## Research Question

Bitcoin is a cryptocurrency invented in 2008 by an unknown person or group of people using the name Satoshi Nakamoto. (Davis,2011), Trading the currency began in 2009 when its implementation was released as open-source software and as a decentralized digital currency (Antonopoulos, 2014), Without a central bank or single administrator, Bitcoin can be sent from user to user on the peer-to-peer network without the need for intermediaries, they can be exchanged for other currencies, products, and services. (Davis,2011), Bitcoins are mainly created as a reward for a process known as mining, Transactions are verified by network nodes through cryptography and recorded in a public distributed ledger called a blockchain. (Antonopoulos, 2014).

On 19 January 2021 Elon Musk placed #Bitcoin in his Twitter profile tweeting “In retrospect, it was inevitable”, which caused the price to briefly rise about $5000 in an hour to $37,299. (Browne,2021), This single tweet was a historic event and clearly showed the great influence of social media on cryptocurrencies prices.

Thinking of utilizing social media in forecasting stock shares and cryptocurrency prices has been started years earlier, with the boom in using social media as a platform to express opinions, Twitter specifically has become known as a location where news is quickly disseminated in a concise format, According to (Colianni,2015) in 2010 Bollen et al. utilized the Profile of Mood States (POMS) to predict the movement of the Dow Jones Industrial Average with 87.6% accuracy, and in 2009, Go et. al. focused on classifying tweets and used several approaches to achieve an accuracy of 84.2% with Multinomial Naïve Bayes, 79.2% with maximum entropy, and 82.9% using a support vector machine. (Colianni,2015), but on another hand, social media is still doubtful when it comes to research and statistical analysis because (for several reasons)

Twitter samples are not necessarily representative of the population, and this should be taken into consideration from a statistical analysis point of view. (White, 2018)

The research question to be asked is “Is there a relationship between the estimated sentiments from social media and the change in Bitcoin price?” and the purpose of this research is to answer this question by measuring the significance of the relationship between the estimated sentiments of Tweets (from Twitter) and the change in Bitcoin price through applying sentiment analysis to tweets about Bitcoin using the “Textblob” python library, then measuring the significance of the variation in Bitcoin price changes explained by the estimated sentiments through a specific period, technically by testing the probability of the dependency of change in Bitcoin prices on the sentiments estimated from tweets, through testing the following null and alternate hypotheses:

Null hypothesis: There is no correlation between the estimated sentiments from social media and the change in Bitcoin price.

Alternate Hypothesis: There is a significant relationship between the estimated sentiments from social media and the change in Bitcoin price.

In this study linear regression has been used to infer the significance of the relationships between the independent variable (Bitcoin price changes or ‘diff’) and dependent variable/s, by estimating the strength of the relationship between two variables related to the proportion of variance in the dependent variable that can be explained by the independent variable/variables. (Freedman, 2009)

The null hypothesis that has been tested by the t-test is that each coefficient vanishes (Freedman, 2009), using the corresponding p-value which represents the probability of obtaining

test results while the null hypothesis is correct, where a very small p-value means that such an extreme observed outcome would be very unlikely under the null hypothesis.

To extract and prepare the data for analysis, Python was used in a Jupyter Notebook. Python was selected because of its flexible nature and because it supports many useful packages for data processing, manipulation, and visualization, The Jupyter Notebook was chosen as the development environment due to its flexibility. The full code used in the analysis including imports and functions is attached as an appendix to this report.

## Data Collection

To achieve the goal of the study, the analysis needed prepared and cleaned text tweets with time stamps and time series of Bitcoin prices through the same period of tweets.
Two datasets have been used; both are public datasets that have been obtained through Kaggle.com

1. Bitcoin Twitter Data (https://www.kaggle.com/kaushiksuresh147/bitcoin-tweets)
2. Bitcoin Price Data (https://www.kaggle.com/maxwells/btcusd)

The two public datasets have been downloaded through the attached web links.

The first raw data set was extracted by the data provider using “Tweepy” for collecting the tweets (no retweets) that have #Bitcoin and #btc hashtags. The collection started on February 6th, 2021 (Kaushik, 2021), the data retrieved on July 18th, 2021 in CSV format contains 13 columns and 423,111 records cover the period from February 5th, 2021 to July 5th,2021 with an irregular timestamp, A collection process, and variables dictionary are explained in the source web page (Kaushik, 2021), one obvious disadvantage of the process is that the collected data is not evenly distributed through the whole period, too dense within some small time intervals and very sparse

through the periods in between, this problem has been fixed and will be explained in the data preparation section.

For this analysis only three variables have been used:

(Date): UTC time and date when the Tweet was created, (Text): The actual UTF-8 text of the Tweet, and (User_followers): The number of followers an account currently has.

```
# Bitcoin twitter Data
#https://www.kaggle.com/kaushiksuresh147/bitcoin-tweets
#Bitcoin Tweets
df_Bitcoin_tweets = pd.read_csv('Bitcoin_tweets.csv')
df_Bitcoin_tweets.drop_duplicates(subset=['user_followers','date','text'],inplace=True)
df_Bitcoin_tweets.sort_values(by='date',inplace=True)
print('df_Bitcoin_tweets dimensions= ',df_Bitcoin_tweets.shape)
df_Bitcoin_tweets=df_Bitcoin_tweets[['user_followers','date','text']]
df_Bitcoin_tweets.head()
```

```
df_Bitcoin_tweets dimensions=  (346512, 13)
```

| | user_followers | date | text |
|---|---|---|---|
| **21523** | 301.0 | 2021-02-05 10:52:04 | 2□ Debunking 9 #Bitcoin Myths by @Patrick_Lowry_ ℹ️ \n\n#cryptocurrency #bitcoin #crypto #blockchain #btc... https://t.co/2CM83fuB2n |
| **21524** | 301.0 | 2021-02-05 10:52:04 | □ Weekend Read □\n\nKeen to learn about #crypto assets? Check out our reading list! \n1□ 2021 CryptoMarket Outlook... https://t.co/opiev94qzl |
| **21522** | 301.0 | 2021-02-05 10:52:06 | 4□ 📎 Bloomberg LP #CryptoOutlook 2021 with @mikemcglone11 ℹ️ \n\n#cryptocurrency #bitcoin #crypto #blockchain #btc... https://t.co/gC3VNGAP6v |
| **21521** | 301.0 | 2021-02-05 10:52:07 | 5□ #Blockchain 50 2021 by @DelRayMan, @Forbes , @ForbesCrypto ℹ️ \n\n#cryptocurrency #bitcoin #crypto #blockchain... https://t.co/L3Xj7j49Fx |
| **21520** | 37.0 | 2021-02-05 10:52:26 | #reddcoin #rdd @reddcoin to the moon #altcoin #turnreddcoinin1dollar #eth #btc #bitcoin https://t.co/ycm15VolsW |

Figure (1) – Bitcoin Twitter Data.

The second raw data set contains several CSV files represent the Bitcoin historical prices versus US dollar as six variables: time, open, high, low, close, and volume for different time frames or sample rates (Maxwells, 2021), For this analysis, the one minute CSV file has been used, the data was retrieved on July 18th, 2021 contains six columns and 1,629,299 records cover the period from March 24th, 2011, to June 15th, 2021, with a one-minute sample rate or Timeframe.

For most investors, the stock close price is much more important than high or low price levels (Simple stock trading, n.d.), The closing price is often the reference point used by

investors to compare a stock's performance since the previous day and closing prices are frequently used to construct line graphs depicting historical price changes over time (Hayes, 2021), using the closing price for stock chart analysis or real trade execution and management is a much better option because provides very important information about the general mood of investors (Simple stock trading, n.d.), for this analysis, the Bitcoin price will be represented by the “close” variable.

```
# Bitcoin Price Data
#BTCUSD
#https://www.kaggle.com/maxwells/btcusd
df_Bitcoin_Price= pd.read_csv('BTCUSD_1.csv')
df_Bitcoin_Price['time'] = pd.to_datetime(df_Bitcoin_Price['time'])
# Set time column as index
df_Bitcoin_Price.set_index('time',drop=True,inplace=True)
print('df_Bitcoin_Price dimensions= ',df_Bitcoin_Price.shape)
df_Bitcoin_Price=df_Bitcoin_Price[['close']]
df_Bitcoin_Price.tail()
```

```
df_Bitcoin_Price dimensions=  (1629299, 5)
```

| time | close |
|---|---|
| 2021-06-15 15:56:00 | 40251.14 |
| 2021-06-15 15:57:00 | 40308.39 |
| 2021-06-15 15:58:00 | 40345.39 |
| 2021-06-15 15:59:00 | 40406.24 |
| 2021-06-15 16:00:00 | 40378.25 |

Figure (2) – Bitcoin Price Data.

```
# Plot and show the time series on axis ax
fig, ax = plt.subplots(figsize = (14,8))
plt.plot(df_Bitcoin_Price.index,
         (df_Bitcoin_Price[['close']]),
         linewidth=1,)
#plt.ylim([0,3500])
plt.legend(['Price-diff'],loc='upper left')

plt.grid()
plt.show()
```

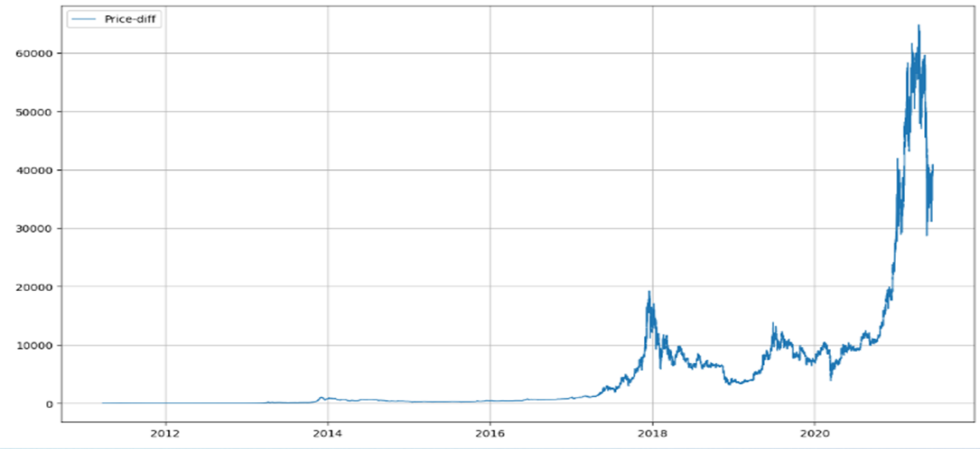


Figure (3) – Bitcoin Close Price plot.

