# CSCE 5300 Project Report: **Telegram Crypto Analysis**

**Participants:**

- Poli Nemkova - poli.nemkova@unt.edu
  - Project lead, price-trends analyzer, Python programmer
- Wesley DeLoach - WesleyDeLoach@my.unt.edu
  - Python programmer
- Riyad Bin Rafiq - riyadrafiq@gmail.com
  - Python and UI programmer
- Richard Tran - RichardTran2@my.unt.edu
  - Python and UI programmer
- Syed Araib Karim - syedaraibkarim@my.unt.edu
  - Python programmer

**Workflow:**

- Communication with the team is done on a need base over: Discord channel (a call each Monday at 5pm, written communication on a need base); Zoom meetings.
- Version Control for the project: Github Repository
- Running the code: Google Colab & PyCharm.
- Collaboration for proposal & report: Google Doc & Google Slides.

**Project Abstract:**

Telegram is one of the biggest messaging platforms in the world with more than 500 million users. With its increased security and functionality of public channels, Telegram quickly attracted audiences from all over the world (especially citizens of non-democratic countries where news is censored). We wanted to identify how Telegram channels may affect the real world. We took prices of cryptocurrencies as an example because we have a lot of detailed data of trading and we found a few relevant, large Telegram channels focused on this topic.

Additionally, Telegram allows users to download all the chat from the public channels. Hence, it is easy to get the data corpus in a raw format.

In our project we were able to apply sentiment analysis, descriptive statistics, and data visualization to the public channels focused on cryptocurrency prices. We identified that the number of mentioning a certain currency in the channels is negatively correlated with its price.

**Data Specification:**

Our datasets was based on the five different cryptocurrency based channels from the Telegram app:

1. Binance
2. Bittrex
3. Huobi
4. Kucoin
5. Okex

We found a dataset which contains group messages from Jan, 2018 to Jan, 2021 and also, we collected a dataset of group messages for the 3 month of 2021.

**Dataset from Jan 2018 to Jan 2021:**
This dataset contains records of approximately 3+ million messages in the official top crypto exchanges Telegram groups from various users. Also, it contains data such as dates, number of views, shares, and many more.

| **Telegram group name** | **Link** | **Total messages** |
|---|---|---|
| Binance official group | https://t.me/binanceexchange | 650k |
| Kucoin official group | https://t.me/Kucoin_Exchange | 860k |
| Bittrex official group | https://t.me/OKExOfficial_English | 70k |
| Huobi official group | https://t.me/huobiglobalofficial | 550k |
| Okex official group | https://t.me/OKExOfficial_English | 1m |

**Dataset from Jan, 2021 to March 2021:**
We collected the latest group messages from Telegram api which allow the user to export the complete chat history of a group. We collected over 1.5m messages from various users and it contains similar features to the first dataset.

| Telegram group name | Total messages |
|---|---|
| Binance official group | 680k |

| | |
|---|---|
| Okex official group | 407k |
| Bittrex official group | 34k |
| Huobi official group | 20k |
| Kucoin official group | 367k |

**Features:**

There are 33 columns in the dataset for example message,id, type, from, from_id,duration_seconds, message_id, action and others. Our most crucial feature were:

1. Messages
2. Date
3. Type

The text in the message column was used for natural language processing and the number of mentions for different cryptocurrencies. The date was used to arrange our data and help us create various graphs showing the difference in the number of mentions per cryptocurrency.

**Project Timeline & Assignments:**

- Project proposal is due: March 9, 2021 (all team is involved)
- Review the sources and agreeing on the tasks: March 8 - 9, 2021 (Poli, Richard)
- Dataset preparation: March 10 - 24, 2021 (Richard, Riyad)
- Working Model code: March 15 - March 31, 2021 (Wesley, Poli, Karim)
- Model UI: March 31 - April 5, 2021 (Richard, Riyad)
- Preparing the project report: April 5 - 6, 2021 (all team is involved)
- Project Report Presentation: April 5 - 6, 2021 (all team is involved)

**Project Design:**

***Understanding and running analysis for dataset (Jan, 2018 - Jan, 2021)***

For the initial dataset and analyses, we followed the kaggle tutorial, *Trends in Crypto Space,* in our references (1). The dataset came from the five previously mentioned channels and ranged from Jan 2018 to Jan 2021. Much preprocessing of the raw data needed to be performed in order to use NLP (NLTK library) to look at the mentions of each cryptocurrency over time and store said data in a consolidated dataframe. Figure 1 denotes an example of the data obtained where we have 'number of mentions' over time - we are comparing BTC vs. DeFi in this particular example.

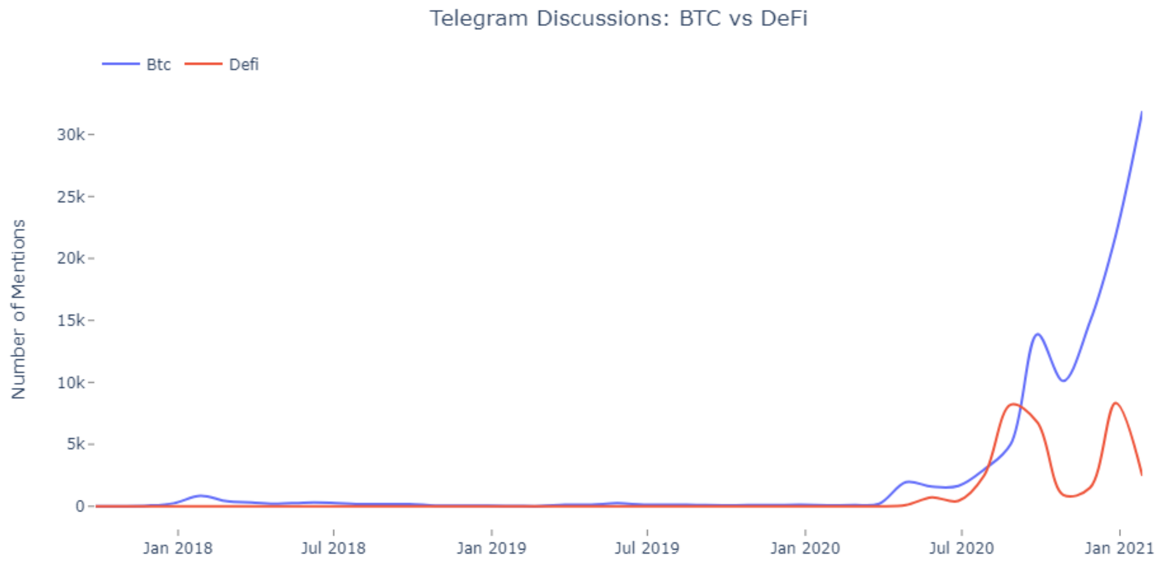
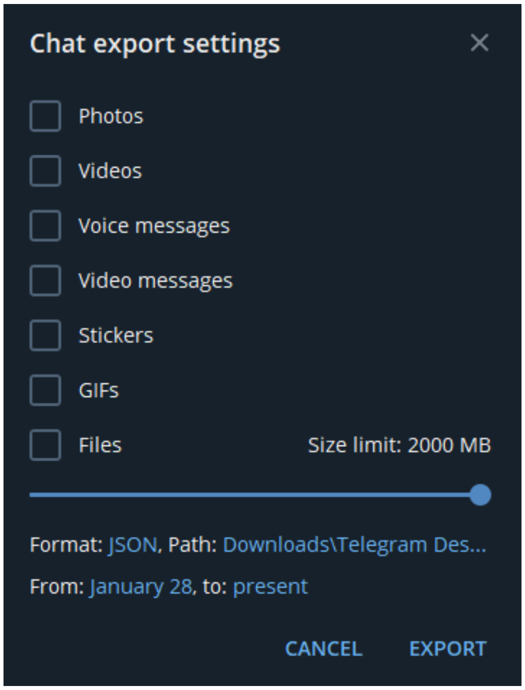


