databricksAsset Ticker vs Tweets

Install

pip install yfinance

Python interpreter will be restarted.

Requirement already satisfied: yfinance in /local_disk0/.ephemeral_nfs/envs/pythonEnv-bf2cd4ea-30fb-4c97-ae67 -3a7c1ed5872e/lib/python3.7/site-packages (0.1.59)

Requirement already satisfied: multitasking>=0.0.7 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-bf2cd4ea-30fb-4c97-ae67-3a7c1ed5872e/lib/python3.7/site-packages (from yfinance) (0.0.9)

Requirement already satisfied: pandas>=0.24 in /databricks/python3/lib/python3.7/site-packages (from yfinanc e) (1.0.1)

Requirement already satisfied: requests>=2.20 in /databricks/python3/lib/python3.7/site-packages (from yfinan ce) (2.22.0)

Requirement already satisfied: numpy>=1.15 in /databricks/python3/lib/python3.7/site-packages (from yfinance) (1.18.1)

Requirement already satisfied: lxml>=4.5.1 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-bf2cd4ea-30fb-4c97-a e67-3a7c1ed5872e/lib/python3.7/site-packages (from yfinance) (4.6.3)

Requirement already satisfied: python-dateutil>=2.6.1 in /databricks/python3/lib/python3.7/site-packages (fro m pandas>=0.24->yfinance) (2.8.1)

Requirement already satisfied: pytz>=2017.2 in /databricks/python3/lib/python3.7/site-packages (from pandas>= 0.24->yfinance) (2019.3)

Requirement already satisfied: six>=1.5 in /databricks/python3/lib/python3.7/site-packages (from python-dateu til>=2.6.1->pandas>=0.24->yfinance) (1.14.0)

Requirement already satisfied: certifi>=2017.4.17 in /databricks/python3/lib/python3.7/site-packages (from requests>=2.20->yfinance) (2020.6.20)

Requirement already satisfied: idna<2.9,>=2.5 in /databricks/python3/lib/python3.7/site-packages (from reques ts>=2.20->yfinance) (2.8)

Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /databricks/python3/lib/python3.7/s ite-packages (from requests>=2.20->yfinance) (1.25.8)

Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/lib/python3/dist-packages (from requests>=2.20-> yfinance) (3.0.4)

Python interpreter will be restarted.

Import libraries

```
from datetime import datetime, timedelta
import numpy as np
import pandas as pd
import yfinance
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import matplotlib.dates as mdates
import textwrap
```

Set initial params Ticker / search criterias

```
#ticker_tweet_data parameters
TICKER_TWEET_PARAM_DF = pd.DataFrame(
    columns =
    ["ticker", "case_insensitive_keywords", "case_sensitive_keywords"],
data = Γ
    #["BTC-USD", "bitcoin, $btc, satoshi, nakamoto, crypto, altcoin", "BTC"],
    ["BTC-USD", "bitcoin, $btc, satoshi, nakamoto", "BTC"],
    #["ETH-USD", "ethereum, $eth, vitalik, buterin, crypto, altcoin", "ETH, Eth, Ether"],
    ["ETH-USD", "ethereum", "ETH"],
    #["DOGE-USD", "dogecoin,$doge,billy,markus,crypto,altcoin", "DOGE,Doge"]
    ["DOGE-USD", "dogecoin,$doge,dogefather", "DOGE,Doge"],
    ["LTC-USD", "litecoin, $ltc", "LTC"],
    ["XRP-USD", "ripple,$XRP", "XRP"],
    ["ADA-USD", "cardano,$ADA", "ADA"]
1)
TICKER_TWEET_PARAM_DF = TICKER_TWEET_PARAM_DF.set_index("ticker")
#Drawing board
CANVAS\_SIZE = (30, 12)
```

Function to filter tweets by keyword and dates

```
def get_filtered_tweets_by_ticker(
       ticker_symbol,
       ticker_tweet_param_df,
       tweets,
       min_tweet_date = None,
       max_tweet_date = None):
    row_params = ticker_tweet_param_df.loc[ticker_symbol]
    case_insensitive_keywords = row_params["case_insensitive_keywords"].split(",")
    # apply case insensitive search - compare .uppercase versions
    case_insensitive_filtered_tweet_index = tweets["text"].apply(lambda sentence: any(word.upper() in
sentence.upper() for word in case_insensitive_keywords))
    case_insensitive_filtered_tweets = tweets[case_insensitive_filtered_tweet_index]
    case_sensitive_keywords = row_params["case_sensitive_keywords"].split(",")
    # apply case sensitive search
    case_sensitive_filtered_tweet_index = tweets["text"].apply(lambda sentence: any(word in sentence for word
in case_sensitive_keywords))
    case_sensitive_filtered_tweets = tweets[case_sensitive_filtered_tweet_index]
    return_value = case_sensitive_filtered_tweets.append(case_insensitive_filtered_tweets)
    return_value = return_value.drop_duplicates(subset="id")
    if min_tweet_date is not None:
        return_value_index = [date.strftime("%Y-%m-%d") >= min_tweet_date for date in return_value["date"]]
        return_value = return_value[return_value_index]
    if min_tweet_date is not None:
        return_value_index = [date.strftime("%Y-%m-%d") <= max_tweet_date for date in return_value["date"]]
        return_value = return_value[return_value_index]
    return return_value
```