## Data Extraction and Preparation

For the Tweets data set, the text for the tweets was prepared by dropping duplicates and cleaned from mentions (ex. @SomeOne), hashtags (ex. #SomeThing), URL links, and converted to lower case, a user-defined function has been created with the use of regular expression (Naik, 2020) and (Gulsen, 2021). This function has been applied to the "text" column of each row in the data frame, also the date column needed minor cleaning from some mis-formatted rows, another user-defined function was created for that, The cleaning process left some Null cells that had to be removed and finally the tweets data frame (df_Bitcoin_tweets) was converted into a time series, and the data was fully prepared for the next step, which is the sentiment analysis.

```
### Dataset Cleaning - tweets cleaning
# Edited after (Naik,2020)

def cleantxt(text):
    text=str(text)
    text = text.lower() #Converting Capital chars into small
    text = re.sub(r'@[a-zA-Z0-9]+', ' ', text) #Removing @ mentions
    text = re.sub("#bitcoin", 'bitcoin', text) # removeing the '#' from bitcoin
    text = re.sub("#btc", 'btc', text) # removeing the '#' from btc
    #text = re.sub('#[A-Za-z0-9]+', '', text) # removing any hastag or string starting with a '#'
    text = re.sub('\\n', '', text) # removing the '\n' string
    text = re.sub(r'https?:\/\/\S+', ' ', text) #Removing hyper links
    #text = re.sub(r'[^a-zA-Z]+', ' ', text) #Removing non ASCII chars
    text = text.split() #Removing extra spaces
    text = ' '.join(text)
    return text
```

```
### Dataset Cleaning - date cleaning

def cleandate(date):
    date=str(date)
    date = re.sub(r'[^0-9\-\:]', ' ', date) #Removing non ASCII chars
    date = date.split() #Removing extra spaces
    date = ' '.join(date)
    return date
```

```
df_Bitcoin_tweets['text']=df_Bitcoin_tweets['text'].apply(cleantxt)
df_Bitcoin_tweets['date']=df_Bitcoin_tweets['date'].apply(cleandate)
df_Bitcoin_tweets.dropna(inplace=True)
df_Bitcoin_tweets.drop(df_Bitcoin_tweets[df_Bitcoin_tweets['text'].str.len()==0].index,inplace=True)

df_Bitcoin_tweets['date'] = pd.to_datetime(df_Bitcoin_tweets['date'])

# Set date column as index
df_Bitcoin_tweets.set_index('date',drop=False, inplace=True)
```

Figure (4) – Bitcoin Twitter Data cleaning.

Sentiment analysis of tweets has been achieved using TextBlob which is a Python library for processing textual data. It provides a consistent API for diving into common natural language

processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more. (TextBlob, n.d.), TextBlob processed objects are similar to Python strings but with Natural Language Processing properties, The sentiment property returns a named tuple of the form Sentiment (polarity, subjectivity). The polarity score is afloat within the range [-1.0, 1.0]. where -1 is the most negative impression and +1 is the most positive, the subjectivity is afloat within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective, Using Textblob was straightforward but came with some disadvantages such as slowness in the comparison to other NLP libraries like spacy, and does not provide features like dependency parsing, word vectors, etc. (Jain, 2018) Also a noticeable disadvantage is that it tends to label most analyzed texts as neutral (or zero sentiment polarity) which add considerable uncertainty to the analysis, many Textblob users suggest converting the fraction polarity score into three categories (positive, neutral, and negative) or (+1, 0, and -1) as a best practice for the sentiment polarity score as it gives more reliable results and decreases the uncertainty in the sentiment analysis using Textblob (James, 2018) and (Gulsen, 2021).

```
def getsubjectivity(text):
    return TextBlob(text).sentiment.subjectivity
def getpolarity(text):
    return TextBlob(text).sentiment.polarity
```

```
df_Bitcoin_tweets['subj']=df_Bitcoin_tweets['text'].apply(getsubjectivity)
df_Bitcoin_tweets['polarity']=df_Bitcoin_tweets['text'].apply(getpolarity)
df_Bitcoin_tweets['count']=1
```

```
df_Bitcoin_tweets['polarity'] = df_Bitcoin_tweets['polarity'].apply(lambda x: 1 if x > 0 else (-1 if x < 0 else x))
df_Bitcoin_tweets.head(10)
```

| date | user_followers | date | text | subj | polarity | count |
|---|---|---|---|---|---|---|
| 2021-02-05 10:52:04 | 301.0 | 2021-02-05 10:52:04 | 2□ debunking 9 bitcoin myths by _lowry_ 1 #cryptocurrency bitcoin #crypto #blockchain btc... | 0.000000 | 0.0 | 1 |
| 2021-02-05 10:52:04 | 301.0 | 2021-02-05 10:52:04 | weekend read keen to learn about #crypto assets? check out our reading list! 1□ 2021 cryptomarket outlook... | 0.000000 | 0.0 | 1 |
| 2021-02-05 10:52:06 | 301.0 | 2021-02-05 10:52:06 | 4□ bloomberg lp #cryptooutlook 2021 with 1 #cryptocurrency bitcoin #crypto #blockchain btc... | 0.000000 | 0.0 | 1 |
| 2021-02-05 10:52:07 | 301.0 | 2021-02-05 10:52:07 | 5□ #blockchain 50 2021 by . . 1 #cryptocurrency bitcoin #crypto #blockchain... | 0.000000 | 0.0 | 1 |
| 2021-02-05 10:52:26 | 37.0 | 2021-02-05 10:52:26 | #reddcoin #rdd to the moon #altcoin #turnreddcoinin1dollar #eth btc bitcoin | 0.000000 | 0.0 | 1 |
| 2021-02-05 10:53:49 | 100.0 | 2021-02-05 10:53:49 | bitcoin and #eth both have bullish setups for a move higher ... btc it would just be great if daily close (in abo... | 0.416667 | 1.0 | 1 |
| 2021-02-05 10:54:52 | 69.0 | 2021-02-05 10:54:52 | Sperl 0.06.i have insisted that since 0.02 it will be 0.071. it increased 300% in about 2 months. bitcoin btc... | 0.000000 | 0.0 | 1 |
| 2021-02-05 | | 2021-02-05 | #amazing #monopoly #crypto #cryptocurrency #cryptocurrencies #cryptonews | | | |

Figure (5) – Bitcoin Twitter sentiment analysis.

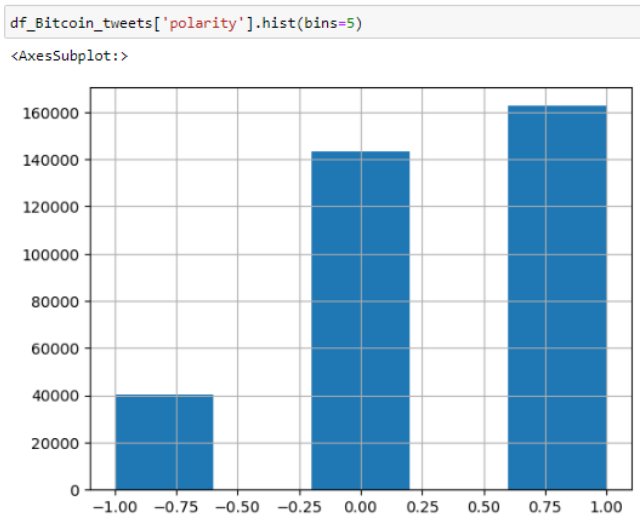


Figure (6) – Bitcoin Twitter sentiment polarity histogram.

Another column or variable has been added to the data and named “count”, which represent the count of tweets and was given a constant value ‘1’, this variable is supposed to be of great importance as it will represent the total number of tweets where Bitcoin was mentioned within a specific period after resampling and summation, provides a direct measurement of the subject popularity, or the strength of what is also known as the “Social media trend” (Lifestyle dictionary, n.d.).