Figure 1: BTC vs DeFi Graph.

### *Collecting most recent dataset from Telegram channels*

We acquired the latest dataset from the five official telegram channels. Telegram channels allow you to export the chat history from specific dates. After selecting the Telegram channel of interest, one may go to settings in the top right corner and click on 'Chat Export Settings'. Figure 2 shows the different options available when exporting the data from the Telegram Desktop Application.

Here we extended the data analysis from the original dataset by looking at the past 3 months of January 2021 to March 2021 and exporting the data in a JSON format.



Chat export settings

□ Photos

□ Videos

□ Voice messages

□ Video messages

□ Stickers

□ GIFs

□ Files Size limit: 2000 MB

Format: JSON, Path: Downloads\Telegram Des...

From: January 28, to: present

Figure 2: Chat Export Settings on Telegram Channels.

### *Running analysis for the most recent dataset (Jan 2021 - March 2021)*

After gathering the newest dataset, we ran the same analysis to see the number mentions per day for different cryptocurrencies.

An example of our result is shown below. It shows the variation of the number of mentions for cryptocurrency Bitcoin and Ethereum:

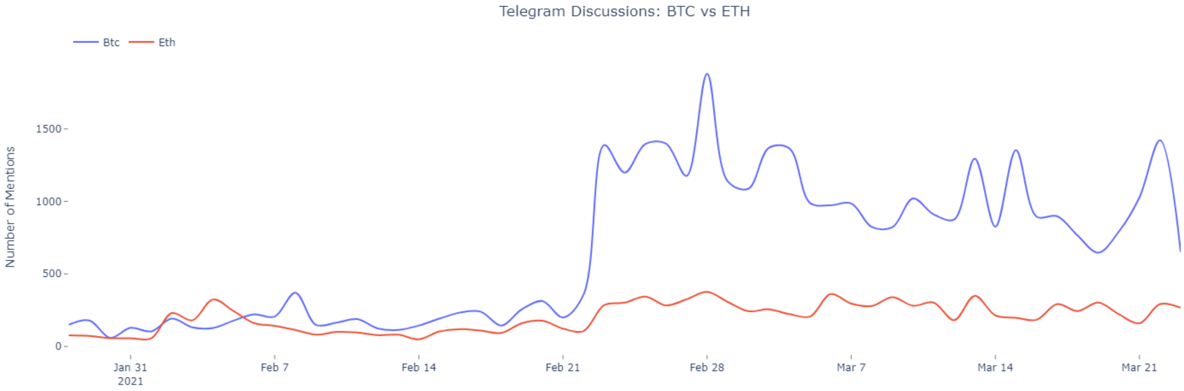


Figure 3:BTC vs ETH Telegram Discussions

Another example is below:

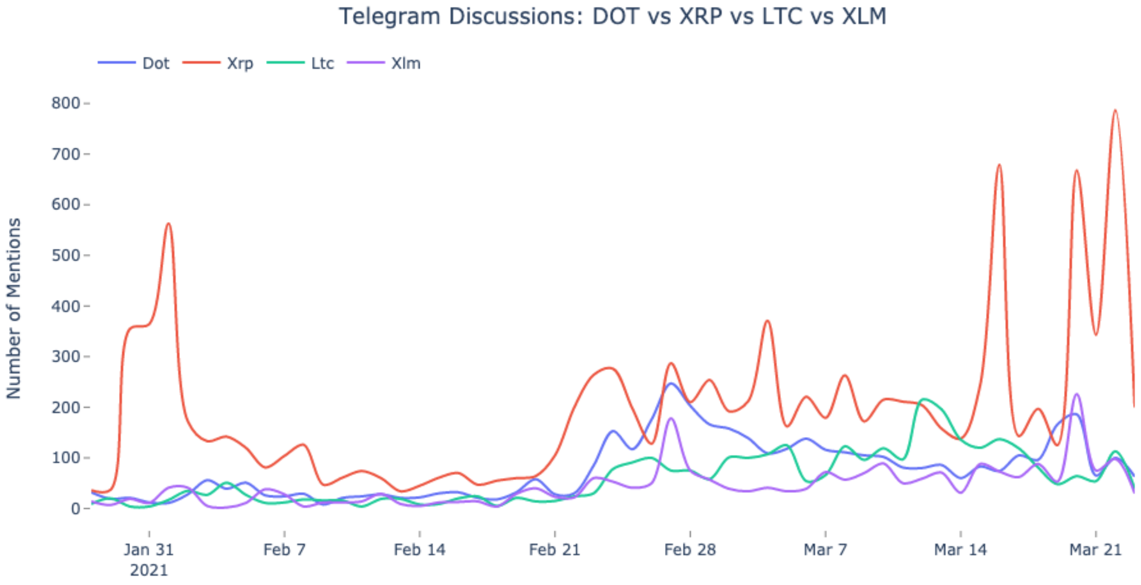
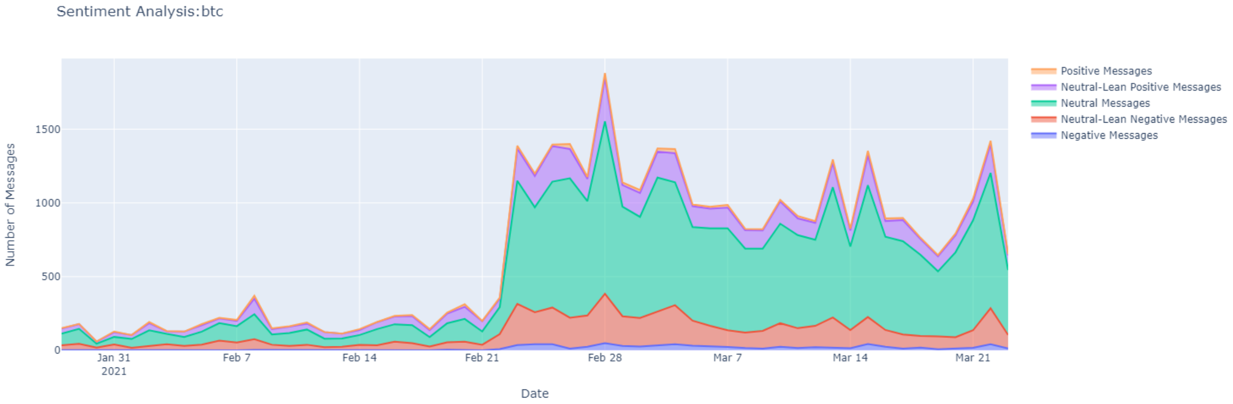


