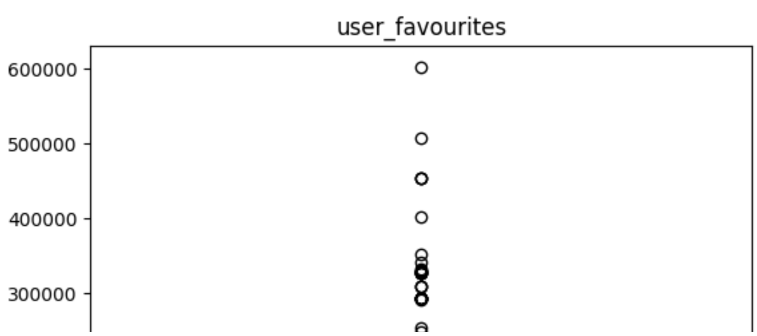
```
import pandas as pd
import matplotlib.pyplot as plt

# Select numerical columns for outlier analysis
numerical_columns = df.select_dtypes(include='number').columns

# Compute descriptive statistics for numerical columns
```

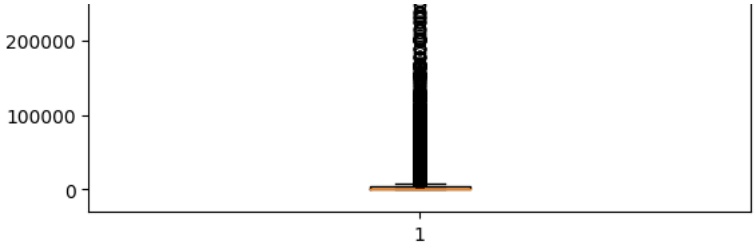
```
df_describe = df[numerical_columns].describe()

# Identify and store outliers
outliers = pd.DataFrame(columns=numerical_columns)
for column in numerical_columns:
    q1 = df_describe.loc['25%', column]
    q3 = df_describe.loc['75%', column]
    iqr = q3 - q1
    threshold = 1.5 * iqr
    column_outliers = df[(df[column] < q1 - threshold) | (df[column] > q3 + threshold)]
    outliers = pd.concat([outliers, column_outliers])

# Visualize outliers
for column in numerical_columns:
    plt.figure()
    plt.boxplot(df[column])
    plt.title(column)

plt.show()

```

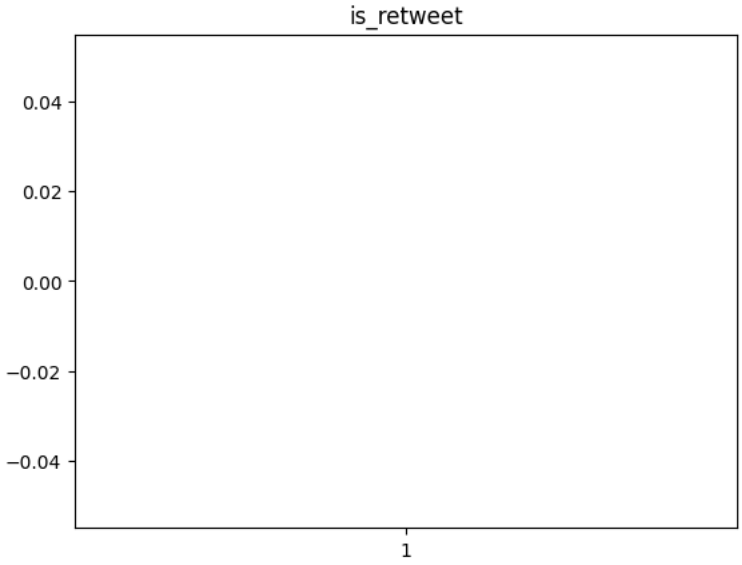
```
<ipython-input-20-245c3cb061a7>:18: FutureWarning: In a future version, object-dtype columns with all-bool values will not be included
  outliers = pd.concat([outliers, column_outliers])
<ipython-input-20-245c3cb061a7>:18: FutureWarning: In a future version, object-dtype columns with all-bool values will not be included
  outliers = pd.concat([outliers, column_outliers])
<ipython-input-20-245c3cb061a7>:18: FutureWarning: In a future version, object-dtype columns with all-bool values will not be included
  outliers = pd.concat([outliers, column_outliers])
<ipython-input-20-245c3cb061a7>:18: FutureWarning: In a future version, object-dtype columns with all-bool values will not be included
  outliers = pd.concat([outliers, column_outliers])
<ipython-input-20-245c3cb061a7>:18: FutureWarning: In a future version, object-dtype columns with all-bool values will not be included
  outliers = pd.concat([outliers, column_outliers])
<ipython-input-20-245c3cb061a7>:18: FutureWarning: In a future version, object-dtype columns with all-bool values will not be included
  outliers = pd.concat([outliers, column_outliers])
<ipython-input-20-245c3cb061a7>:18: FutureWarning: In a future version, object-dtype columns with all-bool values will not be included
  outliers = pd.concat([outliers, column_outliers])
<ipython-input-20-245c3cb061a7>:18: FutureWarning: In a future version, object-dtype columns with all-bool values will not be included
  outliers = pd.concat([outliers, column_outliers])
<ipython-input-20-245c3cb061a7>:18: FutureWarning: In a future version, object-dtype columns with all-bool values will not be included
  outliers = pd.concat([outliers, column_outliers])
<ipython-input-20-245c3cb061a7>:18: FutureWarning: In a future version, object-dtype columns with all-bool values will not be included
  outliers = pd.concat([outliers, column_outliers])
```