The next step was resampling the tweets’ time series to make it match the Bitcoin prices data frame's sample rate (one minute) and aggregating the values of variables for each minute in one row.

```
df_Bitcoin_tweets=df_Bitcoin_tweets.resample('Min').sum()
```

```
# Plot and show the time series on axis ax
fig, ax = plt.subplots(figsize = (14,8))
df_Bitcoin_tweets[['count','subj','polarity']]. plot(ax=ax)
plt.grid()
plt.show()
```

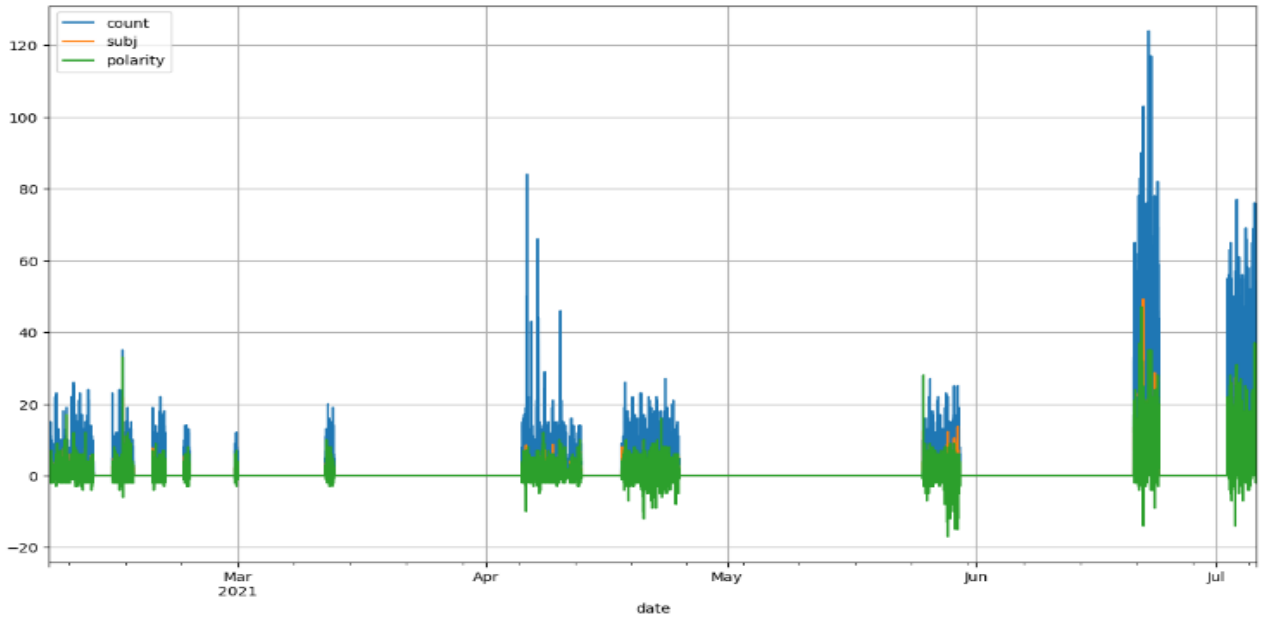


Figure (7) – Bitcoin Twitter sentiment data time series.

Both data frames, df_Bitcoin_tweets and df_Bitcoin_Price have been trimmed to match in length by giving both the same start and end DateTime to join or concatenate them together in one clean data frame starts at '2021-02-05 10:52:00' and ends at '2021-05-29 16:59:00' with one-minute sample-rate.

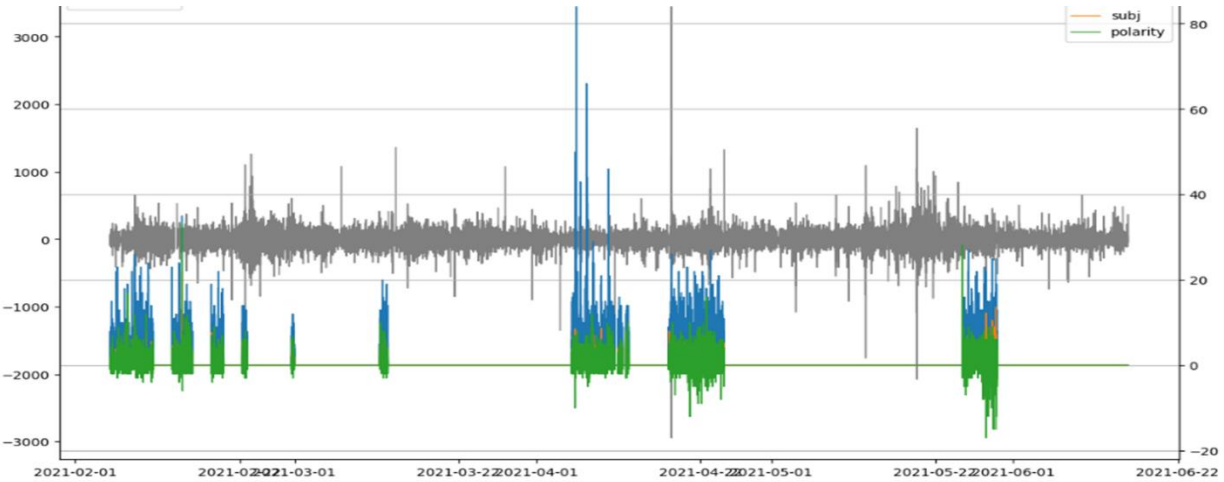


Figure (8) – Joined Bitcoin Twitter and price data.

Many samples didn't contain any tweets (has zero tweets count), those samples have been removed to prevent correlating actual Bitcoin prices with null tweets.

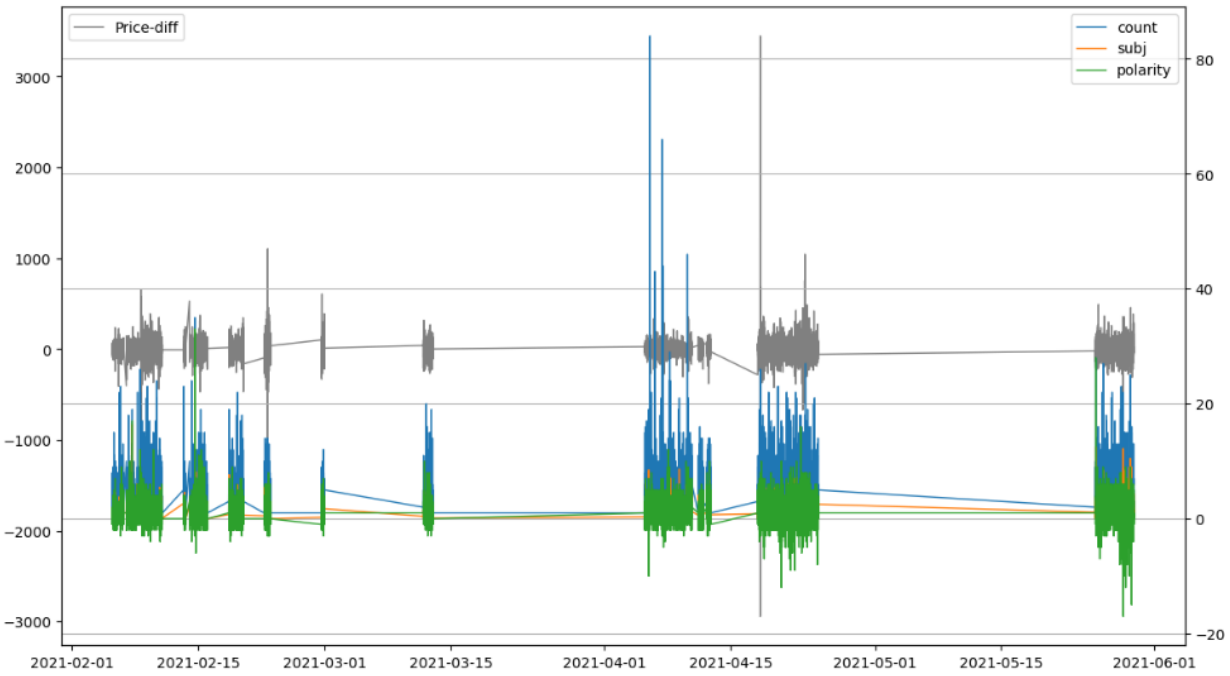


Figure (9) – Joined Bitcoin Twitter and price data (rows with zero tweets deleted).

The next step was resetting the data frame index and removing the timestamp as it's not necessary for the analysis.

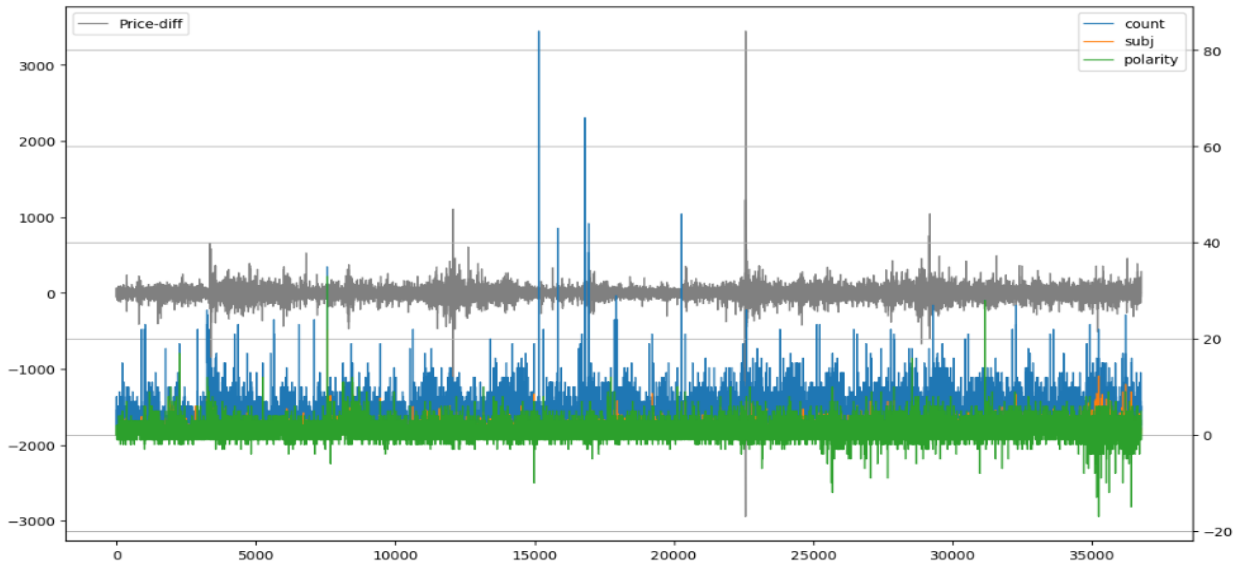


Figure (10) – Bitcoin Twitter Data, timestamp removed, data points stacked.

Data visualization of variables indicated the existence of outliers.