Figure 4: DOT vs XRP vx LTC vs XLM Telegram Discussions

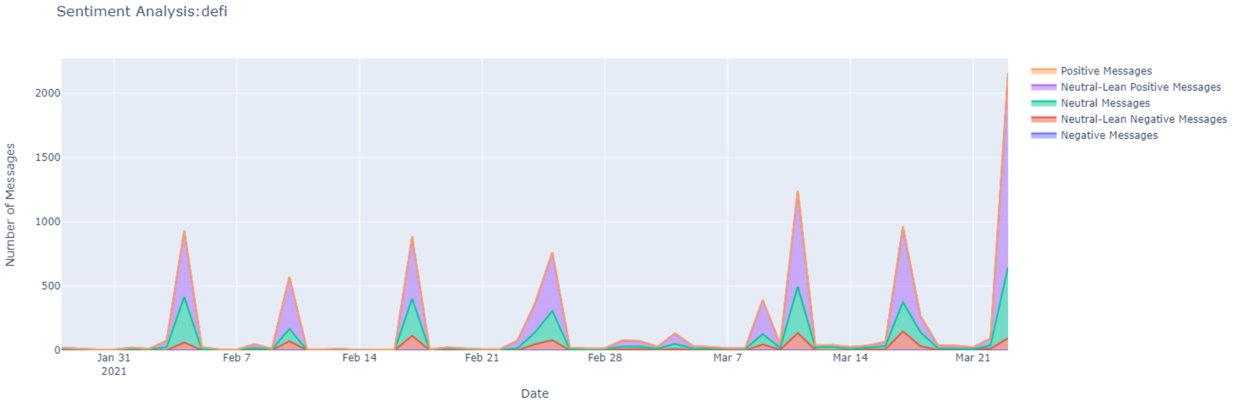
### *Sentiment Analysis*

We used the NLTK library to perform sentiment analysis on the most recent dataset. By using the SentimentIntensityAnalyzer object in the NLTK library, we got polarity scores for each message in the dataset that contained a positive, neutral, and negative score. These scores added up to 1 and were used to deem a message as ‘positive’, ‘negative’, 'neutral’, ‘neutral-lean negative’, or ‘neutral-lean positive’. If the positive or negative score was the largest, then the message would be labeled as positive or negative. If the neutral score was the largest, then the data would be labeled neutral if the positive and negative scores were the same, neutral-lean negative if the negative score was greater, or neutral-lean positive if the positive score was greater.

By using plotly to create area plots, we can see the tone of the messages surrounding different cryptocurrencies such as the ones shown below.



Sentiment Analysis over Bitcoin from Jan 2021-March 2021



Sentiment Analysis over Decentralized Finance from Jan 2021-March 2021

### ***Streamlit Interface***

Streamlit was the UI we decided to use to showcase our data analyses due to multiple reasons:a lightweight and easy-to-use framework, commonly used for machine learning and data science projects, and great integrations for data visualization libraries, such as plotly and matplotlib.

Figures 5 and 6 shows how well Streamlit integrates with these visualization libraries to allow the user to interact with the data from a locally hosted site. The next step if this model needed to be put into production would be to host this site in the cloud to allow access from anywhere.

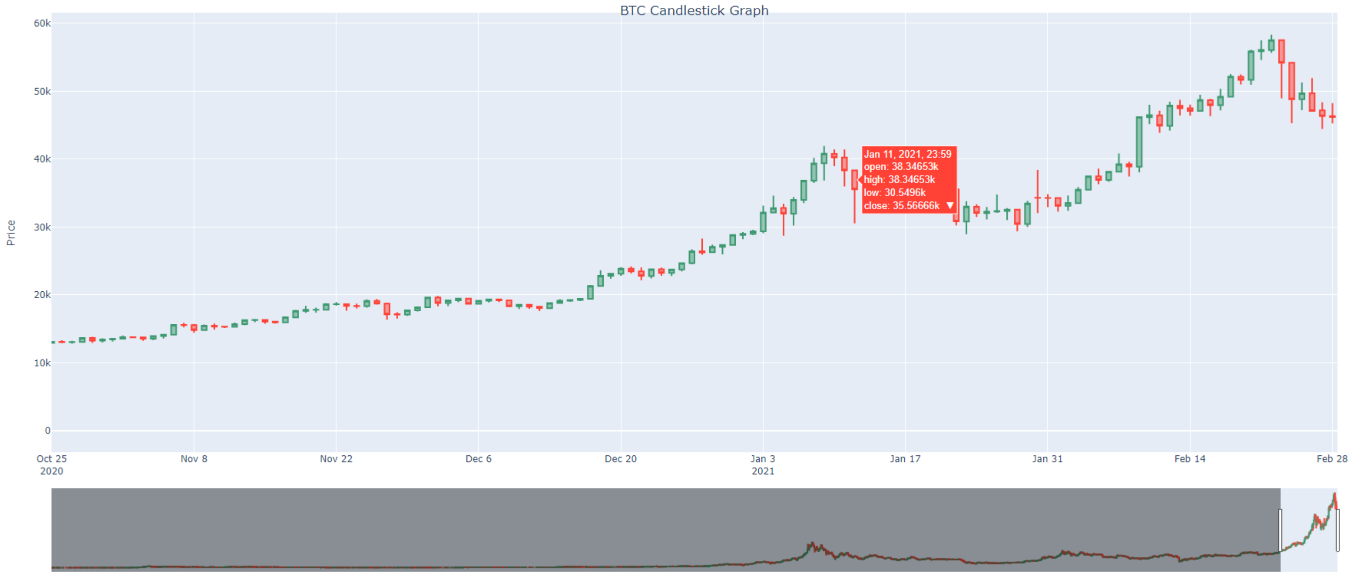


Figure 5: Bitcoin Candlestick Chart. An interactive plotly graph on the Streamlit UI allowing for in-depth financial analysis by looking at individual opening, closing, high, and low prices for each day when using the cursor. Date ranges can be adjusted with the shaded bar at the bottom.