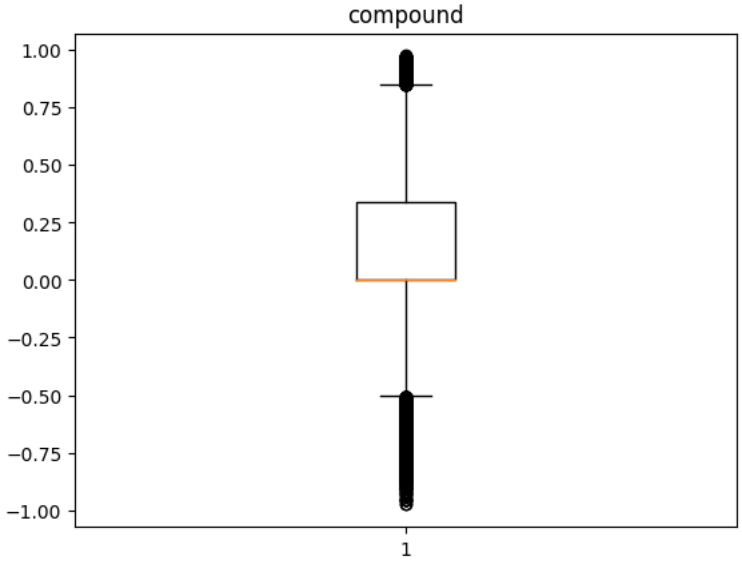


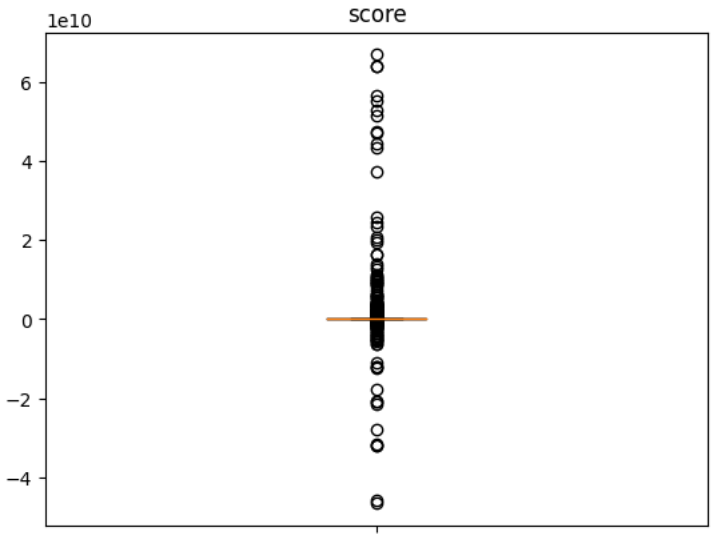




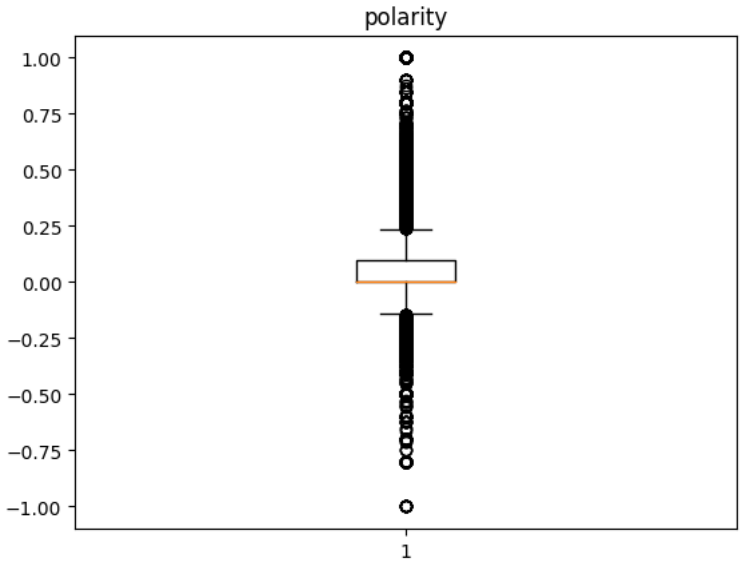

200000
100000
0
1

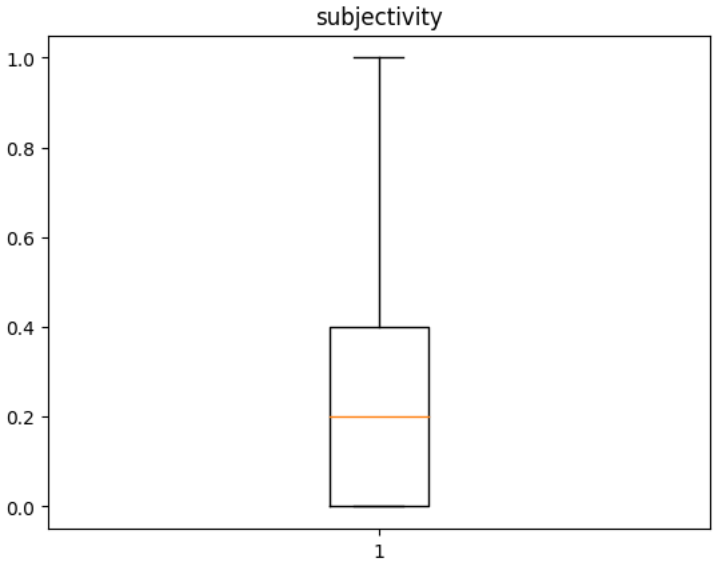

is_retweet
0.04
0.02
0.00
−0.02
−0.04
1

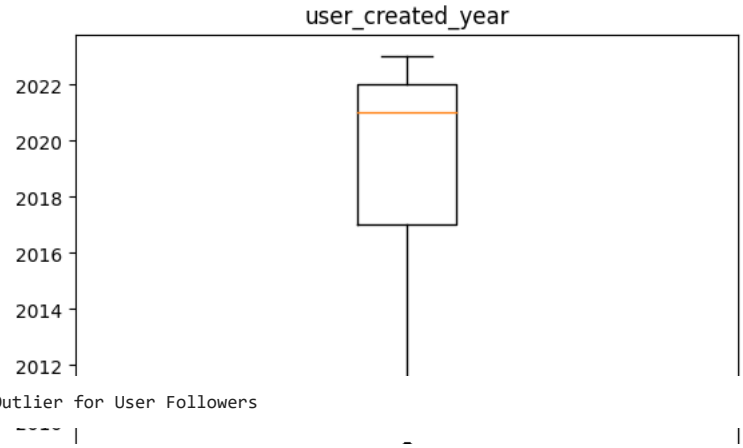

compound
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
1


score
1e10
6
4
2
0
−2
−4

1








```
1 #Outlier for User Followers
```

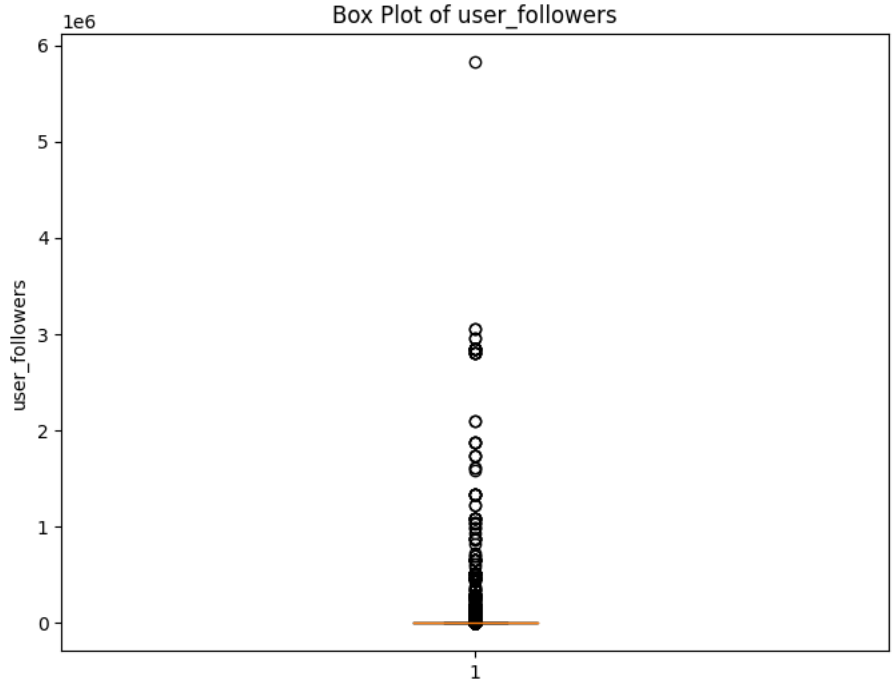
```
1 import pandas as pd
2 import numpy as np
3
4 # Detect outliers using z-score
5 def detect_outliers_zscore(data, threshold=3):
6     z_scores = (data - data.mean()) / data.std()
7     return np.abs(z_scores) > threshold
8
9 # Detect outliers using IQR (Interquartile Range)
10 def detect_outliers_iqr(data, k=1.5):
11     Q1 = data.quantile(0.25)
```

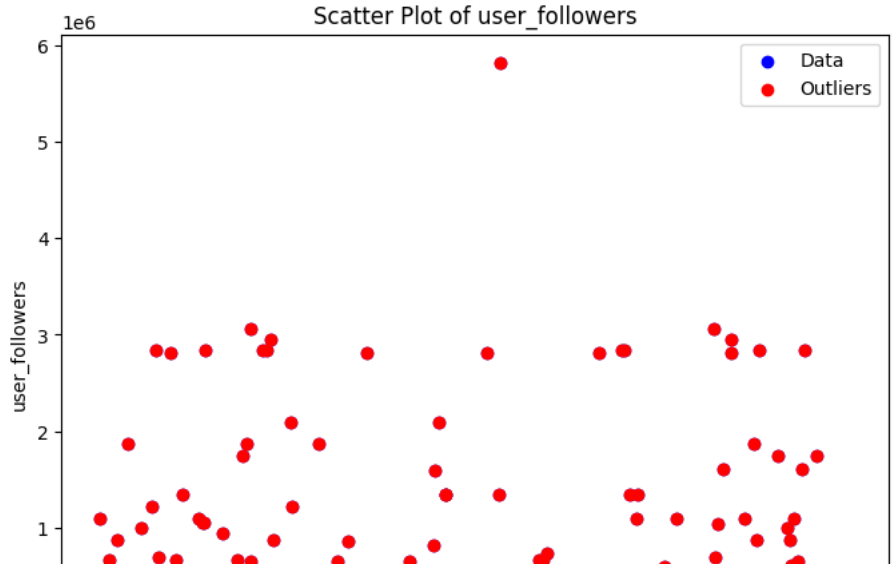
```
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - k * IQR
    upper_bound = Q3 + k * IQR
    return (data < lower_bound) | (data > upper_bound)

# Apply outlier detection on a specific column
outliers_zscore = detect_outliers_zscore(df['user_followers'])
outliers_iqr = detect_outliers_iqr(df['user_followers'])

# Display the outliers
outliers_zscore_indices = df[outliers_zscore].index
outliers_iqr_indices = df[outliers_iqr].index

print("Outliers detected using Z-score:")
print(outliers_zscore_indices)

print("Outliers detected using IQR:")
print(outliers_iqr_indices)

```

```
Outliers detected using Z-score:
Int64Index([   22,   364,   666,  1010,  1079,  1379,  1523,  1542,  1671,
             1935,
             ...
             25448, 25536, 25670, 25723, 25814, 25962, 26053, 26126, 26190,
             26641],
            dtype='int64', length=116)
Outliers detected using IQR:
Int64Index([   17,    22,    35,    40,    41,    42,    46,    51,    66,
               77,
             ...
             27881, 27886, 27902, 27903, 27913, 27917, 27925, 27930, 27934,
             27937],
            dtype='int64', length=3730)
```

```
      |                              |                              |
```

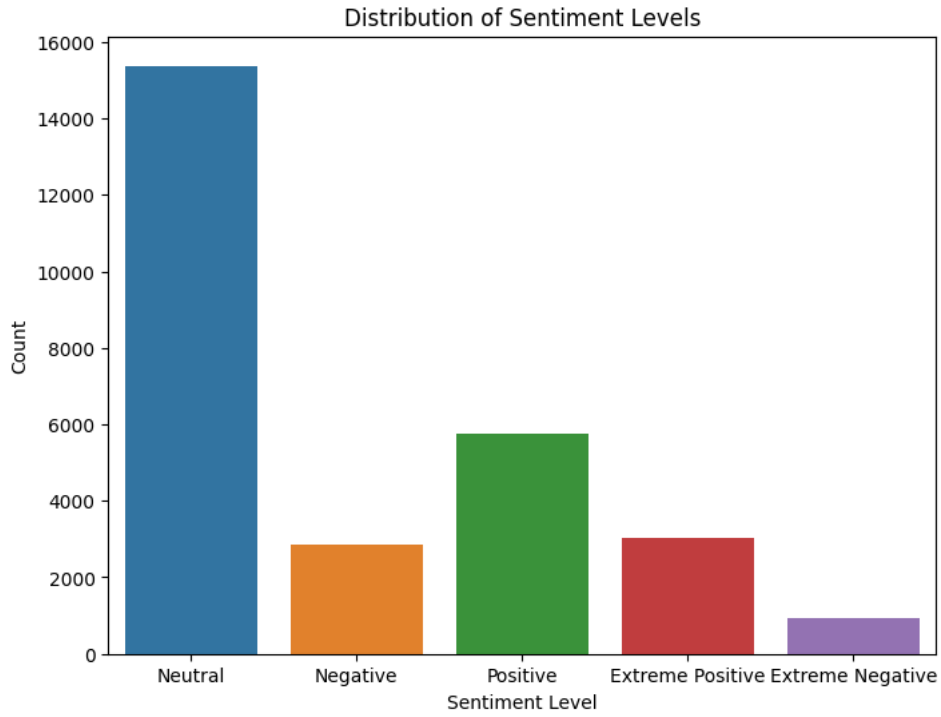
```
import pandas as pd
import matplotlib.pyplot as plt

# Select the column for outlier detection
column_name = 'user_followers'

# Detect outliers using z-score
def detect_outliers_zscore(data, threshold=3):
    z_scores = (data - data.mean()) / data.std()
    return z_scores.abs() > threshold

# Apply outlier detection on the specified column
outliers_zscore = detect_outliers_zscore(df[column_name])

# Get the outliers indices
outliers_indices = df[outliers_zscore].index

# Create a box plot
plt.figure(figsize=(8, 6))
plt.boxplot(df[column_name])
plt.title('Box Plot of {}'.format(column_name))
plt.ylabel(column_name)
plt.show()

# Create a scatter plot highlighting the outliers
plt.figure(figsize=(8, 6))
plt.scatter(df.index, df[column_name], color='blue', label='Data')
plt.scatter(outliers_indices, df.loc[outliers_indices, column_name], color='red', label='Outliers')
plt.title('Scatter Plot of {}'.format(column_name))
plt.xlabel('Index')
plt.ylabel(column_name)
plt.legend()
plt.show()

