# Final Project – Media and Bitcoin Price Movements

# Intro

Cryptocurrency is an emerging asset class that has experienced an exponential uptick in volume and attention since the last bull cycle of 2017. The following Report explores online media and its relationship to Bitcoin price fluctuation.

# Data and Sources

**Data Description:**

The main data set contains the daily exchange rate of US dollar vs Bitcoin prices from 2013 to 2021 in CSV format. The data is sourced from the U.S. Bitstamp exchange and provided by cryptodatadownload.com.

|  |  |  |
| --- | --- | --- |
| **Fields** | **Description** | **Example** |
| Date | Date of trading day for Bitcoin | 2/11/2021 |
| Open | Opening BTC price at start of day | 44807.58 |
| Close | Closing BTC price at end of day | 47969.51 |
| Low | Lowest BTC price recorded during day | 43994.02 |
| High | Highest BTC price recorded during day | 48678.9 |
| Volume | Amount of Bitcoin traded | 81,388,911,810 |

In addition to the Bitcoin price data from Bitstamp, I obtained Bitcoin tweet data and news headlines. The Bitcoin tweets dataset came from a pre-aggregated list on Kaggle. The uploader collected 16 million tweets containing Bitcoin or BTC from 2016-01-01 to 2019-03-29 and stored them in a large CSV. The news headlines also came from Kaggle and were aggregated into a CSV from Coindesk, WSJ and Reuters. The headlines were specific to Bitcoin or crypto.

# Preprocessing Steps:

* Read bitcoin prices, tweets, and news headlines into dataframes
* Cleaned each one, removing null values, renaming columns and assigning proper data types
* Transformed the date column in each of the three dataframes to have a matching type and format—necessary for merging the datasets
* Added some additional columns to the bitcoin price dataframe, including a logarithmic calculation of price difference (closing price minus opening price) for each day or row
* Cleaned the tweet content in the tweets dataframe and removed newline indicators from the text
* For the tweet and headline dataframes, the rows were grouped by date and the text content concatenated together when added to a single row

# Analysis Question:

* Can sentiment analysis of relevant news and tweets predict future movements in Bitcoin’s price?

# Program Description:

The program aimed to accomplish three main tasks: retrieve the three datasets, preprocess them, and conduct analysis after they are merged together. After the preprocessing steps were completed as outlined in the preprocessing section, the TextBlob package was utilized to conduct quick sentiment analysis of all 16 million tweets—concatenated together for each date—and all the news headlines. The sentiment analysis produced two metrics: polarity and subjectivity. The polarity measured how negative or positive the sentiment of the text appeared while the subjectivity measured the intensity of emotional or opinionated expression.

A picture containing table

Description automatically generated

After gathering the results, both the sentiment scores from the tweets dataframe and the news dataframe were outer-joined with the Bitcoin price dataframe based on matching dates. Then, any rows that contained NaN values were dropped, cutting the size of the dataframe down to 348 rows but with all the necessary data available in each.

# Results:

A correlation matrix revealed that the sentiment values did not yield strong correlations with changes in price.

Table

Description automatically generated

However, while weak in general, there was a stronger positive correlation between subjectivity scores and BTC trading volume. This suggests increased investor emotion may result in more trades taking place in a given day.

When running linear regression model, the results were not convincing. However, the r-squared value was highest when looking at BTC volume as the target and using headline and tweet sentiment scores as the predictor variables.

|  |  |  |
| --- | --- | --- |
| **Predictor Variables** | **Target** | **R-Squared** |
| **Headline and Tweet Sentiment Scores (Polarity and Subjectivity)** | **BTC Volume** | **0.14** |
| **Headline and Tweet Sentiment Scores (Subjectivity)** | **BTC Volume** | **0.12** |
| **Headline and Tweet Sentiment Scores (Polarity and Subjectivity)** | **Upward/Downward Price Trend** | **0.01** |

An attempt was made to run a Random Forest Regressor with training data and test data cut from the merged dataframe rows. But, due to the small number of rows after merging (348), and an unequal distribution of time periods, the results were not significant or worth recording.

# Conclusion:

While there wasn’t any strong correlation between media sentiment and direct uptrends or downtrends in price, higher subjectivity among the media seemed to show a somewhat significant correlation with increased trading volume. And when trading volume increases, so too does volatility—meaning high subjectivity may indicate a large movement in the price is occurring or going to soon. Whether that movement is toward the upside or downside, the data could not predict with any meaningful precision or accuracy.