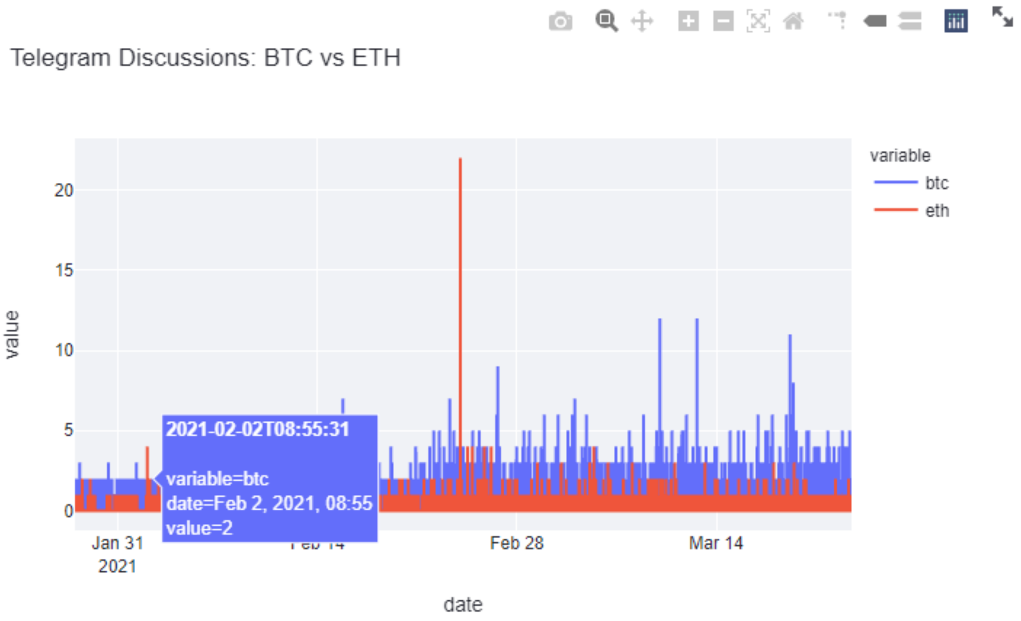


Figure 6: An interactive plotly chart on Streamlit UI which shows
Telegram Discussion Trends: BTC vs ETH

**Project Milestones:**

1. Gathering existing dataset for different cryptocurrency telegram channels
2. Gathering latest dataset from official cryptocurrency telegram channels
3. Running analysis for both dataset and comparing the number of mentions of different cryptocurrencies based on the text messages
4. Running NLP on the dataset and generating a positive, neutral or negative sentiment based on the text messages from users.
5. Gathering a dataset with prices for the major cryptocurrencies (Bitcoin - BTC, Litecoin - LTC, Ethereum - ETH).
6. Creating a Streamlit UI to display our data analysis.
7. Creating a Streamlit UI with interactive data to allow the user to make their own analyses.

**Resources and Related Projects:**

Tutorials:

1. https://www.kaggle.com/aagghh/trends-in-crypto-space-telegram-analysis
2. Sentiment Analysis of telegram chat history using Decision Tree Classifier model
3. Sentiment analysis of Telegram chat participants messages using Azure Cognitive Services
4. General Python Sentiment Analyzer

Streamlit UI:

1. https://docs.streamlit.io/en/stable/api.html#display-charts
2. https://towardsdatascience.com/data-visualization-using-streamlit-151f4c85c79a
3. https://medium.com/swlh/building-interactive-dashboard-with-plotly-and-streamlit-2c390bcfd41a

GitHub Repo for similar task:

1. Twitter Sentiment Analysis

NLTK Documentation for sentiment analysis:

1. NLTK Sentiment

Scholar Papers:

1. Data Collection and Sensemaking from Telegram: A Case Study of Ukrainian Political Leaders Channels and Chat Groups
    a. Research paper that explores features of Telegram and applies social network analysis and text analysis to study social behaviors affecting views on Ukranian politics.

Data:

1. Daily prices for BTC, LTC, ETH
2. Crypto Telegram Groups

**Repository / Archive:**

- Github Repository

# Code Appendix:

## telegramsentimentalanaylsis_group_project.py

```
# -*- coding: utf-8 -*-
"""TelegramSentimentalAnaylsis - Group Project.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/19V19hlHbeQ48jTzTOC2R07TpBnN2EvCk
"""

import numpy as np
import pandas as pd
import plotly
from plotly import graph_objs as go
import datetime

from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
import plotly.express as px
import matplotlib.pyplot as plt

from plotly import tools
import seaborn as sns
init_notebook_mode(connected=True)
from itertools import zip_longest
import string
import re

from nltk.corpus import stopwords
from nltk.util import ngrams
#for sentiment analysis
from nltk.sentiment import SentimentIntensityAnalyzer
import nltk
import zipfile

"""Download the specific truncated dataset here:
https://www.kaggle.com/aagghh/crypto-telegram-
groups?select=group_messages_binance.json

Upload the zip and extract the zip within the session:
"""

!unzip /content/expanded_dataset.zip

pd.set_option('display.max_colwidth',3000)
```

```
binance = pd.read_json('expanded_dataset/binance_1_28-3_23.json')
okex = pd.read_json('expanded_dataset/OKEx_1_28-3_23.json')
bittrex = pd.read_json('expanded_dataset/bittrex_1_28-3_23.json')
huobi = pd.read_json('expanded_dataset/huobi_1_28-3_23.json')
kucoin = pd.read_json('expanded_dataset/kucoin_1_28-3_23.json')

binance = binance['messages']
okex = okex['messages']
bittrex = bittrex['messages']
huobi = huobi['messages']
kucoin = kucoin['messages']

#Used to add sentiment data into the datasets
nltk.download('vader_lexicon')
sia = SentimentIntensityAnalyzer()

#Binance sentiment analysis
for messageInfo in binance:
  if not 'text' in messageInfo:
    continue

  if type(messageInfo['text']) is str:
    polarity_scores = sia.polarity_scores(messageInfo['text'])
  elif type(messageInfo['text']) is dict:
    polarity_scores = sia.polarity_scores(messageInfo['text']['text'])
  else:
    continue

  if (polarity_scores['neg'] > polarity_scores['neu']) and
(polarity_scores['neg'] > polarity_scores['pos']):
    messageInfo['sentiment'] = 'negative'
  elif (polarity_scores['neu'] > polarity_scores['pos']) and
(polarity_scores['pos'] > polarity_scores['neg']):
    messageInfo['sentiment'] = 'neutral-lean positive'
  elif (polarity_scores['pos'] == polarity_scores['neg']):
    messageInfo['sentiment'] = 'neutral'
  elif (polarity_scores['neu'] > polarity_scores['pos']):
    messageInfo['sentiment'] = 'neutral-lean negative'
  else:
    messageInfo['sentiment'] = 'positive'

#Okex sentiment analysis
for messageInfo in okex:
  if not 'text' in messageInfo:
    continue

  if type(messageInfo['text']) is str:
    polarity_scores = sia.polarity_scores(messageInfo['text'])
  elif type(messageInfo['text']) is dict:
    polarity_scores = sia.polarity_scores(messageInfo['text']['text'])
  else:
    continue

  if (polarity_scores['neg'] > polarity_scores['neu']) and
(polarity_scores['neg'] > polarity_scores['pos']):
    messageInfo['sentiment'] = 'negative'
  elif (polarity_scores['neu'] > polarity_scores['pos']) and
(polarity_scores['pos'] > polarity_scores['neg']):
    messageInfo['sentiment'] = 'neutral-lean positive'
  elif (polarity_scores['pos'] == polarity_scores['neg']):
    messageInfo['sentiment'] = 'neutral'
  elif (polarity_scores['neu'] > polarity_scores['pos']):
    messageInfo['sentiment'] = 'neutral-lean negative'
  else:
    messageInfo['sentiment'] = 'positive'

#Bittrex sentiment analysis
for messageInfo in bittrex:
  if not 'text' in messageInfo:
    continue

  if type(messageInfo['text']) is str:
    polarity_scores = sia.polarity_scores(messageInfo['text'])
  elif type(messageInfo['text']) is dict:
    polarity_scores = sia.polarity_scores(messageInfo['text']['text'])
  else:
    continue

  if (polarity_scores['neg'] > polarity_scores['neu']) and
(polarity_scores['neg'] > polarity_scores['pos']):
    messageInfo['sentiment'] = 'negative'
  elif (polarity_scores['neu'] > polarity_scores['pos']) and
(polarity_scores['pos'] > polarity_scores['neg']):
    messageInfo['sentiment'] = 'neutral-lean positive'
  elif (polarity_scores['pos'] == polarity_scores['neg']):
    messageInfo['sentiment'] = 'neutral'
  elif (polarity_scores['neu'] > polarity_scores['pos']):
    messageInfo['sentiment'] = 'neutral-lean negative'
  else:
    messageInfo['sentiment'] = 'positive'

#Huobi sentiment analysis
for messageInfo in huobi:
  if not 'text' in messageInfo:
    continue

  if type(messageInfo['text']) is str:
    polarity_scores = sia.polarity_scores(messageInfo['text'])
  elif type(messageInfo['text']) is dict:
    polarity_scores = sia.polarity_scores(messageInfo['text']['text'])
  else:
    continue

  if (polarity_scores['neg'] > polarity_scores['neu']) and
(polarity_scores['neg'] > polarity_scores['pos']):
    messageInfo['sentiment'] = 'negative'
  elif (polarity_scores['neu'] > polarity_scores['pos']) and
(polarity_scores['pos'] > polarity_scores['neg']):
    messageInfo['sentiment'] = 'neutral-lean positive'
  elif (polarity_scores['pos'] == polarity_scores['neg']):
    messageInfo['sentiment'] = 'neutral'
  elif (polarity_scores['neu'] > polarity_scores['pos']):
    messageInfo['sentiment'] = 'neutral-lean negative'
  else:
    messageInfo['sentiment'] = 'positive'

#Kucoin sentiment analysis
for messageInfo in kucoin:
  if not 'text' in messageInfo:
    continue

  if type(messageInfo['text']) is str:
    polarity_scores = sia.polarity_scores(messageInfo['text'])
  elif type(messageInfo['text']) is dict:
    polarity_scores = sia.polarity_scores(messageInfo['text']['text'])
  else:
    continue

  if (polarity_scores['neg'] > polarity_scores['neu']) and
(polarity_scores['neg'] > polarity_scores['pos']):
    messageInfo['sentiment'] = 'negative'
  elif (polarity_scores['neu'] > polarity_scores['pos']) and