```

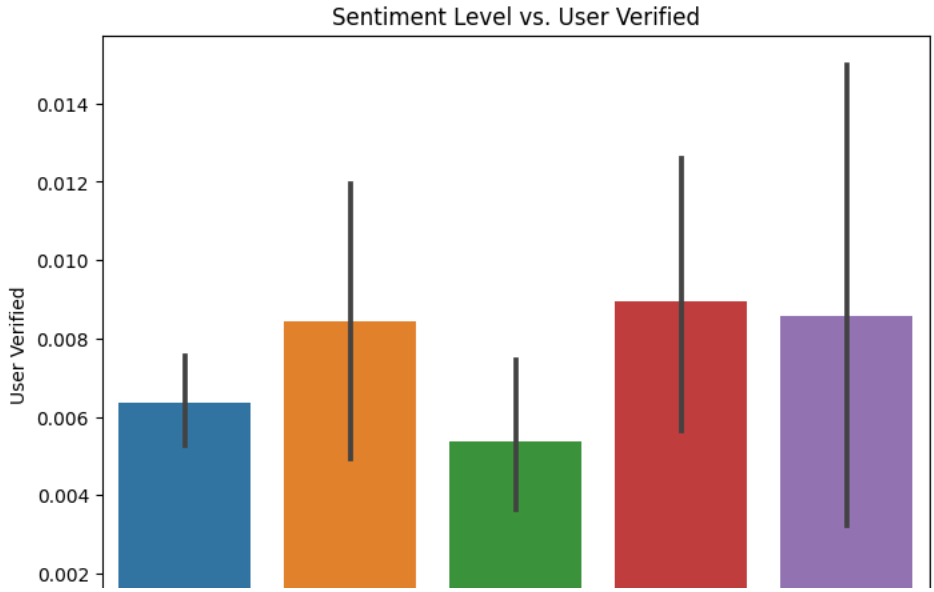





```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Convert 'user_verified' column to numeric representation
df['user_verified_numeric'] = df['user_verified'].astype(int)

# EDA for sentiment_level

# Countplot of sentiment_level
plt.figure(figsize=(8, 6))
sns.countplot(x='sentiment_level', data=df)
plt.title('Distribution of Sentiment Levels')
plt.xlabel('Sentiment Level')
plt.ylabel('Count')
plt.show()

# Relationship with other attributes

# Bar plot of sentiment_level vs. user_verified
plt.figure(figsize=(8, 6))
sns.barplot(x='sentiment_level', y='user_verified_numeric', data=df)
plt.title('Sentiment Level vs. User Verified')
plt.xlabel('Sentiment Level')
plt.ylabel('User Verified')
plt.show()
```

```
27
28 # Bar plot of sentiment_level vs. user_location
29 #plt.figure(figsize=(12, 6))
30 #sns.countplot(x='sentiment_level', hue='user_location', data=df)
31 #plt.title('Sentiment Level vs. User Location')
32 #plt.xlabel('Sentiment Level')
33 #plt.ylabel('Count')
34 #plt.legend(title='User Location')
35 #plt.show()
36
37 # Scatter plot of sentiment_level vs. user_followers
38 plt.figure(figsize=(10, 6))
39 sns.scatterplot(x='sentiment_level', y='user_followers', data=df)
40 plt.title('Sentiment Level vs. User Followers')
41 plt.xlabel('Sentiment Level')
42 plt.ylabel('User Followers')
43 plt.show()
44
45 # Summary statistics of sentiment_level
46 sentiment_stats = df.groupby('sentiment_level').describe()['user_followers']
47 print("\nSummary Statistics of User Followers by Sentiment Level:")
48 print(sentiment_stats)
49
50 # Remove the temporary column
51 df.drop('user_verified_numeric', axis=1, inplace=True)
52
```





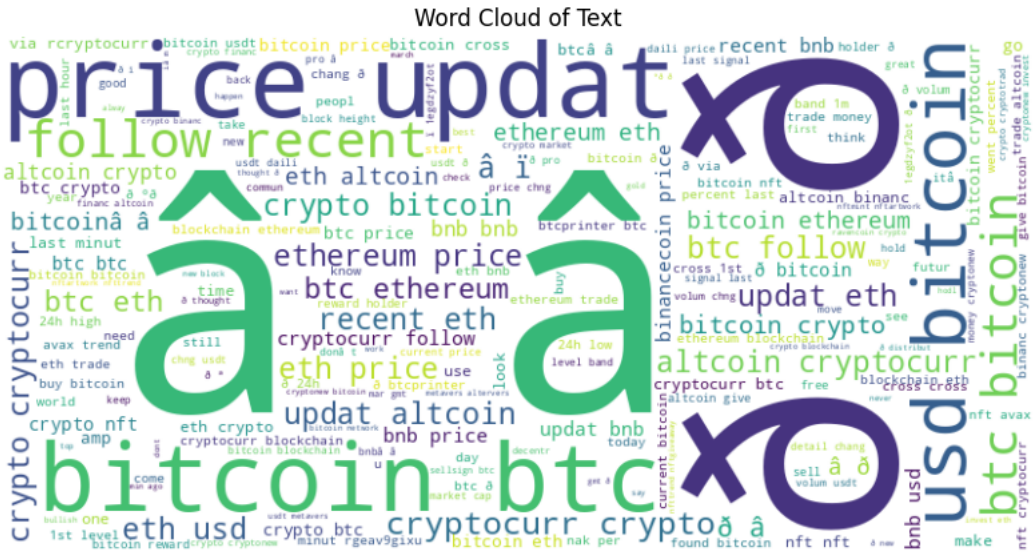

```
1 #Word Cloud
```

```
1 !pip install wordcloud
2
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: wordcloud in /usr/local/lib/python3.10/dist-packages (1.8.2.2)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.10/dist-packages (from wordcloud) (1.22.4)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from wordcloud) (8.4.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from wordcloud) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (4.40.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (23.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (3.1.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (
```

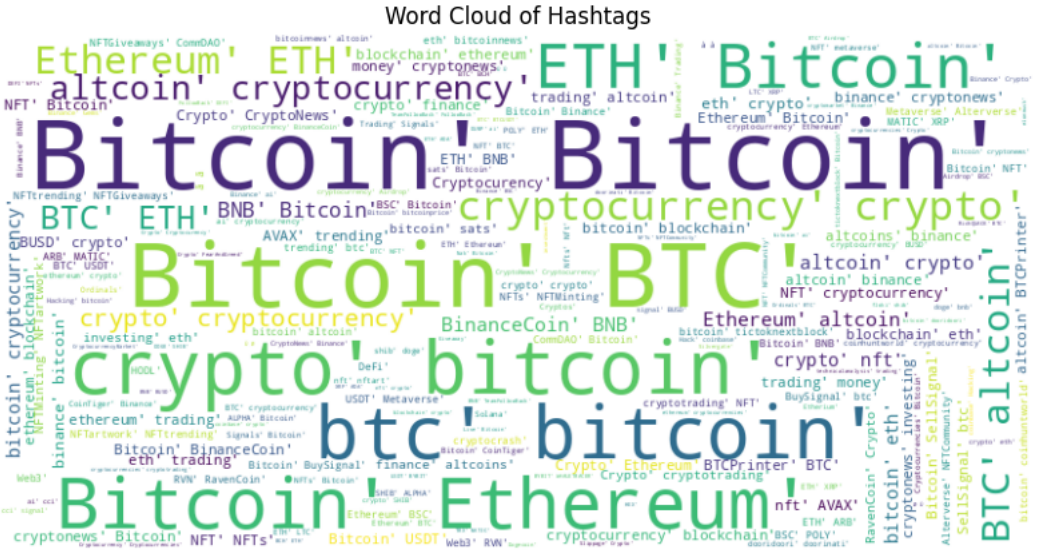
Word Cloud

```
from wordcloud import WordCloud

# Concatenate all the text data
text_data = ' '.join(df['text'].astype(str))

# Create a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text_data)

# Plot the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Text')
plt.show()

```




```
from wordcloud import WordCloud

# Concatenate all the user location data
user_location_data = ' '.join(df['user_location'].astype(str))

# Create a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(user_location_data)

# Plot the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of User Locations')
plt.show()

```


```
from wordcloud import WordCloud

# Concatenate all the hashtags data
hashtags_data = ' '.join(df['hashtags'].astype(str))

# Create a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(hashtags_data)

# Plot the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Hashtags')
plt.show()

```




```
from wordcloud import WordCloud

# Create a dictionary with sentiment levels and their counts
label_counts = {
    'Neutral': 93169,
    'Positive': 35921,
    'Extreme Positive': 17343,
    'Negative': 15903,
    'Extreme Negative': 5316
}

# Generate the word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(label_counts)

# Plot the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Sentiment Levels')
plt.show()