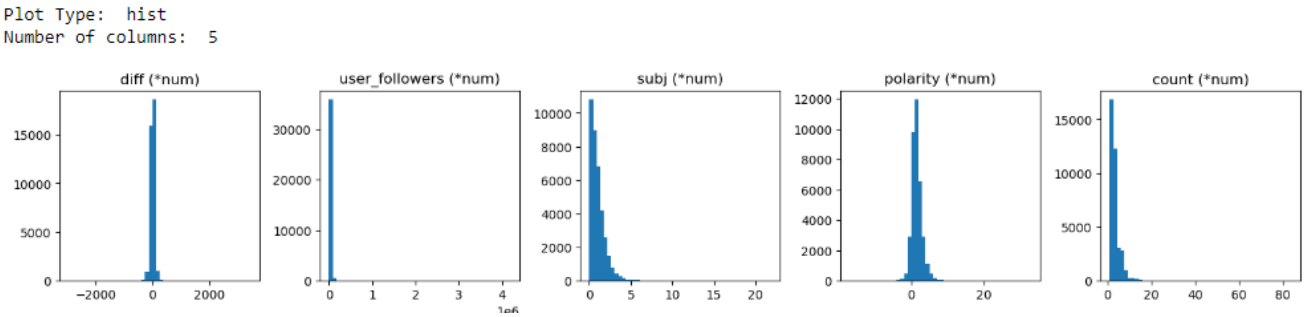


Figure (11) – Variables distribution.

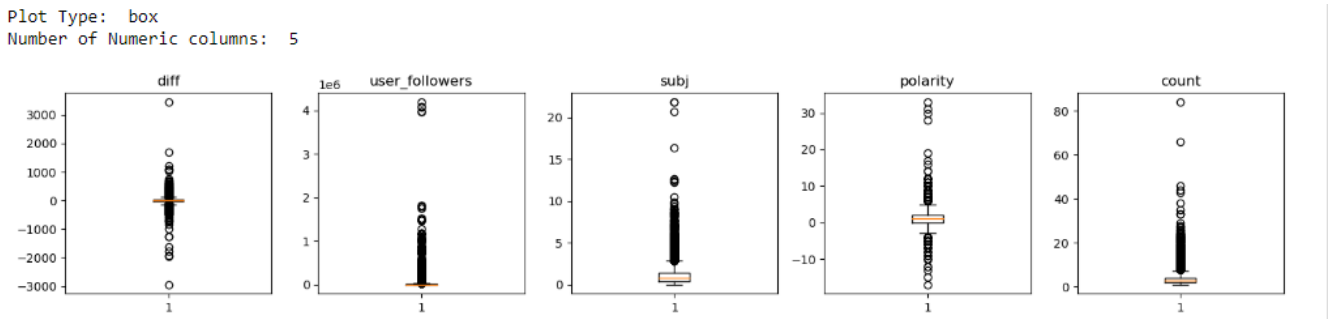


Figure (12) – Variables boxplots.

The final step was removing outliers from the variables, as the linear regression is known for its sensitivity to noise and outliers, dropping the outliers left some NaN values that had to be dropped.

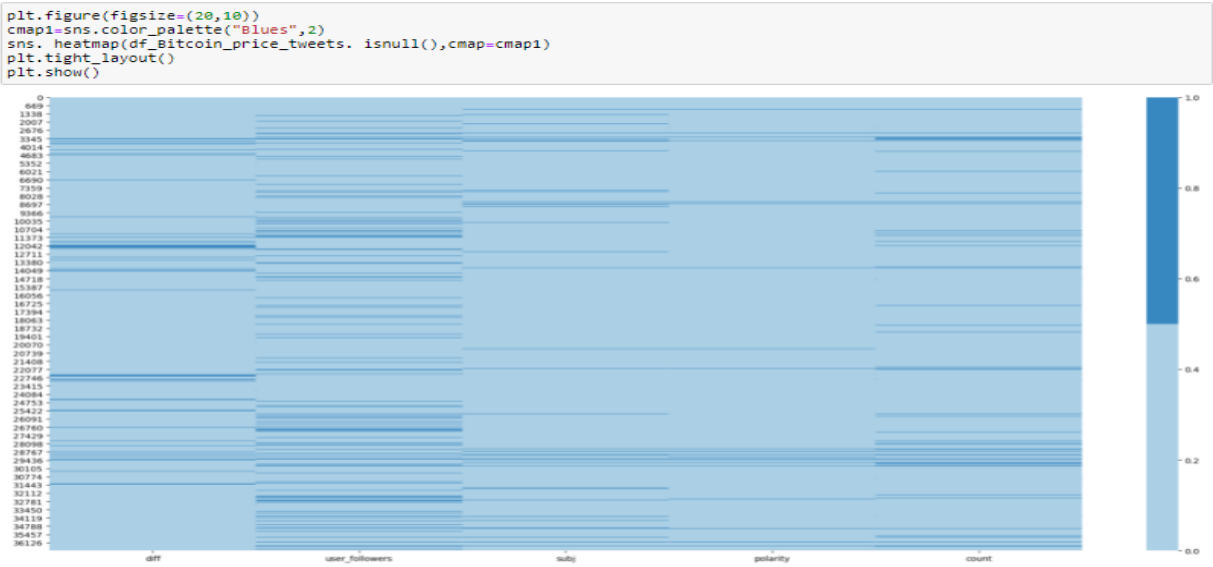


Figure (13) – Null values visualized (after removing outliers).

The final clean and prepared data frame consists of 29437 rows each of them represents one minute period and five columns contain both the independent variables [‘count’, ’polarity’, ’subj’ and’ user_followers’] for this one minute period and the corresponding dependent variable [’diff’] which represents the change of Bitcoin price.

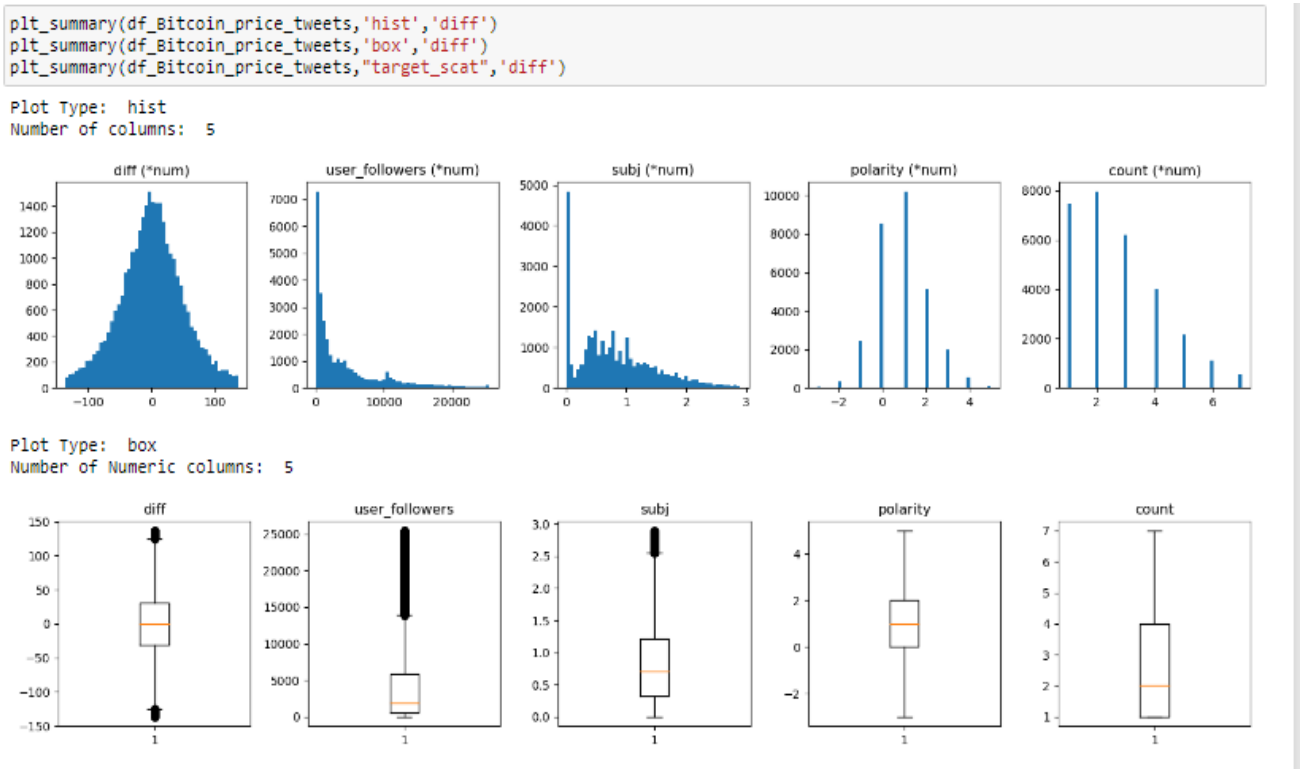


Figure (14) – Variables visualization (after removing outliers).

## Analysis

Linear regression has been implemented between the numeric inputs and the target variable “diff” which represents the change of Bitcoin price overtime at the same time points, While regression analysis is primarily used for prediction and forecasting, in this study it has been used to infer the significance of the relationships between the independent and dependent variables, as it can estimate the strength of the relationship between two variables related to the proportion of variance in the dependent variable that can be explained by the independent variable/variables. (Freedman, 2009).

A typical regression equation will have an intercept and one or more explanatory variables. The regression output includes a t-test, with p-1 degrees of freedom in the numerator and n-p in the denominator. (Freedman, 2009), the t-test estimates the true difference between two group means using the ratio of the difference in group means over the pooled standard error of both groups. (Bevans, 2020)

The null hypothesis will be that each coefficient vanishes, except for the intercept. (Freedman, 2009), The degree of significance of the t-statistics can be evaluated using the corresponding p-value which represents the probability of obtaining test results at least as extreme as the results actually observed, and the null hypothesis is correct, while a very small p-value means that such an extreme observed outcome would be very unlikely under the null hypothesis.