(polarity_scores['pos'] > polarity_scores['neg']):
    messageInfo['sentiment'] = 'neutral-lean positive'
  elif (polarity_scores['pos'] == polarity_scores['neg']):
    messageInfo['sentiment'] = 'neutral'
  elif (polarity_scores['neu'] > polarity_scores['pos']):
    messageInfo['sentiment'] = 'neutral-lean negative'
  else:
    messageInfo['sentiment'] = 'positive'

print(binance.head())

# binance = pd.json_normalize(binance.to_dict(), record_path =['messages'])
binance = pd.json_normalize(binance)
okex = pd.json_normalize(okex)
bittrex = pd.json_normalize(bittrex)
```

```
huobi = pd.json_normalize(huobi)
kucoin = pd.json_normalize(kucoin)

binance = binance.rename(columns={'text': 'message'})
okex = okex.rename(columns={'text': 'message'})
bittrex = bittrex.rename(columns={'text': 'message'})
huobi = huobi.rename(columns={'text': 'message'})
kucoin = kucoin.rename(columns={'text': 'message'})

binance[110000:110005]

consolidated_data = huobi
consolidated_data = consolidated_data.append(okex)
consolidated_data = consolidated_data.append(bittrex)
consolidated_data = consolidated_data.append(binance)
consolidated_data = consolidated_data.append(kucoin)

import nltk
nltk.download('stopwords')

from nltk.corpus import stopwords

#date manipulation
stopwords = stopwords.words('english')
consolidated_data['CreationDate'] = pd.to_datetime(consolidated_data['date'])
consolidated_data['CreationYear'] = consolidated_data['CreationDate'].dt.year
consolidated_data['CreationMonth'] =
consolidated_data['CreationDate'].dt.month
consolidated_data['CreationMonth'] =
consolidated_data['CreationMonth'].apply(lambda x : "0"+str(x) if len(str(x))
< 2 else x)
#consolidated_data['CreationDay'] = "27"
consolidated_data['CreationDay'] = consolidated_data['CreationDate'].dt.day
#consolidated_data['MessageDate'] =
consolidated_data["CreationYear"].astype(str) +"-"+
consolidated_data["CreationMonth"].astype(str) +"-"+
consolidated_data["CreationDay"].astype(str)
consolidated_data['MessageDate'] = pd.to_datetime({'year' :
consolidated_data["CreationYear"],
                                                   'month' :
consolidated_data["CreationMonth"],
                                                   'day' :
consolidated_data["CreationDay"]})
#Putting this here as a temp until the original graphs look right
consolidated_data['MessageDate Sentiment'] =
consolidated_data["CreationYear"].astype(str) +"-"+
consolidated_data["CreationMonth"].astype(str) +"-"+
consolidated_data["CreationDate"].dt.day.astype(str)
consolidated_data['Message'] = consolidated_data['message'].fillna(" ")
```

```
## cleaning text
def clntxt(text):
    if isinstance(text, list):
      return " "
    text = text.lower()
    text = " ".join([c for c in text.split() if c not in stopwords])
    for c in string.punctuation:
        text = text.replace(c, " ")
    text = " ".join([c for c in text.split() if c not in stopwords])

    words = []
    ignorewords = ["www", "http","https" "com"]
    for wrd in text.split():
        if len(wrd) <= 2:
            continue
        if wrd in ignorewords:
            continue
        words.append(wrd)
    text = " ".join(words)
    return text

# counting mentions
def count_presence(txt, wrds):
    cnt = 0
    txt = " "+txt+" "
    for wrd in wrds.split("|"):
        if " "+wrd+" " in txt:
            cnt += 1
    return cnt

consolidated_data['CleanMessage'] = consolidated_data['Message'].apply(lambda
x : clntxt(x))

temp = consolidated_data.groupby('MessageDate')
print(temp)

import plotly.io as pio
pio.renderers.default = 'colab'
def plotit(listed, title):
    traces = []
    for model in listed:
        temp = consolidated_data.groupby('MessageDate').agg({model :
"sum"}).reset_index()
        #temp = temp.sort_values(by=['CreationDate'])
        print(temp)
        trace = go.Scatter(x = temp["MessageDate"], y = temp[model],
name=model.split("|")[0].title(), line=dict(shape="spline", width=2), mode =
"lines")
```

```
        print(trace)
        traces.append(trace)

    layout = go.Layout(
        paper_bgcolor='#fff',
        plot_bgcolor="#fff",
        legend=dict(orientation="h", y=1.1),
        title=title,
        title_x=0.5,
        xaxis=dict(
            gridcolor='rgb(255,255,255)',
            showgrid=True,
            showline=False,
            showticklabels=True,
            tickcolor='rgb(127,127,127)',
            ticks='outside',
            zeroline=False
        ),
        yaxis=dict(
            title="Number of Mentions",
            gridcolor='rgb(255,255,255)',
            showgrid=False,
            showline=False,
            showticklabels=True,
            tickcolor='rgb(127,127,127)',
            ticks='outside',
            zeroline=False
        ),
    )

    fig = go.Figure(data=traces, layout=layout)
    iplot(fig)

#btc vs eth
models = ["btc", "eth"]
for col in models:
    consolidated_data[col] = consolidated_data["CleanMessage"].apply(lambda x
: count_presence(x, col))
plotit(models, "Telegram Discussions: BTC vs ETH")

models = ["dot", "xrp", "ltc", "xlm"]
for col in models:
    consolidated_data[col] = consolidated_data["CleanMessage"].apply(lambda x
: count_presence(x, col))
plotit(models, "Telegram Discussions: DOT vs XRP vs LTC vs XLM")

#btc vs defi
models = ["btc", "defi"]
for col in models:
    consolidated_data[col] = consolidated_data["CleanMessage"].apply(lambda x
: count_presence(x, col))
plotit(models, "Telegram Discussions: BTC vs DeFi")

models = ["usdt", "dai"]
for col in models:
    consolidated_data[col] = consolidated_data["CleanMessage"].apply(lambda x
: count_presence(x, col))
plotit(models, "Stable coins: USDT vs DAI")

models = ["uniswap", "sushiswap", "1inch"]
for col in models:
    consolidated_data[col] = consolidated_data["CleanMessage"].apply(lambda x
: count_presence(x, col))
plotit(models, "DEX: UNISWAP vs Sushiswap vs 1inch")

#defi protocols
models = ["makerdao", "compound"]