Perhaps, a larger dataset of cryptocurrency news headlines from a broader range of sources would have helped. More filtering methods and cleaning of the tweet data may have yielded more compelling sentiment scores on average, and while the TextBlob package did a good job of handling the large volumes of text, it may have been beneficial try different analysis tools for sentiment to see if any of them provided better results. Classification of the texts into categories of positive, negative, or neutral also would have allowed for better prediction models and training sets.

Zooming in on an hourly basis rather than a daily basis may have improved results as well. Since daily fluctuations in price can be quite volatile for Bitcoin, it is possible that the tweets collected had a lot of positive and negative sentiment all in the span of just 24 hours, leading to a less decisive score in relation to the daily price movement.

# Program Code:

import numpy  
import pandas as pd  
import openpyxl  
import xlsxwriter  
import nltk  
from datetime import datetime  
from statistics import \*  
import numpy as np  
from textblob import TextBlob  
  
bitcoinPrices = pd.DataFrame()  
dfTweets = pd.DataFrame()  
dfHeadlines = pd.DataFrame()  
# Read in Bitcoin historical price CSV (collected from Bitstamp exchange),  
# skip first row and make second row column headers  
btcData = pd.read\_csv("Datasets/Bitstamp\_BTCUSD\_d.csv", skiprows=0, header=1)  
headlineData = pd.read\_csv("Datasets/Headline\_Crypto.csv")  
# tweets = pd.read\_csv("Datasets/tweets.csv", delimiter=';', skiprows=0, lineterminator='\n' )  
tweetData = pd.read\_csv('Datasets/tweets.csv', delimiter=';', skiprows=0, lineterminator='\n')  
  
bitcoinPrices = pd.DataFrame(btcData)  
# tweetData = pd.DataFrame(tweets)  
dfTweets = pd.DataFrame(tweetData[['timestamp', 'text\r']])  
dfTweets  
dfHeadlines = pd.DataFrame(headlineData)  
  
# dfTweets[:5]  
#  
# tweets = dfTweets['text\r']  
  
# print(bitcoinPrices[:5])  
# print (tweetData[:5])  
# print(tweetData[:5])  
# print(dfHeadlines[:5])  
  
# 10239 Drop rows from dfHeadlines dataframe after index 10,239 due to date formatting/missing value issues  
print(dfHeadlines[:-5])  
dfHeadlines.drop(dfHeadlines.index[10239:], inplace=True)  
# Drop rows with missing date values  
dfHeadlines.drop(dfHeadlines.index[2066:2070], inplace=True)  
dfHeadlines.drop(dfHeadlines.index[7569:7589], inplace=True)  
# print(dfHeadlines.iloc[[7589]])  
  
# print(dfTweets.iloc[['2019-11-23']])  
# Drop Unix column in Bitcoin price Dataframe since it's not needed  
bitcoinPrices.drop(columns='unix', inplace=True)  
bitcoinPrices.rename(columns={'date':'timestamp'}, inplace=True)  
# Calculate log difference between opening and closing btc price for each day  
bitcoinPrices['log\_diff'] = np.log(bitcoinPrices['close']) - np.log(bitcoinPrices['open'])  
  
# Add next day log price column  
bitcoinPrices['next\_day\_log'] = bitcoinPrices.log\_diff.shift(1)  
bitcoinPrices['prev\_day\_log'] = bitcoinPrices.log\_diff.shift(-1)  
bitcoinPrices['trend'] = np.sign(bitcoinPrices['log\_diff'])  
bitcoinPrices['next\_day\_trend'] = np.sign(bitcoinPrices['next\_day\_log'])  
bitcoinPrices.dropna(how='any', inplace=True)  
  
# Convert Date columns for bitcoin, tweets, and news headlines to same date\_time format  
# This will allow for merging the dataframes by matching dates in each row  
bitcoinPrices['timestamp'] = pd.to\_datetime(bitcoinPrices['timestamp'])  
bitcoinPrices['timestamp'] = bitcoinPrices['timestamp'].apply(lambda t: t.strftime('%Y-%m-%d'))  
  
dfHeadlines['Date'] = pd.to\_datetime(dfHeadlines['Date'])  
dfHeadlines['Date'] = dfHeadlines['Date'].apply(lambda t: t.strftime('%Y-%m-%d'))  
  
dfTweets['timestamp'] = pd.to\_datetime(dfTweets['timestamp'])  
dfTweets['timestamp'] = dfTweets['timestamp'].apply(lambda t: t.strftime('%Y-%m-%d'))  
dfTweets = dfTweets.sort\_values('timestamp')  
# Reset index after sorting  
dfTweets.reset\_index(drop=True, inplace=True)  
dfTweets.columns = ['timestamp', 'text']  
  