Linear Regression is a good tool to analyze the relationships among the variables, but it comes with some well-known disadvantages such as its sensitivity to noise, outliers, and overfitting. (Kumar, 2019)

Scipy.stats.OLS (ordinary least square) has been used for linear regression implementation, the modeling process started with (1) separating the dependent variable or the target (‘diff’) and the independent variable (‘polarity’), (2) adding a constant to the independent variable array, (3) splitting the inputs into train and test data sets (70% of the data used for training the model and 30% left out for testing), (4) initialized then fitted the OLS model to the train inputs (dependent and independent variables), (5) used the fitted model to estimate the parameters (model summary, t-statistic, p-value and input coefficient), the test data points have been used to evaluate the model.

```
X=df_Bitcoin_price_tweets.copy()

target='diff'
inputb='polarity'

# standardization
# scaler= StandardScaler()
# X_scaled = scaler.fit_transform(X)
# X=pd.DataFrame(X_scaled,columns=X.columns)

y = X[target].copy() # assigning the target to vector y
X = X[inputb]

X = sm.add_constant(X) #adding constant that will represent the intercept of the MLR model

# Splitting the data into train and test portions
X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(X, y, test_size = 0.30)

# LR implementation
linear_regression = sm.OLS(y_train,X_train)
fitted_model = linear_regression.fit()
residuals =fitted_model.predict(X_test)-y_test
itercept= fitted_model.params[0]
coefficient =fitted_model.params[1]
```

Figure (15) – Linear regression model.

```
# MLR summary:
print(fitted_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   diff   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                     2.645
Date:                Sun, 25 Jul 2021   Prob (F-statistic):              0.104
Time:                        19:59:40   Log-Likelihood:            -1.0983e+05
No. Observations:               20605   AIC:                         2.197e+05
Df Residuals:                   20603   BIC:                         2.197e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0607      0.434     -0.140      0.889      -0.911       0.790
polarity       0.4741      0.291      1.626      0.104      -0.097       1.045
==============================================================================
Omnibus:                        2.676   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.262   Jarque-Bera (JB):                2.725
Skew:                          -0.008   Prob(JB):                        0.256
Kurtosis:                       3.054   Cond. No.                         2.25
==============================================================================
```

Figure (16) – Linear regression model summary (notice p-value 0.104).

```
# Linear regression plot
fig, ax = plt.subplots(figsize=(8, 6))
#plt.yscale('log')
plt.scatter((X[inputb]),y,s=5,label='Data points')
line= range(int(X[inputb].min()),int(X[inputb].max()+1))
line2 = sm.add_constant(line)
plt.plot(line,fitted_model.predict(line2),'r--',label='Regression Line')
plt.legend(loc='upper right')
plt.ylabel(target)
plt.xlabel(inputb)
plt.grid()
plt.show()
```

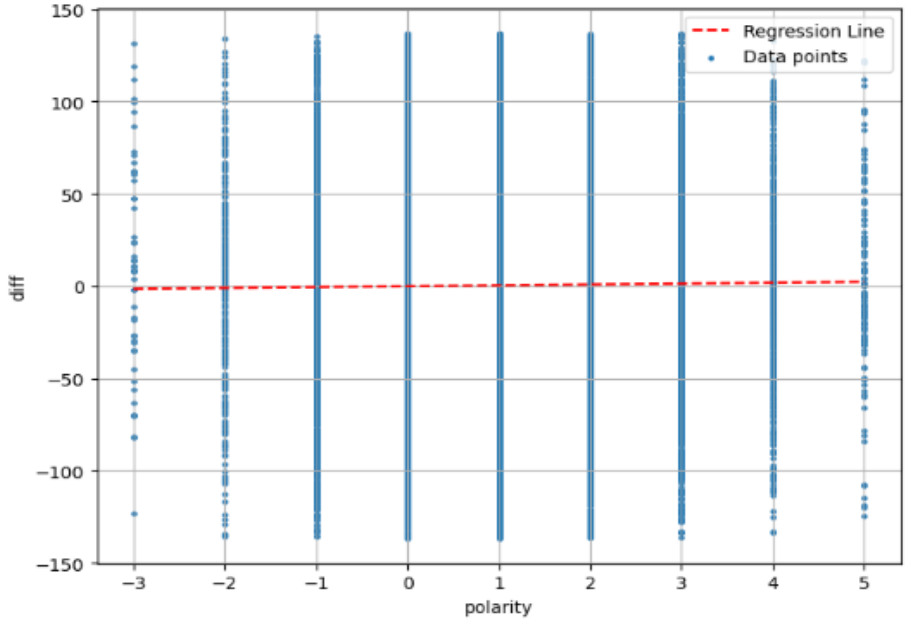


Figure (17) – Correlation between sentiments polarity and Bitcoin price differences (zero correlation), dotted red line represents the linear regression model.

```
#  MLR examination plots
residuals.hist(bins = 50)
plt.title('Residuals')
plt.xlabel('residual')
plt.show()
```

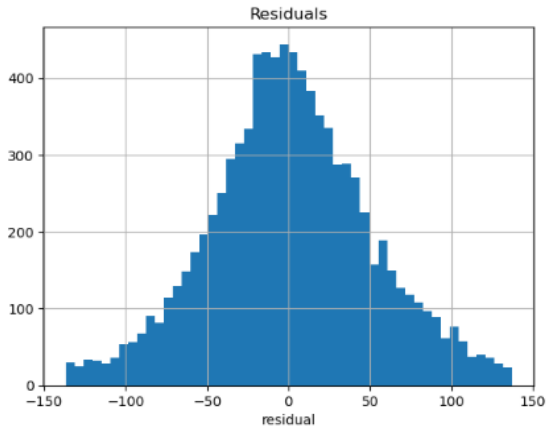


Figure (18) – Linear regression, test data residuals distribution.

Comprehensively, the linear regression has been implemented between the dependent variable ['diff'] and each independent variable separately ['count', 'user_followers', 'subj', 'polarity'], then every possible pair of input variables, every possible combination of three variables and so on using the explained steps (1 to 5), Two user_defined functions (linear_reg_test( ) and all_targets_all_inputs( ) - (Appendix**)** ) have been created and used to loop through all possible combinations and the total number of tests was 15 as summarized in the following table, the maximum $R^2$ obtained is approximately zero (0.000185)

| Test_no. for target | Target | R2 | No. of inputs | coefs_list | p-values |
|---|---|---|---|---|---|
| 1 | diff | 0.000004 | 1 | count 0.067335, | count 0.742073, |
| 2 | diff | 0.000049 | 1 | user_followers -0.000065, | user_followers 0.253961, |
| 3 | diff | 0.000049 | 1 | subj 0.538057, | subj 0.252352, |
| 4 | diff | 0.000045 | 1 | polarity 0.280032, | polarity 0.275428, |
| 5 | diff | 0.000048 | 2 | count 0.214765, user_followers -0.000057, | count 0.335329, user_followers 0.360279, |
| 6 | diff | 0.000097 | 2 | count -0.252177, subj 0.966731, | count 0.336325, subj 0.108665, |
| 7 | diff | 0.000132 | 2 | count -0.102538, polarity 0.514895, | count 0.643626, polarity 0.064380, |
| 8 | diff | 0.000147 | 2 | user_followers -0.000094, subj 0.689390, | user_followers 0.106152, subj 0.150652, |
| 9 | diff | 0.000146 | 2 | user_followers -0.000089, polarity 0.345873, | user_followers 0.120886, polarity 0.179376, |
| 10 | diff | 0.000082 | 2 | subj 0.460455, polarity 0.177298, | subj 0.411837, polarity 0.561699, |
| 11 | diff | 0.000156 | 3 | count -0.235845, user_followers -0.000033, subj 1.155264, | count 0.401813, user_followers 0.601976, subj 0.056153, |
| 12 | diff | 0.000098 | 3 | count 0.014558, user_followers -0.000050, polarity 0.365392, | count 0.951879, user_followers 0.429089, polarity 0.188920, |
| 13 | diff | 0.000103 | 3 | count -0.212190, subj 0.782586, polarity 0.161240, | count 0.419465, subj 0.240958, polarity 0.599665, |
| 14 | diff | 0.000185 | 3 | user_followers -0.000067, subj 0.557448, polarity 0.301649, | user_followers 0.248866, subj 0.327439, polarity 0.322805, |
| 15 | diff | 0.000163 | 4 | count -0.056509, user_followers -0.000069, subj 0.907563, polarity 0.055152, | count 0.841532, user_followers 0.270820, subj 0.175814, polarity 0.857797, |

Figure(19) – Summary of 15 linear regression tests.

As indicated by the tests summary table and the detailed models' summaries, It can be concluded that the proportion of variance in the Bitcoin price changes or ‘diff’ that can be explained by the independent variables and specifically the estimated sentiments from tweets is too small and the corresponding high P-values (larger than 0.05) indicate a very high probability of obtaining test results without any influence from the estimated sentiments, and the null hypothesis couldn’t be rejected.

## Data Summary and Implications

The large p-value (larger than five percent) resulted from the linear regression t-test, the too-small coefficient, and the too-small correlation coefficient led to the failure to reject the null hypothesis stated “There is no correlation between the estimated sentiments from social media and the change in Bitcoin price” which still a valid assumption according to the results of the study.

consequently, indicate that the estimated sentiment polarity values of tweets mentioned Bitcoin and the changes in Bitcoin price don’t have a significant relationship, which answers the research question that had been asked as the subject of this research project.