for col in models:
    consolidated_data[col] = consolidated_data["CleanMessage"].apply(lambda x
: count_presence(x, col))
plotit(models, "DeFi protocols: MakerDAO vs Compound")

import plotly.graph_objects as graphpx

#print(consolidated_data.groupby(['MessageDate', 'sentiment']).shape[0])

def plotArea(model):
  sentiment_temp = consolidated_data.groupby(['MessageDate Sentiment',
'sentiment']).agg({model : 'sum'}).reset_index()
  neutral_sentiment = sentiment_temp[sentiment_temp['sentiment'] ==
'neutral']
  neutral_pos_sentiment = sentiment_temp[sentiment_temp['sentiment'] ==
'neutral-lean positive']
  neutral_neg_sentiment = sentiment_temp[sentiment_temp['sentiment'] ==
'neutral-lean negative']
  positive_sentiment = sentiment_temp[sentiment_temp['sentiment'] ==
'positive']
  negative_sentiment = sentiment_temp[sentiment_temp['sentiment'] ==
'negative']

  sentiment_plot = graphpx.Figure()
  sentiment_plot.add_trace(go.Scatter(
      name = 'Negative Messages',
      x = negative_sentiment['MessageDate Sentiment'],
      y = negative_sentiment[model],
      stackgroup = 'one'
  ))
  sentiment_plot.add_trace(go.Scatter(
      name = 'Neutral-Lean Negative Messages',
      x = neutral_neg_sentiment['MessageDate Sentiment'],
      y = neutral_neg_sentiment[model],
      stackgroup = 'one'
  ))

  sentiment_plot.add_trace(go.Scatter(
      name = 'Neutral Messages',
      x = neutral_sentiment['MessageDate Sentiment'],
      y = neutral_sentiment[model],
      stackgroup = 'one'
  ))

  sentiment_plot.add_trace(go.Scatter(
      name = 'Neutral-Lean Positive Messages',
      x = neutral_pos_sentiment['MessageDate Sentiment'],
      y = neutral_pos_sentiment[model],
      stackgroup = 'one'
  ))

  sentiment_plot.add_trace(go.Scatter(
      name = 'Positive Messages',
      x = positive_sentiment['MessageDate Sentiment'],
      y = positive_sentiment[model],
      stackgroup = 'one'
  ))

  sentiment_plot.update_layout(
      title = "Sentiment Analysis:"+model,
      xaxis_title='Date',
      yaxis_title='Number of Messages'
  )
  sentiment_plot.show()

plotArea('btc')

plotArea('defi')

plotArea('usdt')

plotArea('dai')

plotArea('uniswap')

plotArea('sushiswap')

plotArea('1inch')
plotArea('makerdao')

plotArea('compound')

#Beginning of Polina's price analysis
btc =
pd.read_csv('https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/
main/Bitcoin.csv')
ltc =
pd.read_csv('https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/
main/Litecoin.csv')
eth =
pd.read_csv('https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/
main/Ethereum.csv')

btc.head(5)

btc['mean']=(btc['high'] + btc['low'])/2
btc=btc.drop(['open','close','high','low'], axis=1)

ltc['mean']=(ltc['high'] + ltc['low'])/2
ltc=ltc.drop(['open','close','high','low'], axis=1)

eth['mean']=(eth['high'] + eth['low'])/2
eth=eth.drop(['open','close','high','low'], axis=1)

eth.head(5)

x = btc['mean']
y = ltc['mean']
z = eth['mean']

plt.plot(x)
plt.plot(y, color = 'r')
plt.plot(z, color = 'b')
plt.title('Bitcoin, Litecoin, and Ethereum mean prices')
plt.ylabel('Price')
plt.xlabel('Date')

x = btc['mean']
y = ltc['mean']
z = eth['mean']


plt.plot(y, color = 'r')
plt.plot(z, color = 'b')
plt.title('Litecoin and Ethereum mean prices')
plt.ylabel('Price')
plt.xlabel('Date')
```

```
#Pearson Correlations

ltc_btc_corr = ltc['mean'].corr(btc['mean'])
ltc_eth_corr = ltc['mean'].corr(eth['mean'])
btc_eth_corr = btc['mean'].corr(eth['mean'])
print('ltc_eth_corr', ltc_eth_corr, 'STRONG (>0.8)')
print('ltc_btc_corr', ltc_btc_corr)
print('btc_eth_corrr', btc_eth_corr, 'SIGNIFICANT (>0.6)')


###PREFERABLY ADD HERE CORRELATION WITH NUMBER OF MENTIONING OF THE
CORRESPONDING COIN AND I'TS PRICE

#took the data from the models above
btc_mentions = [ 150,  179,   60,  128,  104,  192,  131,  127,  178,  221,
206,  371,
                 149,  164,  189,  124,  114,  144,  192,  234,  240,  144,
256,  313,
                 200,  358, 1388, 1199, 1396, 1399, 1176, 1880, 1139, 1087,
1369, 1365,
                 989,  974,  986,  824,  822, 1020,  911,  876, 1295,  825,
1353,  895,
                 898,  764,  646,  792, 1028, 1421,  652]

eth_mentions = [ 77,  72,  57,  57,  56, 230, 180, 324, 244, 161, 142, 114,
81, 100,
                 96,  78,  81,  49, 103, 119, 109,  93, 160, 177, 123, 106,
285, 302,
                344, 283, 325, 376, 307, 243, 256, 223, 206, 361, 295, 279,
339, 281,
                303, 181, 349, 214, 198, 184, 292, 244, 302, 222, 160, 292,
267]

ltc_mentions = [  9,  20,   4,   4,  17,  35,  27,  51,  27,  11,  12,  18,
16,  16,
                  4,  19,  19,   8,   9,  20,  24,   5,  21,  14,  15,  24,
30,  78,
                 91, 100,  75,  75,  58, 101, 100, 107, 125,  54,  66, 123,
96, 119,
                 97, 215, 196, 136, 120, 137, 119,  81,  48,  64,  54, 113,
39]

btc_mentions = pd.DataFrame(btc_mentions)
eth_mentions = pd.DataFrame(eth_mentions)
ltc_mentions = pd.DataFrame(ltc_mentions)

#removing unneeded data from prices
btc_55 = btc[:54]
```

```
ltc_55 = ltc[:54]
eth_55 = eth[:54]



#plotting BTC means VS mentions
x = btc['mean']
y = btc_mentions


plt.plot(x, color = 'r')
plt.plot(y, color = 'b')
plt.title('BTC prices VS mentions in Telegram')
plt.ylabel('Price')
plt.xlabel('Date')

x = ltc['mean']
y = ltc_mentions


plt.plot(x, color = 'r')
plt.plot(y, color = 'b')
plt.title('LTC prices VS mentions in Telegram')
plt.ylabel('Price')
plt.xlabel('Date')

x = eth['mean']
y = eth_mentions


plt.plot(x, color = 'r')
plt.plot(y, color = 'b')
plt.title('ETC prices VS mentions in Telegram')
plt.ylabel('Price')
plt.xlabel('Date')

#btc_mentions = btc_mentions.stack()
#btc_mentions = btc_mentions.drop(,axis=1)
#btc_mentions.drop(columns=0, axis=1)
btc_mentions

btc_mentions

btc_55_mean.shape
print(btc_55_mean)

btc_55 = btc[:54]
btc_55_mean = btc_55['mean']
btc_55_mean.corr(btc_mentions)
```

```
btc_mentions.shape

btc.shape

btc.head(50)
```