  
  
# Remove newlines in tweet text  
dfTweets.replace(r'\n', ' ', regex=True, inplace=True)  
dfTweets.replace(r'\\n', ' ', regex=True, inplace=True)  
dfTweets.replace(r'\r', ' ', regex=True, inplace=True)  
  
# dfTweets.groupby(['timestamp'])['text\r'].apply(','.join).reset\_index()  
dfTweetsGrouped = dfTweets.groupby(['timestamp'])['text'].apply(','.join).reset\_index()  
dfTweets = dfTweetsGrouped  
dfTweets.to\_csv('Datasets/tweetsSentimentAnalysisSmall.csv', index=False,  
 encoding='utf-8')  
  
dfTweets[:-5]  
  
tweets = dfTweets['text']  
  
  
# dfTweets.groupby(['timestamp'])  
  
# tweetTknzr = TweetTokenizer(strip\_handles=True, reduce\_len=True)  
# dfTweets['tokenized\_tweet'] = dfTweets.apply(lambda row: tweetTknzr.tokenize(row['text']), axis=1)  
# dfHeadlines['tokenized\_headline'] = dfHeadlines.apply(lambda row: word\_tokenize(row['headline']), axis=1)  
  
  
# Conduct sentiment analysis using TextBlob package on tweets and then on headlines  
# Define function to take input text and analyze sentiment  
def calc\_sentiment(txt):  
 blob = TextBlob(txt)  
 return blob.sentiment.polarity, blob.sentiment.subjectivity  
  
# calc sentiment polarity & subjectivity  
tweets\_sentiments = tweets.apply(calc\_sentiment)  
# Store polarity values in new column  
tweets\_polarity = tweets\_sentiments.apply(lambda x: x[0])  
# Store subjectivity values in new column  
tweets\_subjectivity = tweets\_sentiments.apply(lambda x: x[1])  
  
dfTweets['tweet\_polarity'] = tweets\_polarity  
dfTweets['tweet\_subjectivity'] = tweets\_subjectivity # create the series  
  
dfTweets.sample(10) # display 10 random rows  
  
# Write tweets dataframe with polarity and subjectivity to new csv  
# dfTweets[['timestamp', 'tweet\_polarity', 'tweet\_subjectivity', 'text']].to\_csv('Datasets/tweetsSentimentAnalysisSmall.csv',  
# index=False,  
# encoding='utf-8')  
dfHeadlines.columns = ['timestamp', 'headline']  
# Group headlines comma delimited by date  
dfHeadlinesGrouped = dfHeadlines.groupby(['timestamp'])['headline'].apply(','.join).reset\_index()  
dfHeadlines = dfHeadlinesGrouped  
  
# Repeat sentiment analysis for news headlines  
headlines = dfHeadlines['headline']  
  
# Conduct sentiment analyis using TextBlob package on tweets and then on headlines  
headline\_sentiments = headlines.apply(  
 calc\_sentiment) # calc sentiment polarity & subjectivity, return in a Series of tuples  
headline\_polarity = headline\_sentiments.apply(lambda x: x[0]) # new column of polarity  
headline\_subjectivity = headline\_sentiments.apply(lambda x: x[1]) # new column of subjectivity  
  
dfHeadlines['news\_polarity'] = headline\_polarity  
dfHeadlines['news\_subjectivity'] = headline\_subjectivity  
  
dfHeadlines.sample(10) # display 10 random rows  
  
# Write tweets dataframe with polarity and subjectivity to new csv  
# dfHeadlines.to\_csv('Datasets/headlinesSentimentAnalysis.csv', index=False, encoding='utf-8')  
  