It has to be mentioned that the analysis was limited due to some factors, for example, the analysis results represent the correlation at a relatively short time frame (one minute) while a significant interaction between social media and Bitcoin price changes is supposed to take longer time, also the data was not evenly distributed through the whole period, and using the Textblob python library for sentiment analysis added some uncertainty to the analysis as it tended to label most analyzed texts as neutral (zero sentiment polarity).

For future studies about the same subject, it is highly recommended to collect more comprehensive data sets, that allow using different time frames (or sample rate) longer than one minute (hours or one day) as this would provide a more well-rounded view, testing different sentiment analysis techniques can be very beneficial, another suggestion for future studies would be to include data from other social media platforms not only from Twitter.

## References

Antonopoulos, Andreas M. (April 2014). Mastering Bitcoin: Unlocking Digital Crypto-Currencies. O'Reilly Media. ISBN 978-1-4493-7404-4.

Bevans, Rebecca. (2020) An introduction to t-tests. Retrieved July 18th, 2021 from https://www.scribbr.com/statistics/t-test/

Browne, Ryan. (2021) Bitcoin spikes 20% after Elon Musk adds #bitcoin to his Twitter bio, cnbc.com, Retrieved July 18th, 2021 from https://www.cnbc.com/2021/01/29/bitcoin-spikes-20percent-after-elon-musk-adds-bitcoin-to-his-twitter-bio.html

Colianni, Stuart et. al. (2015) Algorithmic Trading of Cryptocurrency Based on Twitter Sentiment Analysis. Retrieved July 18th, 2021 from http://cs229.stanford.edu/proj2015/029_report.pdf

David A. Freedman (27 April 2009). Statistical Models: Theory and Practice. Cambridge University Press. ISBN 978-1-139-47731-4.

Davis, Josua. (2011) The Crypto-Currency, Retrieved July 18th, 2021 from https://www.newyorker.com/magazine/2011/10/10/the-crypto-currency

Documentation of Python packages: pandas, matplotlib, numpy, sklearn, and textblob, Retrieved July 18th, 2021

Gulsen, Furkan (2021) Bitcoin Sentiment Analysis, Retrieved July 18th, 2021 from https://www.kaggle.com/codeblogger/bitcoin-sentiment-analysis

Hayes, Adam. (2021) Closing Price, Retrieved July 18th, 2021 from https://www.investopedia.com/terms/c/closingprice.asp

James, Sangeetha. (2018) Stackoverflow question, Retrieved July 18th, 2021 from https://stackoverflow.com/questions/51209514/how-does-textblob-calculate-sentiment-polarity-how-can-i-calculate-a-value-for

Jain, Shubham. (2018) Natural Language Processing for Beginners: Using TextBlob, Retrieved July 18th, 2021 from https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/

Kumar, Naresh. (2019) Advantages and Disadvantages of Linear Regression in Machine Learning, Retrieved July 18th, 2021 from

https://theprofessionalspoint.blogspot.com/2019/05/advantages-and-disadvantages-of-linear.html

Krish C. Naik. (2020) Natural-Language-Processing, Retrieved July 18th, 2021 from https://github.com/krishnaik06/Natural-Language-Processing/

Kaushik, Suresh. (2021) Bitcoin Tweets, Tweets with trending #Bitcoin and #btc hashtag, Retrieved July 18th, 2021 from https://www.kaggle.com/kaushiksuresh147/bitcoin-tweets

Lifestyle dictionary. (n.d.) Retrieved July 18th, 2021 from http://www.lifestyledictionary.com/social-media-trends.html

Maxwells (2021) BTCUSD, Bitcoin USD M1 M5 csv 2011-2021 Retrieved July 18th, 2021 from https://www.kaggle.com/maxwells/btcusd

Simple stock trading. (n.d.) Why is a stock close price more important than a high or a low in chart analysis, Retrieved July 18th, 2021 from https://www.simple-stock-trading.com/why-is-a-stock-close-price-more-important-than-high-or-low-in-chart-analysis/

TextBlob. (n.d.). Simplified Text Processing, Retrieved July 18th, 2021 from https://textblob.readthedocs.io/en/dev/

White, Swede. (2018) Analyzing Correlations between #BTC Tweets and Bitcoin Trading Using Natural Language Processing. (Wolfram Technology Conference), Retrieved July 18th, 2021 from https://www.youtube.com/watch?v=DNU2SW4Lb2Y