```

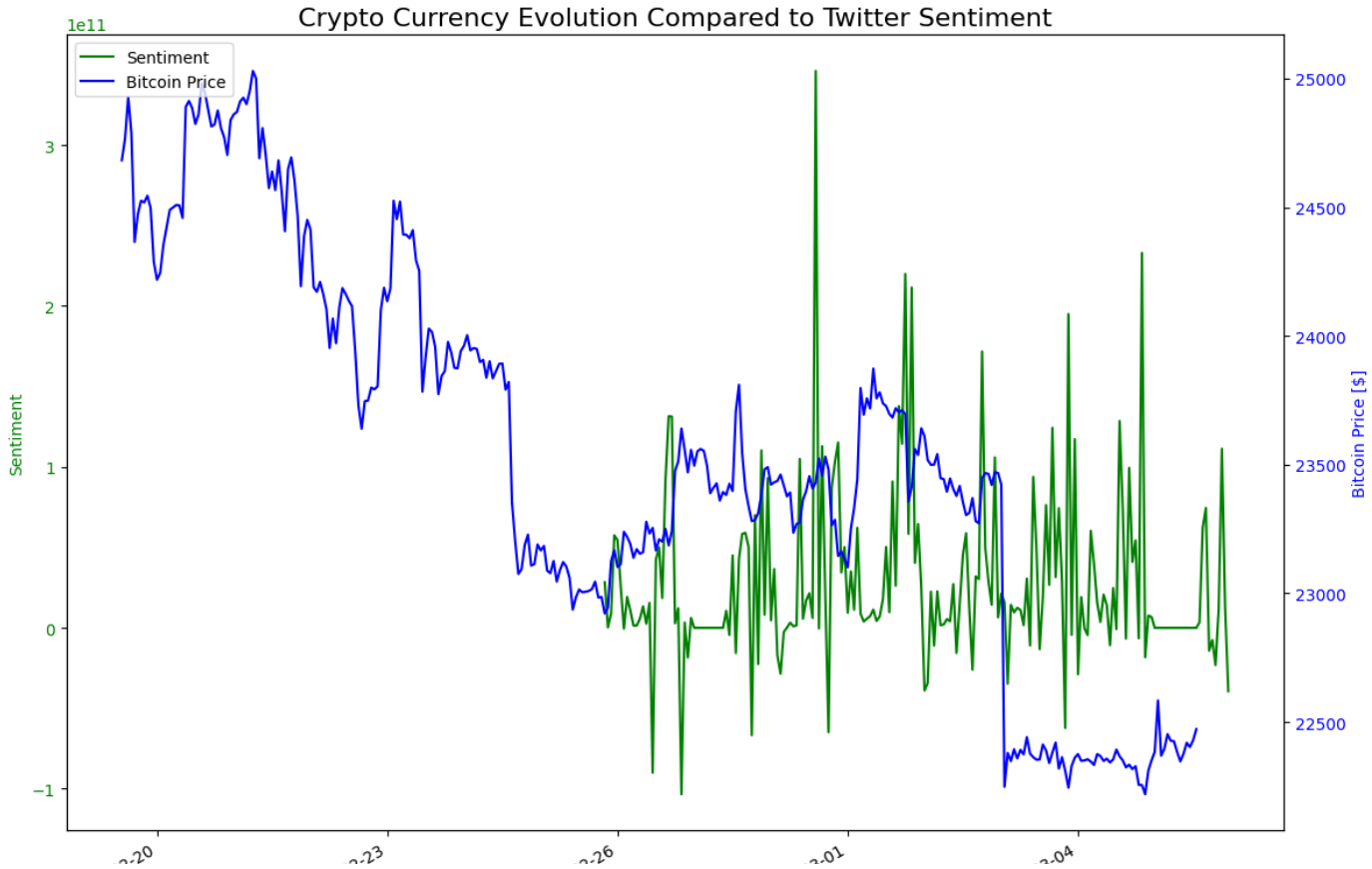
Sentiment Analysis Vs Price [Close]

```
1 #Grouping the data by hours allows you to analyze the sentiment trends at a more granular level and grouping by hours can help in aggregat
```

```
1 import pandas as pd
2
3 # Assuming you have a DataFrame named 'tweets'
4 tweets['date'] = pd.to_datetime(tweets['date'])
5 tweets.set_index('date', inplace=True)
6
7 # Group by hour and sum the 'score' column
8 tweets_grouped = tweets['score'].resample('1H').sum()
```

```
1 #'crypto_usd' DataFrame
2 crypto_usd['time'] = pd.to_datetime(crypto_usd['time'])
3 crypto_usd.set_index('time', inplace=True)
4
5 # Group by hour and calculate the mean of the 'close' column
6 crypto_usd_grouped = crypto_usd['close'].resample('1H').mean()
```

```
1 import matplotlib.pyplot as plt
2
3 fig, ax1 = plt.subplots(figsize=(12, 8))
4 ax2 = ax1.twinx()
5
6 # Plot sentiment analysis data
7 ax1.plot_date(tweets_grouped.index, tweets_grouped, 'g-', label='Sentiment')
8 ax1.set_ylabel("Sentiment", color='g')
9 ax1.tick_params(axis='y', labelcolor='g')
10
11 # Plot cryptocurrency price data
12 ax2.plot_date(crypto_usd_grouped.index, crypto_usd_grouped, 'b-', label='Bitcoin Price')
13 ax2.set_ylabel("Bitcoin Price [$]", color='b')
14 ax2.tick_params(axis='y', labelcolor='b')
15
16 # Set title and x-axis label
17 plt.title("Crypto Currency Evolution Compared to Twitter Sentiment", fontsize=16)
18 plt.xlabel("Date")
19
20 # Add legend
21 lines = ax1.get_lines() + ax2.get_lines()
22 labels = [line.get_label() for line in lines]
23 plt.legend(lines, labels, loc='upper left')
24
25 # Format the x-axis tick labels
26 ax1.xaxis.set_major_locator(plt.MaxNLocator(6))  # Set maximum number of x-axis tick labels
27 fig.autofmt_xdate()  # Automatically format x-axis tick labels as dates
28
29 plt.tight_layout()
30 plt.show()
31
```




```
#Correlation analysis
```

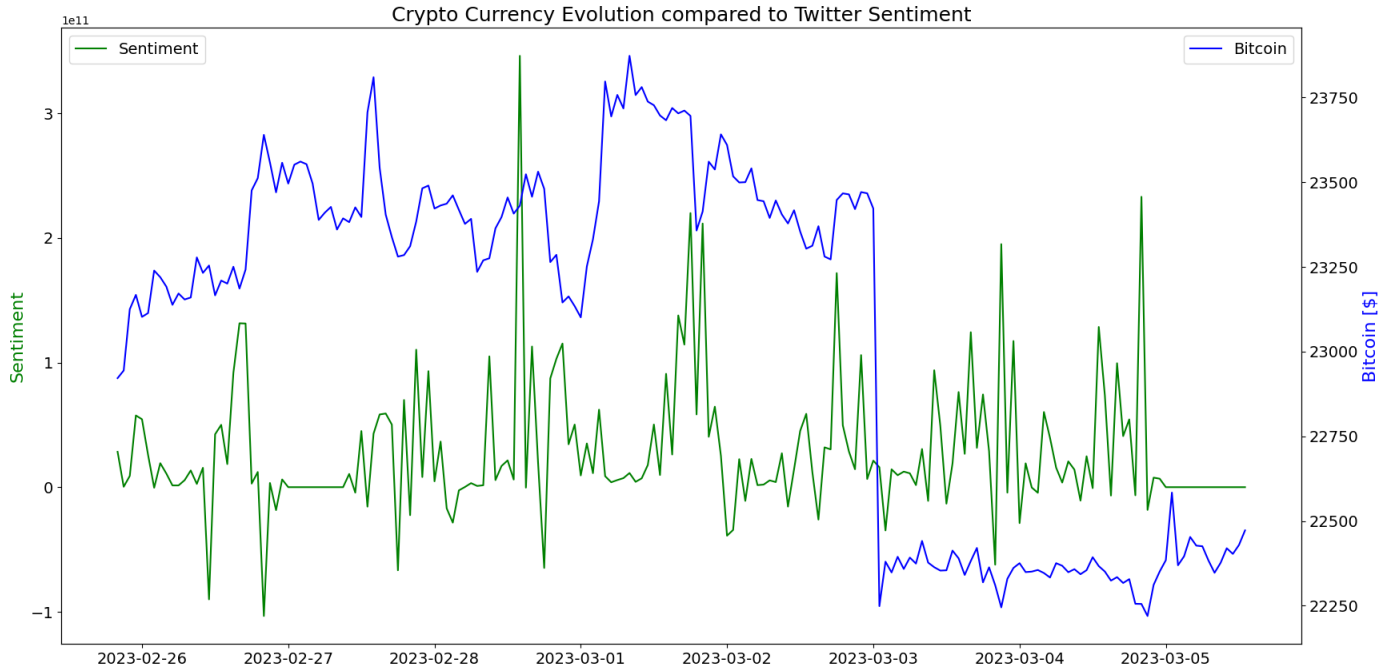
```
# Define the cross-correlation function

def crosscorr(datax, datay, lag=0, method="pearson"):
    """ Lag-N cross correlation.
    Parameters
    ----------
    lag : int, default 0
    datax, datay : pandas.Series objects of equal length

    Returns
    ----------
    crosscorr : float
    """
    return datax.corr(datay.shift(lag), method=method)
```

```
#Crop the time series to match the time frames
```

```
beggining = max(tweets_grouped.index.min(), crypto_usd_grouped.index.min())
end = min(tweets_grouped.index.max(), crypto_usd_grouped.index.max())
tweets_grouped = tweets_grouped[beggining:end]
crypto_usd_grouped = crypto_usd_grouped[beggining:end]
```

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Create figure and axes
fig, ax1 = plt.subplots(figsize=(20, 10))
ax2 = ax1.twinx()

# Set title and axis labels
ax1.set_title("Crypto Currency Evolution compared to Twitter Sentiment", fontsize=18)
ax1.set_ylabel("Sentiment", color='g', fontsize=16)
ax2.set_ylabel("Bitcoin [$]", color='b', fontsize=16)

# Set tick label size
ax1.tick_params(labelsize=14)
ax2.tick_params(labelsize=14)
```

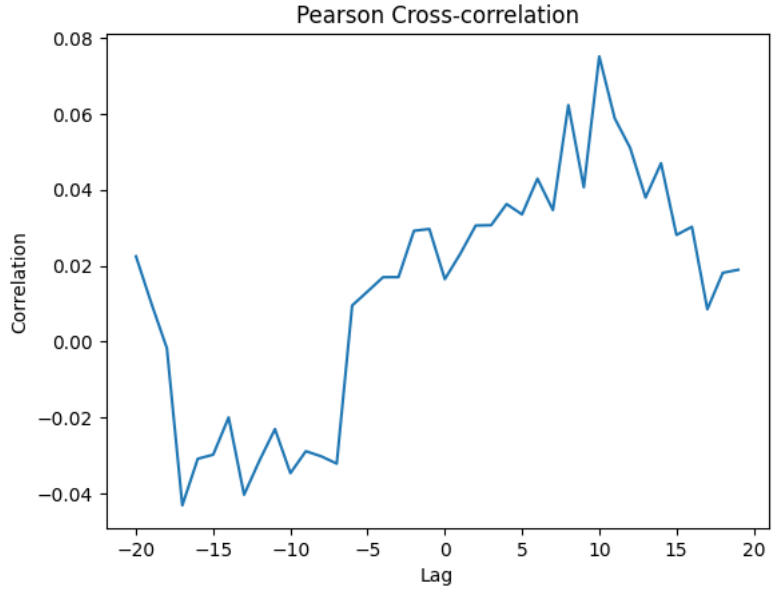
```
16
17 # Plot sentiment data
18 ax1.plot_date(tweets_grouped.index, tweets_grouped, 'g-', label="Sentiment")
19
20 # Plot cryptocurrency data
21 ax2.plot_date(crypto_usd_grouped.index, crypto_usd_grouped, 'b-', label="Bitcoin")
22
23 # Format x-axis as dates
24 ax1.xaxis.set_major_locator(mdates.AutoDateLocator())
25 ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
26
27 # Rotate x-axis tick labels for better visibility
28 plt.xticks(rotation=45)
29
30 # Display legend
31 ax1.legend(loc='upper left', fontsize=14)
32 ax2.legend(loc='upper right', fontsize=14)
33
34 # Show the plot
35 plt.show()
36
```

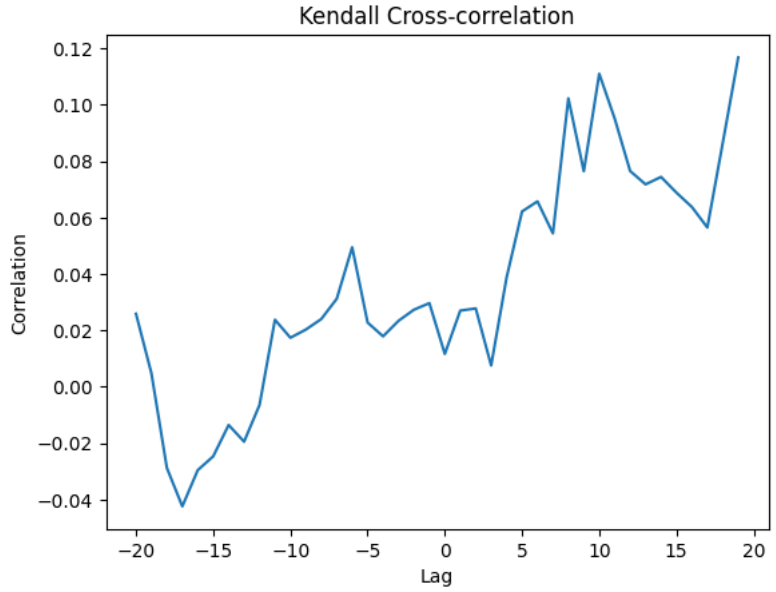