  
# Sort values by date to prepare for merging  
dfTweets = dfTweets.sort\_values('timestamp')  
dfHeadlines = dfHeadlines.sort\_values('timestamp')  
bitcoinPrices = bitcoinPrices.sort\_values('timestamp')  
# Merge sentiment scores from tweets and news headlines with bitcoin price dataframe  
dfMerged = pd.DataFrame()  
dfMerged = pd.merge(bitcoinPrices, dfHeadlines[['timestamp', 'news\_polarity', 'news\_subjectivity', 'headline']], on='timestamp', how='outer')  
dfMerged = pd.merge(dfMerged, dfTweets[['timestamp', 'tweet\_polarity', 'tweet\_subjectivity', 'text']], on='timestamp', how='outer')  
  
dfMerged.dropna(how='any', inplace=True)  
# Sort merged df by date  
dfMerged = dfMerged.sort\_values('timestamp')  
# Reset Index  
dfMerged.reset\_index(drop=True, inplace=True)  
  
dfMergedClean = dfMerged  
# Remove columns with headlines and tweets  
dfMergedClean = dfMergedClean.drop(['text','headline'], axis = 1)  
corrMatrix = dfMergedClean.corr()  
print (corrMatrix)  
  
import numpy as np  
from sklearn.linear\_model import LinearRegression  
  
# Predict BTC Volume with all sentiment results as predictor variables  
x = np.array(dfMerged[['news\_polarity', 'news\_subjectivity', 'tweet\_polarity', 'tweet\_subjectivity']]).reshape((-1, 4)) # coefficient of determination: 0.14175362009736936  
  
# Predict BTC Volume with subjectivity results and then with polarity  
x = np.array(dfMerged[['news\_subjectivity', 'tweet\_subjectivity']]).reshape((-1, 2)) # coefficient of determination: 0.1251332357454148  
x = np.array(dfMerged[['news\_polarity', 'tweet\_polarity']]).reshape((-1, 2)) # coefficient of determination: 0.0850508771200924  
  
# y = np.array(dfMerged['trend'])  
y = np.array(dfMerged['Volume BTC'])  
  
model = LinearRegression()  
model.fit(x,y)  
model = LinearRegression().fit(x, y)  
  
r\_sq = model.score(x, y)  
print('coefficient of determination:', r\_sq)  
  
  
# Predict trend with all sentiment results as predictor variables  
x = np.array(dfMerged[['news\_polarity', 'news\_subjectivity', 'tweet\_polarity', 'tweet\_subjectivity']]).reshape((-1, 4)) # coefficient of determination: 0.014112806389795174  
# y = np.array(dfMerged['trend'])  
y = np.array(dfMerged['trend'])  
  
model = LinearRegression()  
model.fit(x,y)  
model = LinearRegression().fit(x, y)  
  
r\_sq = model.score(x, y)  
print('coefficient of determination:', r\_sq)  
  
  
# Predict BTC Price trend with all sentiment results as predictor variables  
x = np.array(dfMerged[['Volume BTC']]).reshape((-1, 1)) # coefficient of determination: 0.00893998336673818  
# y = np.array(dfMerged['trend'])  
y = np.array(dfMerged['trend'])  
  
model = LinearRegression(normalize=True)  
model.fit(x,y)  
model = LinearRegression().fit(x, y)  
  
r\_sq = model.score(x, y)  
print('coefficient of determination:', r\_sq)  
  
dfModelData = dfMerged  
dfModelData = dfMerged[['next\_day\_trend', 'trend', 'tweet\_polarity', 'tweet\_subjectivity', 'news\_polarity', 'news\_subjectivity']]  
# Drop any rows where trend equals 0 (likely from price input error or missing price change in data)  
dfModelData= dfModelData.loc[(dfModelData[['next\_day\_trend', 'trend']] != 0).all(axis=1)]  
# Use numpy to convert to arrays  
import numpy as np  
# Labels are the values we want to predict  
labels = np.array(dfModelData['trend'])  
# labels = np.array(dfModelData['next\_day\_trend'])  
# Remove the labels from the features  
# axis 1 refers to the columns  
features = dfModelData.drop(['next\_day\_trend', 'trend'], axis = 1)  
# Saving feature names for later use  
feature\_list = list(features.columns)  
# Convert to numpy array  
features = np.array(features)  
  
# Using Skicit-learn to split data into training and testing sets  
from sklearn.model\_selection import train\_test\_split  
# Split the data into training and testing sets  
train\_features, test\_features, train\_labels, test\_labels = train\_test\_split(features, labels, test\_size = 0.2, random\_state = 42)  
  
print('Training Features Shape:', train\_features.shape)  
print('Training Labels Shape:', train\_labels.shape)  
print('Testing Features Shape:', test\_features.shape)  
print('Testing Labels Shape:', test\_labels.shape)  
  
# Import the model we are using  
from sklearn.ensemble import RandomForestRegressor  
# Instantiate model with 1000 decision trees  
rf = RandomForestRegressor(n\_estimators = 1000, random\_state = 42)  
# Train the model on training data  
rf.fit(train\_features, train\_labels);  
  
# Use the forest's predict method on the test data  
predictions = rf.predict(test\_features)  
# Calculate the absolute errors  
errors = abs(predictions - test\_labels)  
# Print out the mean absolute error (mae)  
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')  
  
# Calculate mean absolute percentage error (MAPE)  
mape = 100 \* (errors / test\_labels)  
# Calculate and display accuracy  
accuracy = 100 - np.mean(mape)  
print('Accuracy:', round(accuracy, 2), '%.')