# Appendix

```
#importing the needed libraries
import pandas as pd
#modified Pandas options to fully display the large dataset when needed
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import re
from textblob import TextBlob

import scipy.stats as stats
import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn import linear_model
import statsmodels.api as sm

import warnings
warnings.filterwarnings('ignore') # Ignore warning messages for readability
#-----------------------------------------------------------------------------------------------------------------------
#-----------------------------------------------------------------------------------------------------------------------

# FUNCTIONS:

### Dataset Cleaning - tweets cleaning
# Edited after (Naik,2020)

def cleantxt(text):
    text=str(text)
    text = text.lower() #Converting Capital chars into small
    text = re.sub(r'@[a-zA-Z0-9]+', ' ', text) #Removing @ mentions
    text = re.sub("#bitcoin", 'bitcoin', text) # removeing the '#' from bitcoin
    text = re.sub("#btc", 'btc', text) # removeing the '#' from btc
    #text = re.sub('#[A-Za-z0-9]+', '', text) # removing any hastag or string starting with a '#'
    text = re.sub('\\n', '', text) # removing the '\n' string
    text = re.sub(r'https?:\/\/\S+', ' ', text) #Removing hyper links
    #text = re.sub(r'[^a-zA-Z]+', ' ', text) #Removing non ASCII chars
    text = text.split() #Removing extra spaces
    text = ' '.join(text)
    return text
#-----------------------------------------------------------------------------------------------------------------------

### Dataset Cleaning - date cleaning

def cleandate(date):
    date=str(date)
    date = re.sub(r'[^0-9\-\:]', ' ', date) #Removing non ASCII chars
    date = date.split() #Removing extra spaces
    date = ' '.join(date)
    return date
#-----------------------------------------------------------------------------------------------------------------------

def getsubjectivity(text):
    return TextBlob(text).sentiment.subjectivity
def getpolarity(text):
    return TextBlob(text).sentiment.polarity
#-----------------------------------------------------------------------------------------------------------------------
'''Custom function to inspect the data visually using Histogram/Bar, Boxplot and scattered plots
this function takes 3 parameters, 1st is DataFrame , 2nd is the preferable plot ("hist" , "box" , " scat" , "target_scat" or "target_histplot" )
The target argument will be used only in the case of target vs variables scat plots, and will not be used for other options
For Histograms with non-numeric columns , the function will use Bar charts (in Green color) instead of Histogram
"scat": plotting scatterd plot against the 1st column of the D.F.
"target_scat": plotting scatterd plot of num. variables against a 'target' column of the D.F.
"target_histplot":plotting histplot of numeric or countplot of cat. variables against the target' column
'''

def plt_summary(df_1,plt_type,target):
    numerics = ['uint8','uint16','uint32','int16', 'int32', 'int64', 'float16', 'float32', 'float64']

    if plt_type =='target_histplot':
        pltlist=df_1.columns
#         for number,cols in enumerate(df_1.columns):
#             if (len(df_1[cols].value_counts())>=20):
#                 pltlist.remove(cols)
        print("Plot Type:  "+plt_type) #Printing plot type.
        print("Number of columns:  "+str(len(pltlist))) #Printing the number of columns.


    elif plt_type =='hist':
        pltlist=df_1.columns
        print("Plot Type:  "+plt_type) #Printing plot type.
        print("Number of columns:  "+str(df_1.shape[1])) #Printing the number of columns.

    else :
        df_n = df_1.select_dtypes(include=numerics).copy()   # selecting only the numeric columns as it's suitable for mentioned plots
        pltlist=df_n.columns
        print("Plot Type:  "+plt_type) #Printing plot type.
        print("Number of Numeric columns:  "+str(df_n.shape[1])) #Printing the number of numeric columns.


    plt.style.use('default')
    plt.figure(figsize=(15,30))
    plt_rows = 11
    plt_cols = 5

    for n, col in enumerate(pltlist):   #enumerating each column of the data, starting from index
        b=n+1
        ax=plt.subplot2grid(shape=(plt_rows,plt_cols),loc=((int(np.ceil(b/plt_cols)-1)),((b-1)%plt_cols)))
        if plt_type == 'hist':
            if df_1[col].dtype in numerics:
                ax.hist(df_1.loc[:,col],bins = 50)  #plotting Histogram
                ax.set_title(str(col)+' (*num)')
            else:
                if len(df_1[col].value_counts())>2000:
                    ax.set_title(str(col)+' ('+str(len(df_1[col].value_counts()))+' unq val.)')
                    continue
                df_1.groupby(col).size().plot.bar(rot=0,color='g') #plotting Bar chart
                ax.set_title(str(col)+' (*categ. '+str(len(df_1[col].value_counts()))+' cat.)')
                if len(df_1[col].value_counts())>20:
                    ax.tick_params(bottom=False,labelbottom=False)

        elif plt_type == 'box':
            msk = ~np.isnan(df_n.loc[:,col])
            ax.boxplot(df_n.loc[msk,col])  #plotting Box plot (used a mask to filter the null values)
            #ax.boxplot(df.loc[:,col])
            ax.set_title(col)
        elif plt_type == 'scat':
            ax.scatter(df_n.iloc[:,0],df_n.loc[:,col],s=0.1,alpha=0.1) #plotting scatterd plot against the index of the D.F.
            ax.set_title(col)

        elif plt_type == 'target_scat':
```

```
            #ax.set_title(col)
            ax.set_xlabel(col)

        elif plt_type == 'target_hexbin':
            ax = df_n.plot.hexbin(ax=ax,x=col, y=target,gridsize=50, sharex=False) #plotting hexbin plot
            ax.set_title(col)

        elif (plt_type == 'target_histplot'):
            if (df_1[target].dtype not in numerics):
                ax.set_title(str(col)+' (*categ. '+str(len(df_1[col].value_counts()))+' cat.)')
                if (len(df_1[col].value_counts())<=700):
                    sns.countplot(ax=ax, data = df_1,x = col,hue = target,alpha=0.7) #plotting countplot of cat. variables against the 'target' column
                    if len(df_1[col].value_counts())>20:
                        ax.tick_params(bottom=False,labelbottom=False)
                elif (df_1[col].dtype in numerics):
                        sns.histplot(df_1, x=col, hue=target, element="poly",ax=ax) #plotting histplot of num. variables against the 'target' column
                        ax.set_title(str(col)+' (*num)')
#                         ax.plt.legend(fontsize='xx-small', title_fontsize='xx-small')
#                         plt.setp(ax.get_legend().get_texts(), fontsize='5') # for legend text
#                         plt.setp(ax.get_legend().get_title(), fontsize='32') # for legend title
            elif df_1[target].dtype in numerics:
                if (df_1[col].dtype not in numerics):
                    ax.set_title(str(col)+' (*categ. '+str(len(df_1[col].value_counts()))+' cat.)')
                    if (len(df_1[col].value_counts())<=5):
                        sns.histplot(df_1, x=target, hue=col, element="poly",ax=ax) #plotting histplot of num. variables against the 'target' column
                elif (df_1[col].dtype in numerics):
                    ax = df_1.plot.hexbin(ax=ax,x=target, y=col,gridsize=30, sharex=False, cmap="cubehelix_r") #plotting hexbin plot
#                     continue


    plt.tight_layout()
    plt.show()

    return
#-----------------------------------------------------------------------------------------------------------------------

def linear_reg_test(df,target,inputb):
    X=df.copy()

    # standardization
#     scaler= StandardScaler()
#     X_scaled = scaler.fit_transform(X)
#     X=pd.DataFrame(X_scaled,columns=X.columns)

    y = X[target].copy() # assigning the target to vector y
    X = X[inputb]

    X = sm.add_constant(X) #adding constant that will represent the intercept of the MLR model

    # Splitting the data into train and test portions
    X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(X, y, test_size = 0.10)

    # LR implementation
    linear_regression = sm.OLS(y_train,X_train)
    fitted_model = linear_regression.fit()
    residuals =fitted_model.predict(X_test)-y_test

    # MLR summary:
    print(fitted_model.summary())
    r2=fitted_model.rsquared;
    p_min=min(fitted_model.pvalues[1:])
    p_values=str(fitted_model.pvalues[1:]).replace('\n',', ')
    coefs=str(fitted_model.params[1:]).replace('\n',', ')


#     MLR examination plots
    residuals.hist(bins = 50)
    plt.title('Residuals')
    plt.xlabel('residual')
    plt.show()

    plt.scatter(X_test.index,residuals,s=20,alpha=0.4,c=y_test,cmap='plasma')
    plt.title('Residuals positions (colored by "'+ target +'")');plt.xlabel('Test data idx');plt.ylabel('Test data residuals')
    plt.colorbar();plt.grid('on')
    plt.show()

#     QQ-Plot
    fig, ax = plt.subplots(figsize=(4, 4))
    stats.probplot(residuals, plot=ax) # QQ-Plot
    plt.title('Residuals normality')
    plt.tight_layout()
    plt.show()

    return r2, p_min,p_values[:-14],coefs[:-14]
#-----------------------------------------------------------------------------------------------------------------------

def all_targets_all_inputs(df,targets,inputs,frame):
    test_number=0
    test_ser=[];r2=[];p_min=[];p_values=[];target_list=[];input_list=[];number_of_inputs=[];frame_list=[];coefs_list=[]
    for targeta in targets:
        inputs_list=[]
        # one input
        for input1 in inputs:
            inputb = input1
            test_number=test_number+1
            print('Time_Frame= ',frame,'Min')
            print('Test number= ',test_number)
            print('target= ',targeta)
            print('number of inputs= 1')
            print('inputs= ',inputb)
            #print(50*'-')
            inputs_list.append(inputb)
            r2_val, p_min_val,p_values_val,coefs_val=linear_reg_test(df,targeta,inputb)
            print(100*'-')
            coefs_list.append(coefs_val);frame_list.append(frame);test_ser.append(test_number);r2.append(r2_val);p_min.append(p_min_val);p_values.append(p_values_val);target_list.append(targeta);input_l


        # two inputs
        for input1 in inputs:
            for input2 in inputs:
                inputb=[]
                if ((input1 == input2)):
                    continue
                inputb.append(input1)
                inputb.append(input2)
                if ((set(inputb) in inputs_list)):
                    continue
                inputs_list.append(set(inputb))
                #print(iputs_list)
                test_number=test_number+1
                print('Time_Frame= ',frame,'Min')
                print('Test number= ',test_number)
                print('target= ',targeta)
                print('number of inputs= ',len(inputb))
                print('inputs= ',inputb)
```

```
                print(100*'-')
                coefs_list.append(coefs_val);frame_list.append(frame);test_ser.append(test_number);r2.append(r2_val);p_min.append(p_min_val);p_values.append(p_values_val);target_list.append(targeta);inp

        # three inputs
        for input1 in inputs:
            for input2 in inputs:
                for input3 in inputs:
                    inputb=[]
                    if ((input1 == input2) or (input1 == input3) or (input2 == input3)):
                        continue
                    inputb.append(input1)
                    inputb.append(input2)
                    inputb.append(input3)
                    if ((set(inputb) in inputs_list)):
                        continue
                    inputs_list.append(set(inputb))
                    #print(iputs_list)
                    test_number=test_number+1
                    print('Time_Frame= ',frame,'Min')
                    print('Test number= ',test_number)
                    print('target= ',targeta)
                    print('number of inputs= ',len(inputb))
                    print('inputs= ',inputb)
                    #print(50*'-')
                    r2_val, p_min_val,p_values_val,coefs_val=linear_reg_test(df,targeta,inputb)
                    print(100*'-')
                    coefs_list.append(coefs_val);frame_list.append(frame);test_ser.append(test_number);r2.append(r2_val);p_min.append(p_min_val);p_values.append(p_values_val);target_list.append(targeta)
        # four inputs
        for input1 in inputs:
            for input2 in inputs:
                for input3 in inputs:
                    for input4 in inputs:
                        inputb=[]
                        if ((input1 == input2) or (input1 == input3) or (input1 == input4) or (input2 == input3) or (input2 == input4) or (input3 == input4)):
                            continue