```
1 #The code imports the matplotlib.pyplot library for plotting and matplotlib.dates library for date formatting
```

```
1 #The plot visually suggests a potential relationship between sentiment extracted from Twitter data and the price evolution of Bitcoin, inc
```

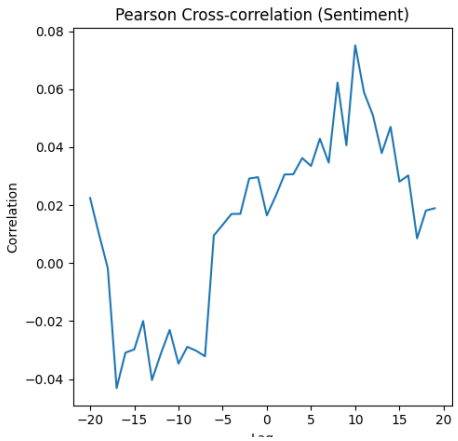
```
1 import matplotlib.pyplot as plt
2
3 methods = ["pearson", "kendall", "spearman"]
4
5 for method in methods:
6     xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=i, method=method) for i in range(-20, 20)]
7     plt.plot(range(-20, 20), xcov)
8     plt.title(f"{method.capitalize()} Cross-correlation")
9     plt.xlabel("Lag")
10    plt.ylabel("Correlation")
11    plt.show()
12
```

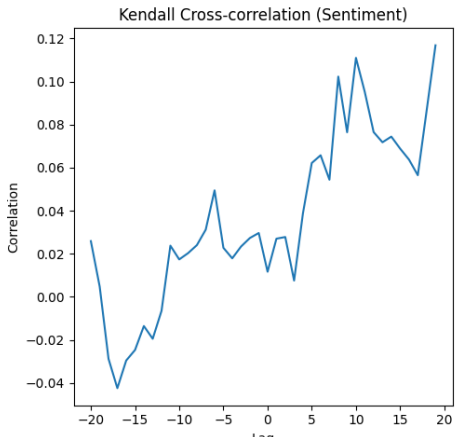


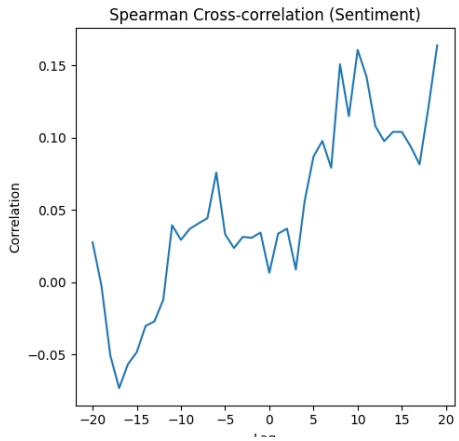



```
#To analyze the results and interpret the cross-correlation plots
```

```
import matplotlib.pyplot as plt

methods = ["pearson", "kendall", "spearman"]

fig, axs = plt.subplots(1, 3, figsize=(15, 5))

# Sentiment Analysis
for i, method in enumerate(methods):
    xcov_sentiment = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=j, method=method) for j in range(-20, 20)]
    axs[i].plot(range(-20, 20), xcov_sentiment)
    axs[i].set_title(f"{method.capitalize()} Cross-correlation (Sentiment)")
    axs[i].set_xlabel("Lag")
    axs[i].set_ylabel("Correlation")

plt.tight_layout()
plt.show()

```








```
from scipy.stats import pearsonr, kendalltau, spearmanr

methods = ["pearson", "kendall", "spearman"]

for method in methods:
    if method == "pearson":
        correlation_coefficient, p_value = pearsonr(tweets_grouped, crypto_usd_grouped)
    elif method == "kendall":
        correlation_coefficient, p_value = kendalltau(tweets_grouped, crypto_usd_grouped)
    elif method == "spearman":
        correlation_coefficient, p_value = spearmanr(tweets_grouped, crypto_usd_grouped)

    print(f"{method.capitalize()} correlation coefficient:", correlation_coefficient)
    print(f"{method.capitalize()} p-value:", p_value)

    if p_value < 0.05:
        print(f"The {method} correlation is statistically significant at this confidence level.")
    else:
        print(f"The {method} correlation is not statistically significant at this confidence level.")
    print()

```

```
Pearson correlation coefficient: 0.01643626585537878
Pearson p-value: 0.8237964874502892
The pearson correlation is not statistically significant at this confidence level.

Kendall correlation coefficient: 0.01166030969779228
Kendall p-value: 0.8144610941602477
The kendall correlation is not statistically significant at this confidence level.

Spearman correlation coefficient: 0.006553754143596371
Spearman p-value: 0.9292569945760621
The spearman correlation is not statistically significant at this confidence level.
```

```
#The Pearson, Kendall, and Spearman correlation coefficients between the sentiment and Bitcoin prices are all very close to zero, indicati
```

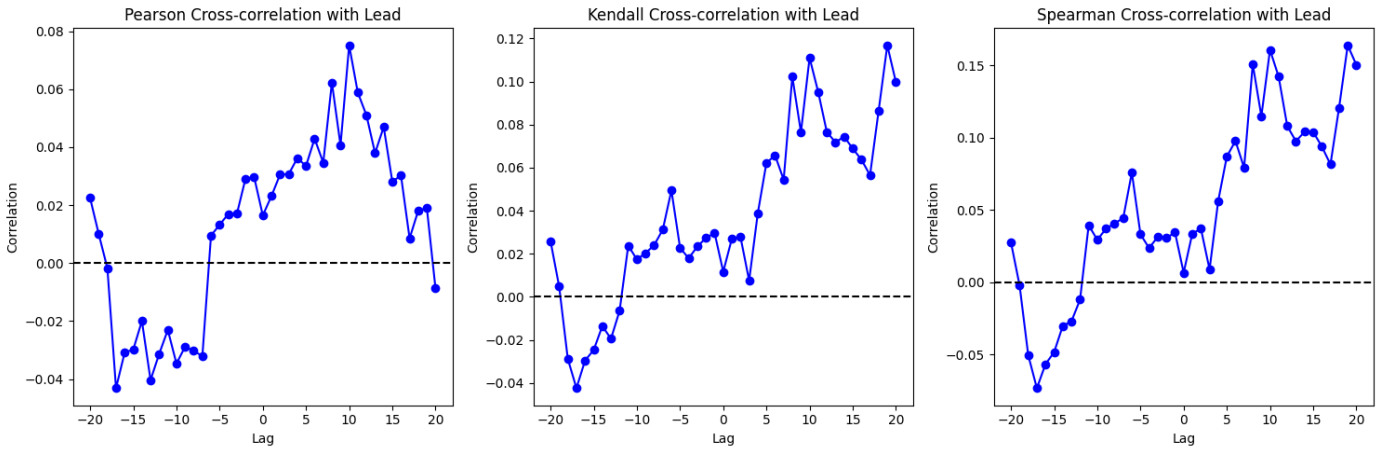
```
# Identify interesting correlations
interesting_correlations = []

# Loop through different lag values
for lag in range(-20, 20):
    # Calculate cross-correlation
    correlation = crosscorr(tweets_grouped, crypto_usd_grouped, lag=lag, method="pearson")

    # Check if correlation is interesting (adjust criteria as needed)
    if abs(correlation) > 0.5:
        interesting_correlations.append((lag, correlation))

# Print the interesting correlations
for lag, correlation in interesting_correlations:
    print(f"Correlation at lag {lag}: {correlation}")

# Conduct additional analysis for the interesting correlations
for lag, correlation in interesting_correlations:
    # Perform specific analysis for each interesting correlation
    # Example: Analyze the relationship during specific time periods or incorporate additional variables
```

```
    print(f"Additional analysis for correlation at lag {lag}")

```

```
#The cross-correlation with different lags for the lead (future) relationship between sentiment and cryptocurrency prices
```

```
import matplotlib.pyplot as plt

methods = ["pearson", "kendall", "spearman"]
lags = range(-20, 21)  # Include both positive and negative lags

for method in methods:
    for lag in lags:
        xcov = crosscorr(tweets_grouped, crypto_usd_grouped, lag=lag, method=method)
        plt.plot(lag, xcov, marker='o', color='b')

    plt.title(f"{method.capitalize()} Cross-correlation with Lead")
    plt.xlabel("Lag")
    plt.ylabel("Correlation")
    plt.axhline(0, color='black', linestyle='--')  # Add a horizontal line at y=0 for reference
    plt.show()

```

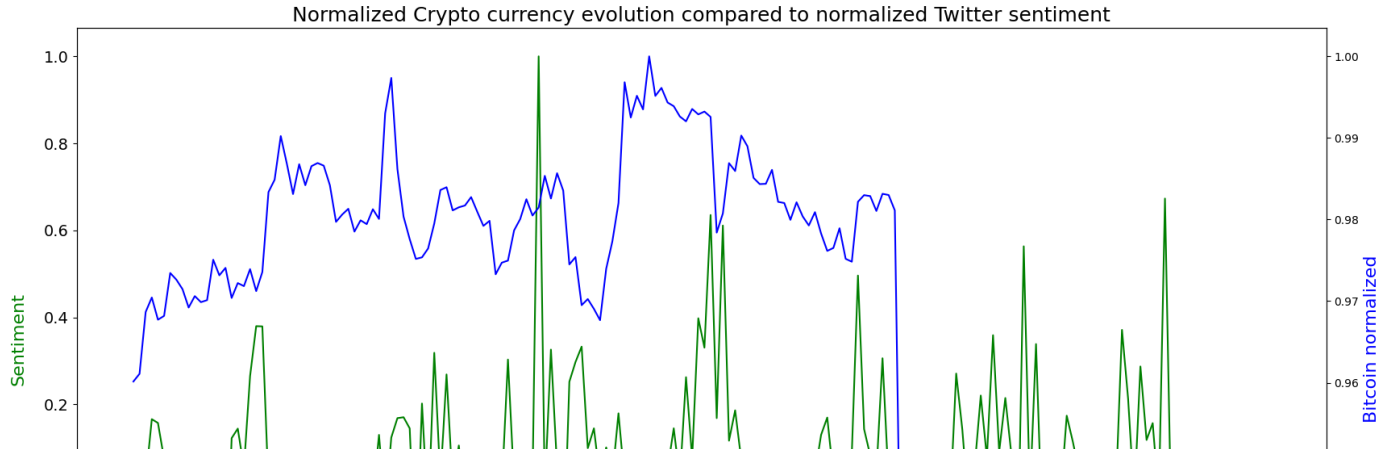
```
import matplotlib.pyplot as plt