## streamlit_app.py

```
import streamlit as st
import pandas as pd
import numpy as np
import plotly.graph_objects as go
import plotly.express as px


# titles and descriptions
st.title("Telegram Cryptocurrency Analysis")

st.header("Project looking at cryptocurrency trends over time")

st.image("https://img.freepik.com/free-photo/3d-rendering-bitcoin-other-crypto-currencies-led-glow-dark-glossy-glass-board-with-blockchain-data-dots-lines_163855-4.jpg?size=626&ext=jpg")


# sidebar
st.sidebar.title("Options")

st.sidebar.header("Cryptocurrency Symbols")

option = st.sidebar.selectbox("Select from the following cryptocurrencies:",
('AAVE', 'BNB', 'BTC', 'ADA', 'LINK', 'ATOM', 'CRO', 'DOGE', 'EOS', 'ETH',
'MIOTA', 'LTC', 'XMR', 'XEM', 'DOT', 'SOL', 'XLM', 'USDT', 'TRX', 'UNI',
'USDC', 'WBTC', 'XRP'))

# displaying the data
st.header(option)

'''
Dataset holds the following:

Name, Symbol, Date, High, Low, Opening Price, Closing Price
* Name/Symbol = Cryptocurrency
* Date = Observation date
* High = The highest price that day
* Low = The lowest price that day
* Open = The opening price that day
* Close = The closing price that day
* Volume = Volume of transactions that day
* Market Cap = Market capitalization in USD

[Link] (https://www.kaggle.com/sudalairajkumar/cryptocurrencypricehistory) to
the source of the dataset (Cryptocurrency Historical Prices).
'''

# crypto dictionary
if option == 'AAVE':
   filename = 'Aave'
elif option == 'BNB':
   filename = 'BinanceCoin'
elif option == 'BTC':
   filename = 'Bitcoin'
elif option == 'ADA':
   filename = 'Cardano'
elif option == 'LINK':
   filename = 'ChainLink'
elif option == 'ATOM':
   filename = 'Cosmos'
elif option == 'CRO':
   filename = 'CryptocomCoin'
elif option == 'DOGE':
   filename = 'Dogecoin'
elif option == 'EOS':
   filename = 'EOS'
elif option == 'ETH':
   filename = 'Ethereum'
elif option == 'MIOTA':
   filename = 'Iota'
elif option == 'LTC':
   filename = 'Litecoin'
elif option == 'XMR':
   filename = 'Monero'
elif option == 'XEM':
   filename = 'NEM'
elif option == 'DOT':
   filename = 'Polkadot'
elif option == 'SOL':
   filename = 'Solana'
elif option == 'XLM':
   filename = 'Stellar'
elif option == 'USDT':
   filename = 'Tether'
elif option == 'TRX':
   filename = 'Tron'
elif option == 'UNI':
   filename = 'Uniswap'
elif option == 'USDC':
   filename = 'USDCoin'
elif option == 'WBTC':
   filename = 'WrappedBitcoin'
elif option == 'XRP':
   filename = 'XRP'


# loading the dataframe
df = pd.read_csv(f"C:/Users/richa/Documents/Coding+/SPR21/csce 5300 (big
data)/telegram py/crypto_data/coin_{filename}.csv")

st.dataframe(df)



# sidebar options logic
st.header(option)

st.subheader("Classic Candlestick Chart for Financial Analysis")

st.write("Note: A Candlestick Chart has similar features to a boxplot.
However, the bottom and top whiskers denote "
        "the Low and High for each day, respectively. The green color
indicates a 'Bullish Candle Stick' where the "
        "Closing price was greater than the Opening price; red indicates a
'Bearish Candle Stick' where the Opening "
        "was greater than the Closing.")

# plotly candlestick graph
candlestick = go.Candlestick(x=df['Date'],
              open=df['Open'],
              high=df['High'],
              low=df['Low'],
              close=df['Close']
              )

figure = go.Figure(data=[candlestick])

figure.update_layout(
   title={
       'text': f"{option} Candlestick Graph",
       'y':0.9,
       'x':0.5,
       'xanchor': 'center',
       'yanchor': 'top'
       },
   xaxis_title="Date",
   yaxis_title="Price"
)

st.plotly_chart(figure)

### riyad's code ###
st.header('Past 3 Months Analysis')
st.subheader('Data and Analyses from 1/28/21 to 3/23/21')

@st.cache
def load_binance():
  col_names = ['id','date','from','from_id','text']
  data = pd.read_csv('binance.csv', names=col_names)
  data = data.iloc[1:]
  return data
@st.cache
def load_bittrex():
  col_names = ['id','date','from','from_id','reply_to_message_id','text']
  data = pd.read_csv('bittrex.csv', names=col_names)
  data = data.iloc[1:]
  return data
@st.cache
def load_huobi():
  col_names = ['id','date','from','from_id','text']
  data = pd.read_csv('huobi.csv', names=col_names)
  data = data.iloc[1:]
  return data
@st.cache
def load_kucoin():
  col_names = ['id','date','from','from_id','text']
  data = pd.read_csv('kucoin.csv', names=col_names)
  data = data.iloc[1:]
  return data
@st.cache
def load_OKEx():
  col_names = ['id','date','from','from_id','reply_to_message_id','text']
  data = pd.read_csv('OKEx.csv', names=col_names)
  data = data.iloc[1:]
  return data


show_data = st.selectbox("Show Sample Data", ["Please select", "Binance",
"Bittrex", "Huobi", "Kucoin", "OKEx"])
if show_data == "Binance":
  df1 = load_binance()
  st.subheader('Binance data')
  st.write(df1.head(10))
elif show_data == "Bittrex":
  df2 = load_bittrex()
  df2 = df2.drop('reply_to_message_id', 1)
  st.subheader('Bittrex data')
  st.write(df2.head(10))
if show_data == "Huobi":
  df3 = load_huobi()
  st.subheader('Huobi data')
  st.write(df3.head(10))
if show_data == "Kucoin":
  df4 = load_kucoin()
  st.subheader('Kucoin data')
  st.write(df4.head(10))
if show_data == "OKEx":
  df5 = load_OKEx()
  df5 = df5.drop('reply_to_message_id', 1)
  st.subheader('OKEx data')
  st.write(df5.head(10))


trend = st.selectbox("Telegram Discussion Trends", ["Please select", "Bitcoin
vs Ether", "DOT vs XRP vs LTC vs XLM",
  "Bitcoin vs DeFi", "Centralized stablecoins (USDT) vs decentralized (DAI)",
  "Decentralized exchanges (UNISWAP vs Sushiswap vs 1inch)", "DeFi protocols
(MakerDAO vs Compound)"])

@st.cache
def load_crypto():
  df = pd.read_csv('crypto.csv')
  return df

crypto = load_crypto()

if trend == "Bitcoin vs Ether":
  fig = px.scatter(crypto,
               x='date',
               y=['btc','eth'],
               hover_name='date',
               title="Telegram Discussions: BTC vs ETH")
  st.plotly_chart(fig)
elif trend == "DOT vs XRP vs LTC vs XLM":
  fig = px.scatter(crypto,
               x='date',
               y=['dot','xrp','ltc','xlm'],
               hover_name='date',
               title="Telegram Discussions: DOT vs XRP vs LTC vs XLM")
  st.plotly_chart(fig)
elif trend == "Bitcoin vs DeFi":
  fig = px.scatter(crypto,
               x='date',
               y=['btc','defi'],
               hover_name='date',
               title="Telegram Discussions: BTC vs DeFi")
  st.plotly_chart(fig)
elif trend == "Centralized stablecoins (USDT) vs decentralized (DAI)":
  fig = px.scatter(crypto,
               x='date',
               y=['usdt','dai'],
               hover_name='date',
               title="Stable coins: USDT vs DAI")
  st.plotly_chart(fig)
elif trend == "Decentralized exchanges (UNISWAP vs Sushiswap vs 1inch)":
  fig = px.scatter(crypto,
               x='date',
               y=['uniswap','sushiswap','1inch'],
               hover_name='date',
               title="DEX: UNISWAP vs Sushiswap vs 1inch")
  st.plotly_chart(fig)
elif trend == "DeFi protocols (MakerDAO vs Compound)":
  fig = px.scatter(crypto,
               x='date',
               y=['makerdao','compound'],
               hover_name='date',
               title="DeFi protocols (MakerDAO vs Compound)")
  st.plotly_chart(fig)

### end of riyad's code ###

### araib's analysis ###
st.subheader('More Analyses from 1/28/21 to 3/23/21')
'''
[Link]
(https://github.com/PoliNemkova/Telegram_analysis/tree/streamlit/images_from_
others) to source of the data analyses.
'''

st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/araib/3mo_BTCvsDeFi.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/araib/3mo_BTCvsETH.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/araib/3mo_DEX.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/araib/3mo_DOT_XRP_LTC_XLM.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/araib/3mo_MakerDaovsCompound.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/araib/3mo_USDTvsDAI.PNG")

### poli's analysis ###
```

```
st.subheader('Prices vs Mentions in Telegram, 1/28/21 to 3/23/21')

st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/poli/btc.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/poli/etc.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/poli/ltc.PNG")

### wesley's analysis ###
st.subheader('Crypto Sentiment Analysis, 1/28/21 to 3/23/21')

st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/wesley/SA_1inch.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/wesley/SA_btc.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/wesley/SA_compound.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/wesley/SA_dai.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/wesley/SA_defi.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/wesley/SA_makerdao.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/wesley/SA_sushiswap.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/wesley/SA_uniswap.PNG")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/wesley/SA_usdt.PNG")

### previous 3 years ###
st.header('Past 3 Years')
st.subheader('Data Analyses from 2018 till Jan 2021')

st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/3_yrs/3yrs_btc_defi.png")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/3_yrs/3yrs_btc_eth.png")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/3_yrs/3yrs_dexs.png")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/3_yrs/3yrs_dot_xrp_ltc_xlm.png")
st.image("https://raw.githubusercontent.com/PoliNemkova/Telegram_analysis/str
eamlit/images_from_others/3_yrs/3yrs_usdt_dai.png")
```