# Program Output:

**runfile('/Users/jspector/OneDrive/College/Masters/Data Science/IST 652 - Scripting for Data Analysis/Final Project/sentimentAnalysis.py', wdir='/Users/jspector/OneDrive/College/Masters/Data Science/IST 652 - Scripting for Data Analysis/Final Project')**

**<ipython-input-2-3c65f09fc472>:1: DtypeWarning: Columns (3) have mixed types.Specify dtype option on import or set low\_memory=False.**

**runfile('/Users/jspector/OneDrive/College/Masters/Data Science/IST 652 - Scripting for Data Analysis/Final Project/sentimentAnalysis.py', wdir='/Users/jspector/OneDrive/College/Masters/Data Science/IST 652 - Scripting for Data Analysis/Final Project')**

**Date Headline**

**0 2018-04-26T11:31:02+00:00 Square Books Small Profit for First Quarter of...**

**1 2018-05-03T15:30:59+00:00 Daily Volatility Decline? Bitcoin Has Seen $1K...**

**2 2018-05-03T15:30:59+00:00 Sell In May and Go Away? Not for Bitcoin Bulls**

**3 2018-05-02T21:10:20+00:00 Reddit to Relaunch Bitcoin Payments (And Add M...**

**4 2018-05-03T18:00:39+00:00 Bitcoin Futures Trading Questioned By Chinese ...**

**... ...**

**16596 \r\r\n Updated April 18, 2013 6:08 p.m. ET\r\r\n Money Transfers in Bitcoins?\r\r\n\t\t\r\r\n\t...**

**16597 \r\r\n April 16, 2013 8:13 p.m. ET\r\r\n Bitcoin Investors Hang On for the Ride**

**16598 \r\r\n April 16, 2013 7:19 p.m. ET\r\r\n Bits and Pieces**

**16599 \r\r\n April 15, 2013 6:45 p.m. ET\r\r\n Investors Go Shopping For Grocery Deliverers**

**16600 \r\r\n Updated April 12, 2013 3:14 p.m. ET\r\r\n Japan, Bitcoin and the Gold Bug's Lament**

**[16601 rows x 2 columns]**

**open high ... tweet\_polarity tweet\_subjectivity**

**open 1.000000 0.998302 ... 0.411985 0.617675**

**high 0.998302 1.000000 ... 0.409920 0.616299**

**low 0.996002 0.996447 ... 0.417785 0.621015**

**close 0.995769 0.998509 ... 0.412424 0.618617**

**Volume BTC 0.656283 0.664942 ... 0.290807 0.338494**

**Volume USD 0.346932 0.338027 ... 0.027799 0.331512**

**log\_diff -0.081267 -0.041394 ... 0.013233 0.011003**

**next\_day\_log -0.030254 -0.027500 ... -0.024370 -0.004065**

**prev\_day\_log 0.036951 0.038330 ... 0.065091 0.067463**

**trend -0.110718 -0.088308 ... -0.026095 -0.025194**

**next\_day\_trend -0.034493 -0.033242 ... -0.039718 -0.038721**

**news\_polarity 0.079300 0.080078 ... 0.130647 0.042007**

**news\_subjectivity 0.292285 0.294966 ... 0.165319 0.207591**

**tweet\_polarity 0.411985 0.409920 ... 1.000000 0.518482**

**tweet\_subjectivity 0.617675 0.616299 ... 0.518482 1.000000**

**[15 rows x 15 columns]**

**coefficient of determination: 0.0850508771200924**

**coefficient of determination: 0.014112806389795174**

**coefficient of determination: 0.00893998336673818**

**Training Features Shape: (273, 4)**

**Training Labels Shape: (273,)**

**Testing Features Shape: (69, 4)**

**Testing Labels Shape: (69,)**

**Mean Absolute Error: 0.93 degrees.**

**Accuracy: 85.42 %.**