methods = ["pearson", "kendall", "spearman"]
lags = range(-20, 21)  # Include both positive and negative lags

fig, axs = plt.subplots(1, 3, figsize=(15, 5))  # Create a 1x3 grid of subplots

for i, method in enumerate(methods):
    correlations = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=lag, method=method) for lag in lags]
    axs[i].plot(lags, correlations, marker='o', color='b')
    axs[i].set_title(f"{method.capitalize()} Cross-correlation with Lead")
    axs[i].set_xlabel("Lag")
    axs[i].set_ylabel("Correlation")
    axs[i].axhline(0, color='black', linestyle='--')  # Add a horizontal line at y=0 for reference

plt.tight_layout()  # Adjust spacing between subplots
plt.show()
```




```
#The term "lead" suggests that the correlation is examined for the sentiment leading the Bitcoin prices at various time shifts. The plot h
```

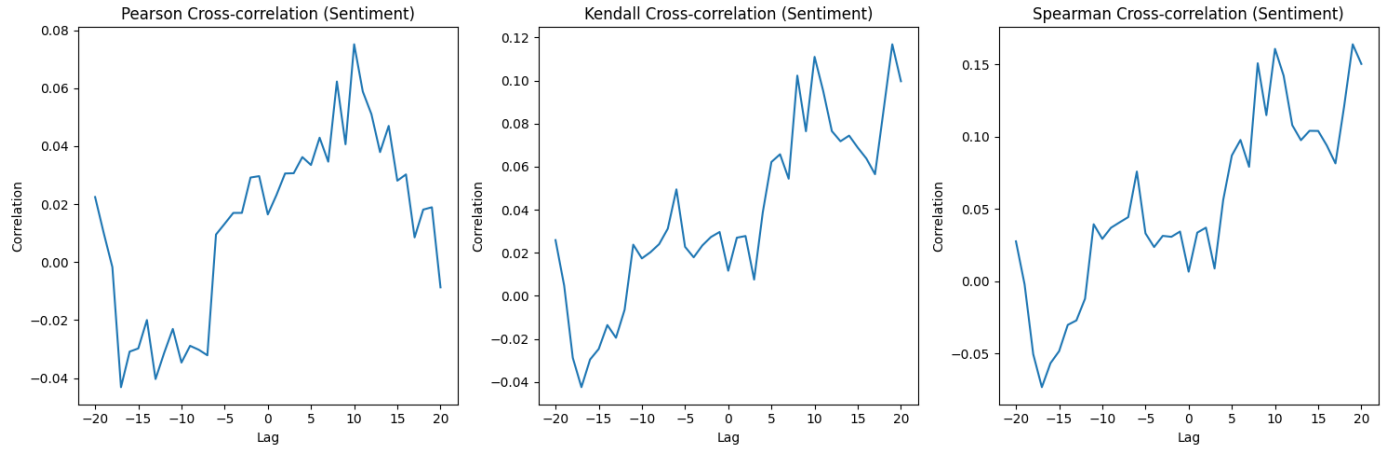
```
#Normalize
```

```
import matplotlib.pyplot as plt

# Normalize time series data
tweets_grouped_normalized = tweets_grouped / max(tweets_grouped.max(), abs(tweets_grouped.min()))
crypto_usd_grouped_normalized = crypto_usd_grouped / max(crypto_usd_grouped.max(), abs(crypto_usd_grouped.min()))

# Create the plot
fig, ax1 = plt.subplots(figsize=(20, 10))
ax1.set_title("Normalized Crypto currency evolution compared to normalized Twitter sentiment", fontsize=18)
ax1.tick_params(labelsize=14)

ax2 = ax1.twinx()
ax1.plot_date(tweets_grouped_normalized.index, tweets_grouped_normalized, 'g-')
ax2.plot_date(crypto_usd_grouped_normalized.index, crypto_usd_grouped_normalized, 'b-')

ax1.set_ylabel("Sentiment", color='g', fontsize=16)
ax2.set_ylabel("Bitcoin normalized", color='b', fontsize=16)

plt.show()
```



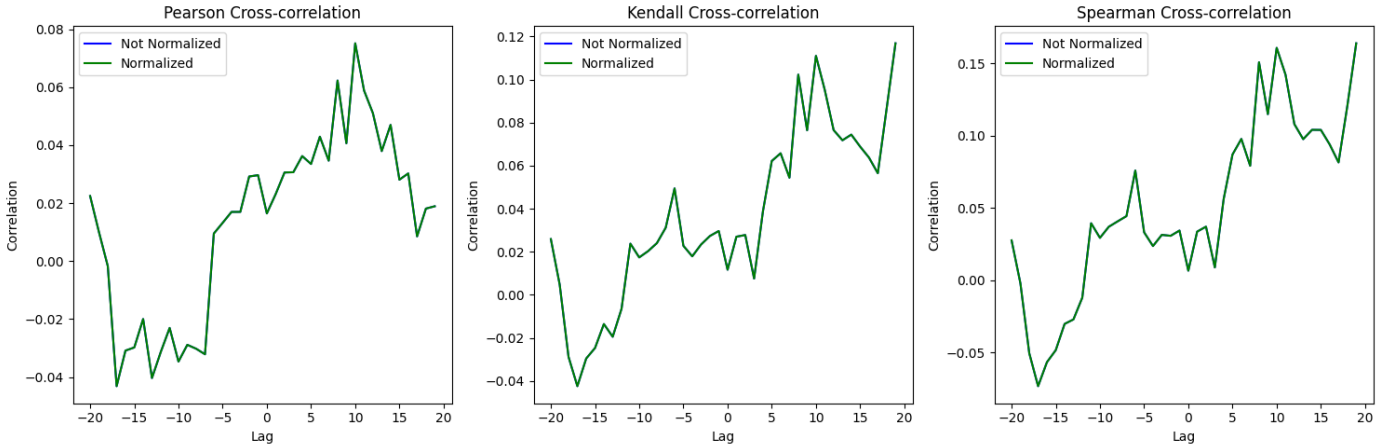

```
#The plot allows for visual analysis of the relationship between the two variables, after normalizing the data to a common scale.
```

```
import matplotlib.pyplot as plt

methods = ["pearson", "kendall", "spearman"]
lags = range(-20, 21)  # Include both positive and negative lags

fig, axs = plt.subplots(1, 3, figsize=(15, 5))

# Normalize time series data
tweets_grouped_norm = (tweets_grouped - tweets_grouped.mean()) / tweets_grouped.std()
crypto_usd_grouped_norm = (crypto_usd_grouped - crypto_usd_grouped.mean()) / crypto_usd_grouped.std()

# Sentiment Analysis
for i, method in enumerate(methods):
    xcov_sentiment = [crosscorr(tweets_grouped_norm, crypto_usd_grouped_norm, lag=j, method=method) for j in lags]
    axs[i].plot(lags, xcov_sentiment)
    axs[i].set_title(f"{method.capitalize()} Cross-correlation (Sentiment)")
    axs[i].set_xlabel("Lag")
    axs[i].set_ylabel("Correlation")

plt.tight_layout()
plt.show()

```




```
import matplotlib.pyplot as plt

methods = ["pearson", "kendall", "spearman"]

fig, axs = plt.subplots(1, 3, figsize=(15, 5))

for i, method in enumerate(methods):
    # Calculate cross-correlation without normalization
    xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=j, method=method) for j in range(-20, 20)]
    axs[i].plot(range(-20, 20), xcov, color='blue', label='Not Normalized')

    # Calculate cross-correlation with normalization
```

```
    xcov_norm = [crosscorr(tweets_grouped_norm, crypto_usd_grouped_norm, lag=j, method=method) for j in range(-20, 20)]
    axs[i].plot(range(-20, 20), xcov_norm, color='green', label='Normalized')

    axs[i].set_title(f"{method.capitalize()} Cross-correlation")
    axs[i].set_xlabel("Lag")
    axs[i].set_ylabel("Correlation")
    axs[i].legend()