                        inputb.append(input1)
                        inputb.append(input2)
                        inputb.append(input3)
                        inputb.append(input4)
                        if ((set(inputb) in inputs_list)):
                            continue
                        inputs_list.append(set(inputb))
                        #print(iputs_list)
                        test_number=test_number+1
                        print('Time_Frame= ',frame,'Min')
                        print('Test number= ',test_number)
                        print('target= ',targeta)
                        print('number of inputs= ',len(inputb))
                        print('inputs= ',inputb)
                        #print(50*'-')
                        r2_val, p_min_val,p_values_val,coefs_val=linear_reg_test(df,targeta,inputb)
                        print(100*'-')
                        coefs_list.append(coefs_val);frame_list.append(frame);test_ser.append(test_number);r2.append(r2_val);p_min.append(p_min_val);p_values.append(p_values_val);target_list.append(targ

    data={'Test_no. for target':test_ser,'Target':target_list,'R2':r2,'No. of inputs':number_of_inputs,'min-p':p_min,'Inputs':input_list,'p-values':p_values,'Time_Frame':frame_list,'coefs_list':coefs_li
    df_all_results = pd.DataFrame(data=data,columns = ['Test_no. for target','Target','R2','No. of inputs','min-p','Inputs','coefs_list','p-values','Time_Frame'])

    df_all_results.set_index('Test_no. for target',drop=True,inplace=True)
    return df_all_results
#-----------------------------------------------------------------------------------------------------------------------------
#-----------------------------------------------------------------------------------------------------------------------------

# Data Loading

# Bitcoin twitter Data
#https://www.kaggle.com/kaushiksuresh147/bitcoin-tweets
#Bitcoin Tweets
df_Bitcoin_tweets = pd.read_csv('Bitcoin_tweets.csv')
df_Bitcoin_tweets.drop_duplicates(subset=['user_followers','date','text'],inplace=True)
df_Bitcoin_tweets.sort_values(by='date',inplace=True)
print('df_Bitcoin_tweets dimensions= ',df_Bitcoin_tweets.shape)
df_Bitcoin_tweets=df_Bitcoin_tweets[['user_followers','date','text']]
df_Bitcoin_tweets.head()
#-----------------------------------------------------------------------------------------------------------------------------

# Bitcoin Price Data
#BTCUSD
#https://www.kaggle.com/maxwells/btcusd
df_Bitcoin_Price= pd.read_csv('BTCUSD_1.csv')
df_Bitcoin_Price['time'] = pd.to_datetime(df_Bitcoin_Price['time'])
# Set time column as index
df_Bitcoin_Price.set_index('time',drop=True,inplace=True)
print('df_Bitcoin_Price dimensions= ',df_Bitcoin_Price.shape)
df_Bitcoin_Price=df_Bitcoin_Price[['close']]
df_Bitcoin_Price.tail()
#-----------------------------------------------------------------------------------------------------------------------------

# Plot and show the time series on axis ax
fig, ax = plt.subplots(figsize = (14,8))
plt.plot(df_Bitcoin_Price.index,
         (df_Bitcoin_Price[['close']]),
         linewidth=1,)
#plt.ylim([0,3500])
plt.legend(['Price-diff'],loc='upper left')

plt.grid()
plt.show()
#-----------------------------------------------------------------------------------------------------------------------------

df_Bitcoin_tweets['text']=df_Bitcoin_tweets['text'].apply(cleantxt)
df_Bitcoin_tweets['date']=df_Bitcoin_tweets['date'].apply(cleandate)
df_Bitcoin_tweets.dropna(inplace=True)
df_Bitcoin_tweets.drop(df_Bitcoin_tweets[df_Bitcoin_tweets['text'].str.len()==0].index,inplace=True)

df_Bitcoin_tweets['date'] = pd.to_datetime(df_Bitcoin_tweets['date'])

# Set date column as index
df_Bitcoin_tweets.set_index('date',drop=False, inplace=True)
#-----------------------------------------------------------------------------------------------------------------------------

df_Bitcoin_tweets['subj']=df_Bitcoin_tweets['text'].apply(getsubjectivity)
df_Bitcoin_tweets['polarity']=df_Bitcoin_tweets['text'].apply(getpolarity)
df_Bitcoin_tweets['count']=1
#-----------------------------------------------------------------------------------------------------------------------------

df_Bitcoin_tweets['polarity'] = df_Bitcoin_tweets['polarity'].apply(lambda x: 1 if x > 0 else (-1 if x < 0 else x))
df_Bitcoin_tweets.head(10)
#-----------------------------------------------------------------------------------------------------------------------------

df_Bitcoin_tweets['polarity'].hist(bins=5)
#-----------------------------------------------------------------------------------------------------------------------------

df_Bitcoin_tweets=df_Bitcoin_tweets.resample('Min').sum()
#-----------------------------------------------------------------------------------------------------------------------------

# Plot and show the time series on axis ax
```

```
df_Bitcoin_tweets[['count','subj','polarity']].plot(ax=ax)
plt.grid()
plt.show()


df_Bitcoin_Price['diff']=df_Bitcoin_Price['close'].diff()
df_Bitcoin_Price=df_Bitcoin_Price[['diff']]
#-----------------------------------------------------------------------------------------------------------------------

start_time=max(df_Bitcoin_tweets.index.min(),df_Bitcoin_Price.index.min())
end_time=min(df_Bitcoin_tweets.index.max(),df_Bitcoin_Price.index.max())
df_Bitcoin_Price=df_Bitcoin_Price.loc[(df_Bitcoin_Price.index >= start_time) & (df_Bitcoin_Price.index  <= end_time)]
df_Bitcoin_tweets=df_Bitcoin_tweets.loc[(df_Bitcoin_tweets.index >= start_time) & (df_Bitcoin_tweets.index  <= end_time)]

df_Bitcoin_price_tweets = pd.concat([df_Bitcoin_Price, df_Bitcoin_tweets], axis=1)
#-----------------------------------------------------------------------------------------------------------------------

# Plot and show the time series on axis ax
fig, ax = plt.subplots(figsize = (14,8))
plt.plot(df_Bitcoin_price_tweets.index,
         (df_Bitcoin_price_tweets[['diff']]),
         color='gray',
         linewidth=1,)
#plt.ylim([0,3500])
plt.legend(['Price-diff'],loc='upper left')

ax.twinx()

plt.plot(df_Bitcoin_price_tweets.index,
         df_Bitcoin_price_tweets[['count','subj','polarity']],
         linewidth=1,)
#plt.ylim([0,100])
plt.legend(['count','subj','polarity'],loc='upper right')

plt.grid()
plt.show()
#-----------------------------------------------------------------------------------------------------------------------

df_Bitcoin_price_tweets['count'].replace(0.0, np.NaN,inplace=True) #
df_Bitcoin_price_tweets=df_Bitcoin_price_tweets[(~df_Bitcoin_price_tweets['count'].isnull() & ~df_Bitcoin_price_tweets['diff'].isnull())]
#-----------------------------------------------------------------------------------------------------------------------

# Plot and show the time series on axis ax
fig, ax = plt.subplots(figsize = (14,8))
plt.plot(df_Bitcoin_price_tweets.index,
         (df_Bitcoin_price_tweets[['diff']]),
         color='gray',
         linewidth=1,)
#plt.ylim([0,3500])
plt.legend(['Price-diff'],loc='upper left')

ax.twinx()

plt.plot(df_Bitcoin_price_tweets.index,
         df_Bitcoin_price_tweets[['count','subj','polarity']],
         linewidth=1,)
#plt.ylim([0,100])
plt.legend(['count','subj','polarity'],loc='upper right')

plt.grid()
plt.show()
#-----------------------------------------------------------------------------------------------------------------------

df_Bitcoin_price_tweets_1min=df_Bitcoin_price_tweets.copy()
#df_Bitcoin_price_tweets.index=df_Bitcoin_price_tweets.index.astype(str)
df_Bitcoin_price_tweets.reset_index(drop=True,inplace= True)
#-----------------------------------------------------------------------------------------------------------------------

print('df_Bitcoin_price_tweets dimensions= ',df_Bitcoin_price_tweets.shape)

# Plot and show the time series on axis ax
fig, ax = plt.subplots(figsize = (14,8))
plt.plot(df_Bitcoin_price_tweets.index,
         (df_Bitcoin_price_tweets[['diff']]),
         color='gray',
         linewidth=1,)
#plt.ylim([0,3500])
plt.legend(['Price-diff'],loc='upper left')

ax.twinx()

plt.plot(df_Bitcoin_price_tweets.index,
         df_Bitcoin_price_tweets[['count','subj','polarity']],
         linewidth=1,)
#plt.ylim([0,100])
plt.legend(['count','subj','polarity'],loc='upper right')

plt.grid()
plt.show()
#-----------------------------------------------------------------------------------------------------------------------

plt_summary(df_Bitcoin_price_tweets,'hist','diff')
plt_summary(df_Bitcoin_price_tweets,'box','diff')
plt_summary(df_Bitcoin_price_tweets,"target_scat",'diff')
#-----------------------------------------------------------------------------------------------------------------------

df_otl = df_Bitcoin_price_tweets.copy()

q1=df_otl.quantile(0.25)
q3=df_otl.quantile(0.75)
iqr=q3-q1
pmin=q1-iqr*1.5
pmax=q3+iqr*1.5
otl_cols = ['diff','count','polarity','subj','user_followers']
for n,col in enumerate(otl_cols):
    df_otl.loc[:,col]=df_otl.loc[:,col].where(df_otl.loc[:,col].between(pmin[col],pmax[col]))

df_Bitcoin_price_tweets = df_otl.copy()
#-----------------------------------------------------------------------------------------------------------------------

plt.figure(figsize=(20,10))
cmap1=sns.color_palette("Blues",2)
sns.heatmap(df_Bitcoin_price_tweets.isnull(),cmap=cmap1)
plt.tight_layout()
plt.show()
#-----------------------------------------------------------------------------------------------------------------------

df_Bitcoin_price_tweets.dropna(inplace=True)
df_Bitcoin_price_tweets.reset_index(drop=True,inplace= True)
print('df_Bitcoin_price_tweets dimensions= ',df_Bitcoin_price_tweets.shape)
#-----------------------------------------------------------------------------------------------------------------------

plt_summary(df_Bitcoin_price_tweets,'hist','diff')
plt_summary(df_Bitcoin_price_tweets,'box','diff')
plt_summary(df_Bitcoin_price_tweets,"target_scat",'diff')
```

```
X=df_Bitcoin_price_tweets.copy()

target='diff'
inputb='polarity'

# standardization
# scaler= StandardScaler()
# X_scaled = scaler.fit_transform(X)
# X=pd.DataFrame(X_scaled,columns=X.columns)

y = X[target].copy() # assigning the target to vector y
X = X[inputb]

X = sm.add_constant(X) #adding constant that will represent the intercept of the MLR model

# Splitting the data into train and test portions
X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(X, y, test_size = 0.30)

# LR implementation
linear_regression = sm.OLS(y_train,X_train)
fitted_model = linear_regression.fit()
residuals =fitted_model.predict(X_test)-y_test
itercept= fitted_model.params[0]
coefficient =fitted_model.params[1]
#-------------------------------------------------------------------------------------------------------------------

# MLR summary:
print(fitted_model.summary())
#-------------------------------------------------------------------------------------------------------------------

# Linear regression plot
fig, ax = plt.subplots(figsize=(8, 6))
#plt.yscale('log')
plt.scatter((X[inputb]),y,s=5,label='Data points')
line= range(int(X[inputb].min()),int(X[inputb].max()+1))
line2 = sm.add_constant(line)
plt.plot(line,fitted_model.predict(line2),'r--',label='Regression Line')
plt.legend(loc='upper right')
plt.ylabel(target)
plt.xlabel(inputb)
plt.grid()
plt.show()
#-------------------------------------------------------------------------------------------------------------------

#  MLR examination plots
residuals.hist(bins = 50)
plt.title('Residuals')
plt.xlabel('residual')
plt.show()
#-------------------------------------------------------------------------------------------------------------------

plt.scatter(X_test.index,residuals,s=10,alpha=0.4,c=y_test,cmap='plasma')
plt.title('Residuals positions (colored by "'+ target +'")');plt.xlabel('Test data index');plt.ylabel('Test data residuals')
plt.colorbar();plt.grid('on')
plt.show()
#-------------------------------------------------------------------------------------------------------------------

#  QQ-Plot
fig, ax = plt.subplots(figsize=(7, 4))
stats.probplot(residuals, plot=ax) # QQ-Plot
plt.title('Residuals normality')
plt.tight_layout()
plt.show()
#-------------------------------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------------------------------
targets= ['diff']
inputs=['count','user_followers','subj','polarity']
#-------------------------------------------------------------------------------------------------------------------

df_all_results=all_targets_all_inputs(df_Bitcoin_price_tweets,targets,inputs,'1')
#-------------------------------------------------------------------------------------------------------------------

df_all_results.drop(['min-p','Inputs','Time_Frame'],axis=1).sort_values(by='No. of inputs',axis=0,ascending=True)
#-------------------------------------------------------------------------------------------------------------------
```