plt.tight_layout()
plt.show()

```



Sentiment Analysis Vs Price Change Delta

```
print(crypto_usd.columns)
```

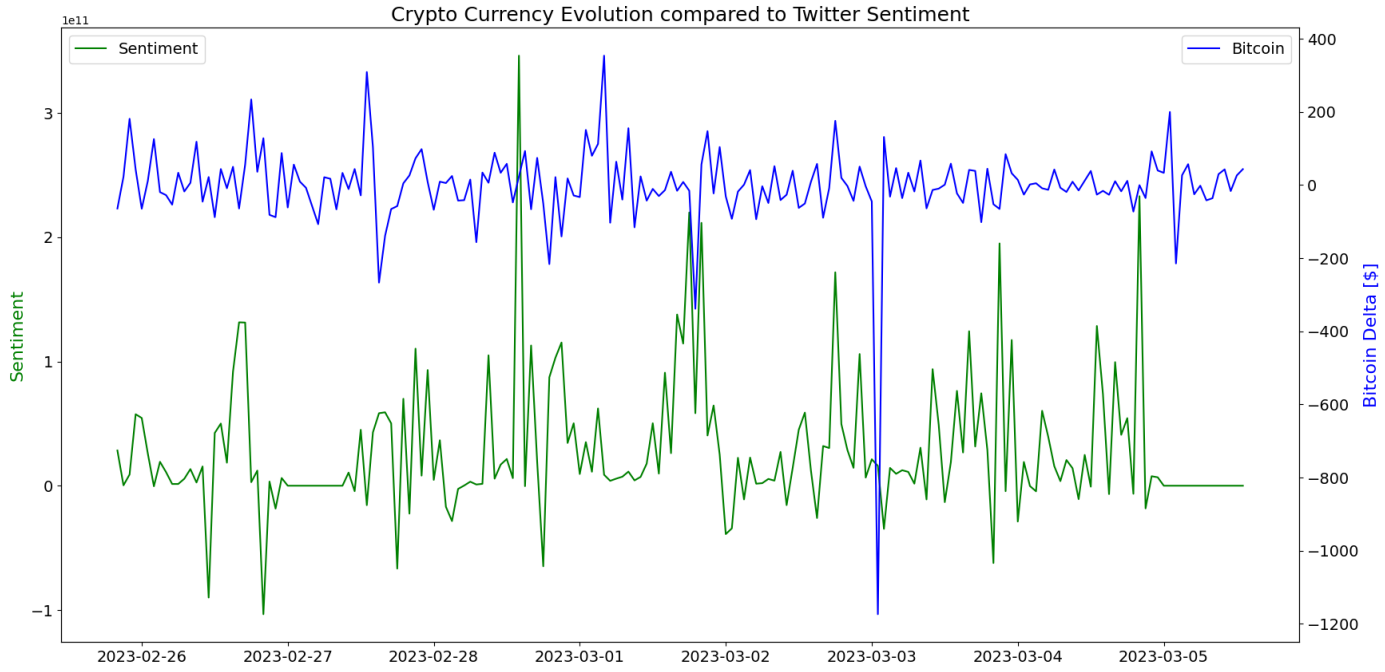
```
Index(['close', 'high', 'low', 'open', 'volumefrom', 'volumeto', 'Date',
       'Time', 'volume', 'marketcap', 'price_delta'],
      dtype='object')
```

```
# Group by hour and calculate the mean of the 'close' column
crypto_usd_grouped2 = crypto_usd['price_delta'].resample('1H').mean()
```

```
beggining = max(tweets_grouped.index.min(), crypto_usd_grouped2.index.min())
end = min(tweets_grouped.index.max(), crypto_usd_grouped2.index.max())
tweets_grouped = tweets_grouped[beggining:end]
crypto_usd_grouped2 = crypto_usd_grouped2[beggining:end]
```

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Create figure and axes
fig, ax1 = plt.subplots(figsize=(20, 10))
ax2 = ax1.twinx()

# Set title and axis labels
ax1.set_title("Crypto Currency Evolution compared to Twitter Sentiment", fontsize=18)
ax1.set_ylabel("Sentiment", color='g', fontsize=16)
ax2.set_ylabel("Bitcoin Delta [$]", color='b', fontsize=16)

# Set tick label size
ax1.tick_params(labelsize=14)
ax2.tick_params(labelsize=14)

# Plot sentiment data
ax1.plot_date(tweets_grouped.index, tweets_grouped, 'g-', label="Sentiment")

# Plot cryptocurrency data
ax2.plot_date(crypto_usd_grouped2.index, crypto_usd_grouped2, 'b-', label="Bitcoin")

# Format x-axis as dates
ax1.xaxis.set_major_locator(mdates.AutoDateLocator())
```

```
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))

# Rotate x-axis tick labels for better visibility
plt.xticks(rotation=45)

# Display legend
ax1.legend(loc='upper left', fontsize=14)
ax2.legend(loc='upper right', fontsize=14)

# Show the plot
plt.show()

```

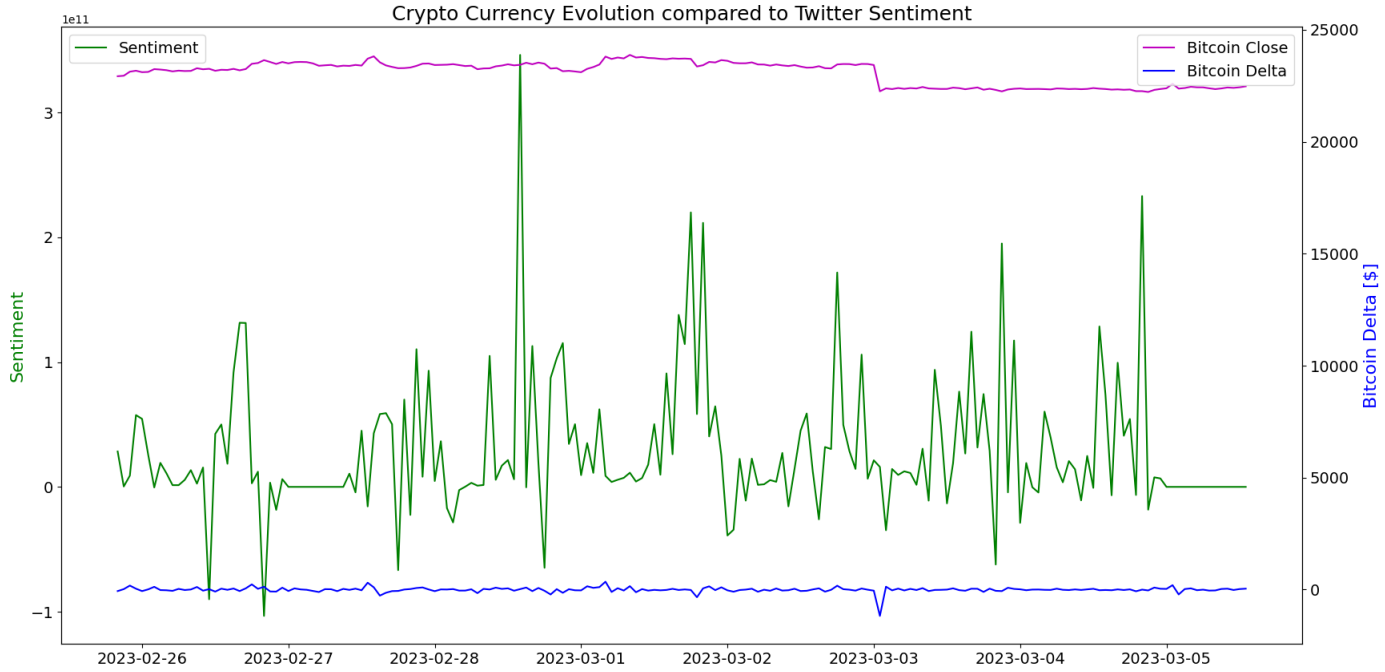



```
# Group by hour and sum the 'score' column
tweets_grouped = tweets['score'].resample('1H').sum()
# Group by hour and calculate the mean of the 'close' column
crypto_usd_grouped = crypto_usd['close'].resample('1H').mean()
# Group by hour and calculate the mean of the 'close' column
crypto_usd_grouped2 = crypto_usd['price_delta'].resample('1H').mean()
```

```
beggining = max(tweets_grouped.index.min(), crypto_usd_grouped.index.min())
end = min(tweets_grouped.index.max(), crypto_usd_grouped.index.max())
tweets_grouped = tweets_grouped[beggining:end]
crypto_usd_grouped = crypto_usd_grouped[beggining:end]
crypto_usd_grouped2 = crypto_usd_grouped2[beggining:end]
```

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Create figure and axes
fig, ax1 = plt.subplots(figsize=(20, 10))
ax2 = ax1.twinx()

# Set title and axis labels
ax1.set_title("Crypto Currency Evolution compared to Twitter Sentiment", fontsize=18)
ax1.set_ylabel("Sentiment", color='g', fontsize=16)
ax2.set_ylabel("Bitcoin Delta [$]", color='b', fontsize=16)

# Set tick label size
ax1.tick_params(labelsize=14)
ax2.tick_params(labelsize=14)

# Plot sentiment data
ax1.plot_date(tweets_grouped.index, tweets_grouped, 'g-', label="Sentiment")

# Plot cryptocurrency data (crypto_usd_grouped)
```

```
ax2.plot_date(crypto_usd_grouped.index, crypto_usd_grouped, 'm-', label="Bitcoin Close")

# Plot additional cryptocurrency data (crypto_usd_grouped2)
ax2.plot_date(crypto_usd_grouped2.index, crypto_usd_grouped2, 'b-', label="Bitcoin Delta")

# Format x-axis as dates
ax1.xaxis.set_major_locator(mdates.AutoDateLocator())
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))

# Rotate x-axis tick labels for better visibility
plt.xticks(rotation=45)

# Display legend
ax1.legend(loc='upper left', fontsize=14)
ax2.legend(loc='upper right', fontsize=14)

# Show the plot
plt.show()

```



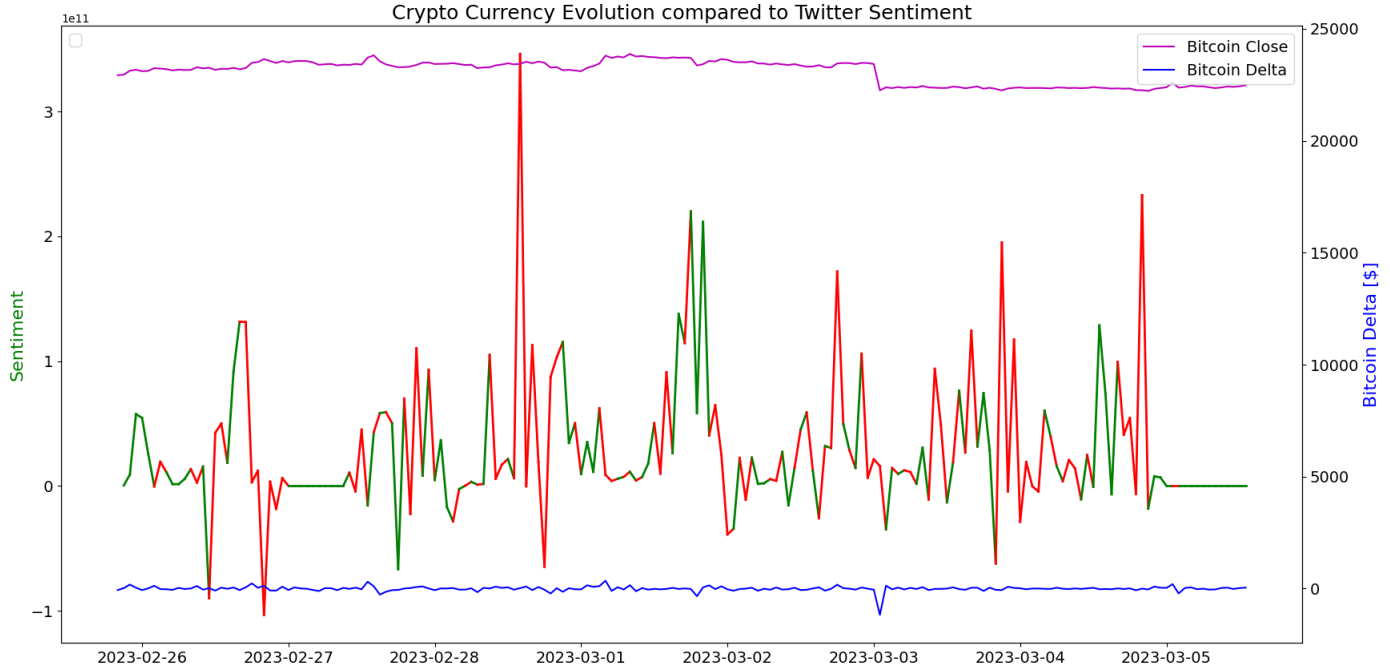

```
#the sentiment line is plotted based on the direction of the sentiment slope and the delta slope. If the sign of the sentiment slope at a

#This way, the areas where the sentiment line has the same direction as the delta line will be highlighted in red for better observation.
```

```
#consider the sentiment slope 1 hour before the delta slope to observe if the sentiment affects the price change
```

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Create figure and axes
fig, ax1 = plt.subplots(figsize=(20, 10))
ax2 = ax1.twinx()

# Set title and axis labels
ax1.set_title("Crypto Currency Evolution compared to Twitter Sentiment", fontsize=18)
ax1.set_ylabel("Sentiment", color='g', fontsize=16)
ax2.set_ylabel("Bitcoin Delta [$]", color='b', fontsize=16)

# Set tick label size
ax1.tick_params(labelsize=14)
ax2.tick_params(labelsize=14)

# Calculate sentiment slope and delta slope
sentiment_slope = np.gradient(tweets_grouped)
delta_slope = np.gradient(crypto_usd_grouped2)
```

```
21
22 # Plot sentiment data
23 for i in range(1, len(tweets_grouped)-1):
24     if np.sign(sentiment_slope[i-1]) == np.sign(delta_slope[i]):
25         ax1.plot([tweets_grouped.index[i], tweets_grouped.index[i+1]], [tweets_grouped[i], tweets_grouped[i+1]], 'r-', linewidth=2)
26     else:
27         ax1.plot([tweets_grouped.index[i], tweets_grouped.index[i+1]], [tweets_grouped[i], tweets_grouped[i+1]], 'g-', linewidth=2)
28
29 # Plot cryptocurrency data (crypto_usd_grouped)
30 ax2.plot_date(crypto_usd_grouped.index, crypto_usd_grouped, 'm-', label="Bitcoin Close")
31
32 # Plot additional cryptocurrency data (crypto_usd_grouped2)
33 ax2.plot_date(crypto_usd_grouped2.index, crypto_usd_grouped2, 'b-', label="Bitcoin Delta")
34
35 # Format x-axis as dates
36 ax1.xaxis.set_major_locator(mdates.AutoDateLocator())
37 ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
38
39 # Rotate x-axis tick labels for better visibility
40 plt.xticks(rotation=45)
41
42 # Display legend
43 ax1.legend(loc='upper left', fontsize=14)
44 ax2.legend(loc='upper right', fontsize=14)
45
46 # Show the plot
47 plt.show()
48
```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ig
```




```
1 #The plot visualizes the comparison between sentiment and Bitcoin delta, indicating the segments where their slopes have the same sign usi
```

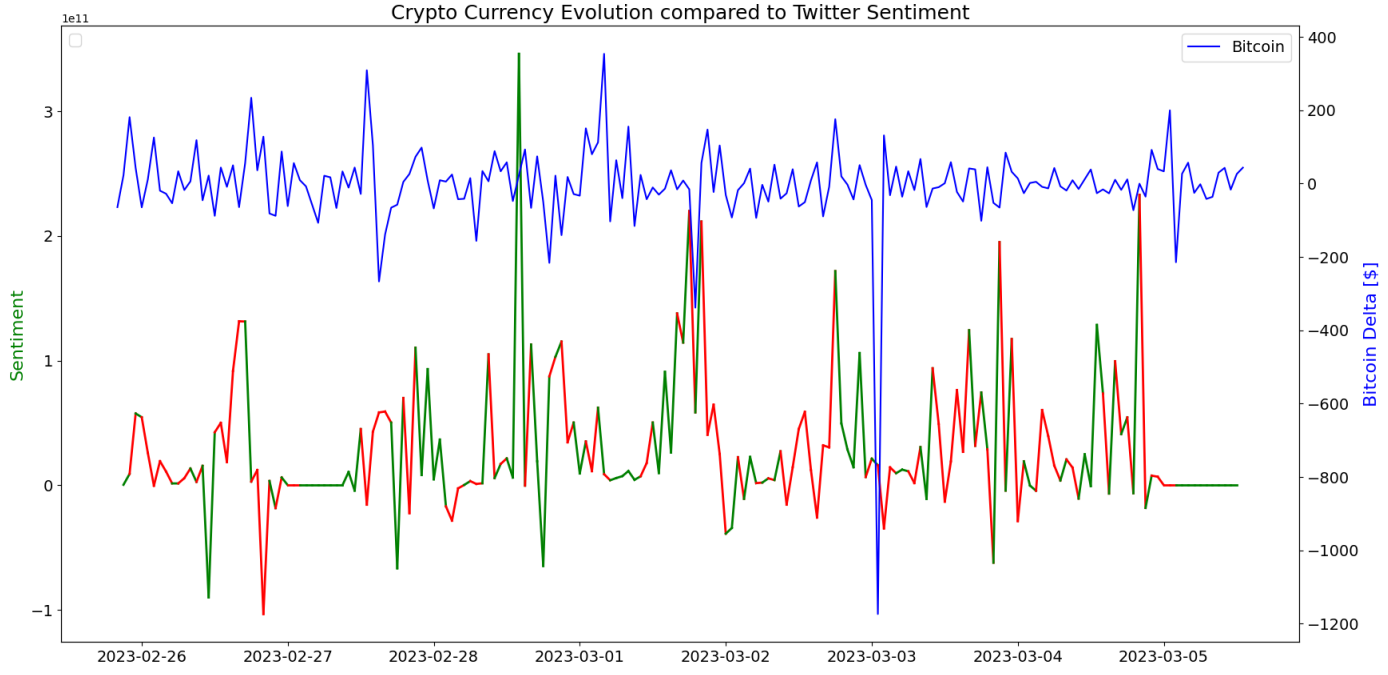
```
1 #Delta and Sentiment - consider the sentiment slope 1 hour before the delta slope to observe if the sentiment affects the price change
```

```
1 # If the sign of the sentiment slope at index i-2 is the same as the sign of the delta slope at index i, a red line ('r-') is plotted betw
```

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import matplotlib.dates as mdates
4
5 # Create figure and axes
6 fig, ax1 = plt.subplots(figsize=(20, 10))
7 ax2 = ax1.twinx()
8
```

```
# Set title and axis labels
ax1.set_title("Crypto Currency Evolution compared to Twitter Sentiment", fontsize=18)
ax1.set_ylabel("Sentiment", color='g', fontsize=16)
ax2.set_ylabel("Bitcoin Delta [$]", color='b', fontsize=16)

# Set tick label size
ax1.tick_params(labelsize=14)
ax2.tick_params(labelsize=14)

# Calculate sentiment slope and delta slope
sentiment_slope = np.gradient(tweets_grouped)
delta_slope = np.gradient(crypto_usd_grouped2)

# Plot sentiment data
for i in range(2, len(tweets_grouped)-1):
    if np.sign(sentiment_slope[i-2]) == np.sign(delta_slope[i]):
        ax1.plot_date([tweets_grouped.index[i-1], tweets_grouped.index[i]], [tweets_grouped[i-1], tweets_grouped[i]], 'r-', linewidth=2)
    else:
        ax1.plot_date([tweets_grouped.index[i-1], tweets_grouped.index[i]], [tweets_grouped[i-1], tweets_grouped[i]], 'g-', linewidth=2)

# Plot cryptocurrency data
ax2.plot_date(crypto_usd_grouped2.index, crypto_usd_grouped2, 'b-', label="Bitcoin")

# Format x-axis as dates
ax1.xaxis.set_major_locator(mdates.AutoDateLocator())
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))

# Rotate x-axis tick labels for better visibility
plt.xticks(rotation=45)

# Display legend
ax1.legend(loc='upper left', fontsize=14)
ax2.legend(loc='upper right', fontsize=14)

# Show the plot
plt.show()

```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ig
```




```
#To consider both the slope direction and magnitude proportionality
```

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

```

```
# Create figure and axes
fig, ax1 = plt.subplots(figsize=(20, 10))
ax2 = ax1.twinx()

# Set title and axis labels
ax1.set_title("Crypto Currency Evolution compared to Twitter Sentiment", fontsize=18)
ax1.set_ylabel("Sentiment", color='g', fontsize=16)
ax2.set_ylabel("Bitcoin Delta [$]", color='b', fontsize=16)

# Set tick label size
ax1.tick_params(labelsize=14)
ax2.tick_params(labelsize=14)

# Calculate sentiment slope and delta slope
sentiment_slope = np.gradient(tweets_grouped)
delta_slope = np.gradient(crypto_usd_grouped2)

# Calculate proportionality factor
proportionality = 0.2  # Adjust this value as desired

# Plot sentiment data
for i in range(2, len(tweets_grouped)-1):
    if np.sign(sentiment_slope[i-2]) == np.sign(delta_slope[i]):
        if np.abs(sentiment_slope[i-2]) >= proportionality * np.abs(delta_slope[i]):
            ax1.plot_date([tweets_grouped.index[i-1], tweets_grouped.index[i]], [tweets_grouped[i-1], tweets_grouped[i]], 'r-', linewidth=
        else:
            ax1.plot_date([tweets_grouped.index[i-1], tweets_grouped.index[i]], [tweets_grouped[i-1], tweets_grouped[i]], 'g-', linewidth=
    else:
        ax1.plot_date([tweets_grouped.index[i-1], tweets_grouped.index[i]], [tweets_grouped[i-1], tweets_grouped[i]], 'g-', linewidth=2)

# Plot cryptocurrency data
ax2.plot_date(crypto_usd_grouped2.index, crypto_usd_grouped2, 'b-', label="Bitcoin")

# Format x-axis as dates
ax1.xaxis.set_major_locator(mdates.AutoDateLocator())
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))

# Rotate x-axis tick labels for better visibility
plt.xticks(rotation=45)

# Display legend
ax1.legend(loc='upper left', fontsize=14)
ax2.legend(loc='upper right', fontsize=14)

# Show the plot
plt.show()

```