Open tweets available from MongoDB

This collection may hold several usernames, we're retrieving only the tweets from the user we're plotting

cluster, edit, spark config

spark.mongodb.input.uri mongodb+srv://admin:pYVSN3RcDS8huA6@cluster0.5ryg7.mongodb.net/

spark.mongodb.output.uri mongodb+srv://admin:pYVSN3RcDS8huA6@cluster0.5ryg7.mongodb.net/

```
# This Mongo DB Database has 131K tweets
#displayHTML("<img src ='dbfs:/FileStore/tables/MongoDBscreenshot.PNG'>")
#displayHTML("<img src ='/FileStore/tables/MongoDBscreenshot.PNG/'>")
displayHTML("<img src ='FileStore/tables/MongoDBscreenshot.PNG/'>")
```



```
def get_tweets_from_mongo(twitter_user):
    tweets_df = spark.read.format("com.mongodb.spark.sql.DefaultSource").option("database",
"twitter").option("collection", "user_tweets").load()
    tweets_df = tweets_df.filter(tweets_df["username"] == twitter_user)
    tweets_df = tweets_df.toPandas()
    tweets_df['date'] = tweets_df['date'].astype('datetime64[ns]')
    return tweets_df

#TWEETS = spark.read.format("com.mongodb.spark.sql.DefaultSource").option("database",
"twitter").option("collection", "elonmusk_01").load()
#TWEETS = spark.read.format("com.mongodb.spark.sql.DefaultSource").option("database",
"twitter").option("collection", "user_tweets").load()
#TWEETS = TWEETS.filter(TWEETS["username"] == TWITTER_USER)
#TWEETS = TWEETS.toPandas()
#TWEETS = TWEETS.toPandas()
#TWEETS['date'] = TWEETS['date'].astype('datetime64[ns]')
```

Get filtered version of tweets and prices according to selected ticker

This is very inneficient needs some major tuning

This section loads objects

filtered_tweets price history

```
def get_filtered_tweets_and_prices(
   ticker_symbol,
   ticker_tweet_param_df,
   tweets,
   min_tweet_date = None,
   max_tweet_date = None):
    #load the tweet DF
    return_value_filtered_tweets = get_filtered_tweets_by_ticker(ticker_symbol, ticker_tweet_param_df, tweets,
min_tweet_date, max_tweet_date)
    #filtered_tweets.head(3)
    #min(filtered_tweets["date"])
    # Get 1 week prior
   ticker_price_start_date = min(return_value_filtered_tweets["date"])
   ticker_price_start_date = ticker_price_start_date - timedelta(days = ticker_price_start_date.weekday())
    # Get 1 week after
   ticker_price_end_date = max(return_value_filtered_tweets["date"])
   ticker_price_end_date = ticker_price_end_date + timedelta(days = ticker_price_end_date.weekday())
    #print("beg {0} end{1}".format(ticker_price_start_date, ticker_price_end_date))
    # Now to open price records
    price_history = yfinance.download(ticker_symbol, start=ticker_price_start_date, end=ticker_price_end_date)
    #display(price_history.head(2))
    # Remove Tweets, if there is no ticker price data
   filtered_tweets_index = return_value_filtered_tweets["date"].apply(lambda tweet_date:
tweet_date.strftime("%Y-%m-%d") in [price_dates.strftime("%Y-%m-%d") for price_dates in price_history.index])
    return_value_filtered_tweets = return_value_filtered_tweets[filtered_tweets_index]
    # re fetch with new filtered tweet data
    # Get 1 week prior
   ticker_price_start_date = min(return_value_filtered_tweets["date"])
   ticker_price_start_date = ticker_price_start_date - timedelta(days = ticker_price_start_date.weekday())
```

```
# Get 1 week after
ticker_price_end_date = max(return_value_filtered_tweets["date"])
ticker_price_end_date = ticker_price_end_date + timedelta(days = ticker_price_end_date.weekday())
#print("beg {0} end{1}".format(ticker_price_start_date, ticker_price_end_date))
# To-DO
# Just filter this at the dataset instead of refetching it from the ticker data source
price_history = yfinance.download(ticker_symbol, start=ticker_price_start_date, end=ticker_price_end_date)
return return_value_filtered_tweets, price_history
```

Get Function to draw ticker prices vs tweets

```
def render_visual(
    # Dataframe with ticker prices
   ticker_prices_df,
    #ticker symbol
   ticker_symbol,
   #Ticker column. Usually "Close" as closing price
    # is the one most commonly rendered in daily charts
   ticker_column_name_to_plot,
    # Dataframe with relevant tweets to the ticker
   relevant_tweets_df,
    # twitter username
   tweets_username
   ):
    # Tweet information to draw
    # as prices are given by day, we need to round to the day the twitter post date
    tweets_date = [date.date() for date in relevant_tweets_df["date"]]
    prices_at_tweet = list(ticker_prices_df.loc[date.strftime("%Y-%m-%d")][ticker_column_name_to_plot] for
date in tweets_date)
    tweets_text = relevant_tweets_df["text"]
    #Initialize canvas
   fig, ax = plt.subplots(figsize = CANVAS_SIZE)
    #Draw line of prices
    ax.plot(ticker_prices_df.index, ticker_prices_df[ticker_column_name_to_plot])
    #draw tweet vertical stems
    # I'm sure there's a smarter way to do this: get range and leave space to draw the vertical stem
   min_price = min(ticker_prices_df[ticker_column_name_to_plot])
    range_price = max(ticker_prices_df[ticker_column_name_to_plot]) - min_price
    # The vertical stems.
```

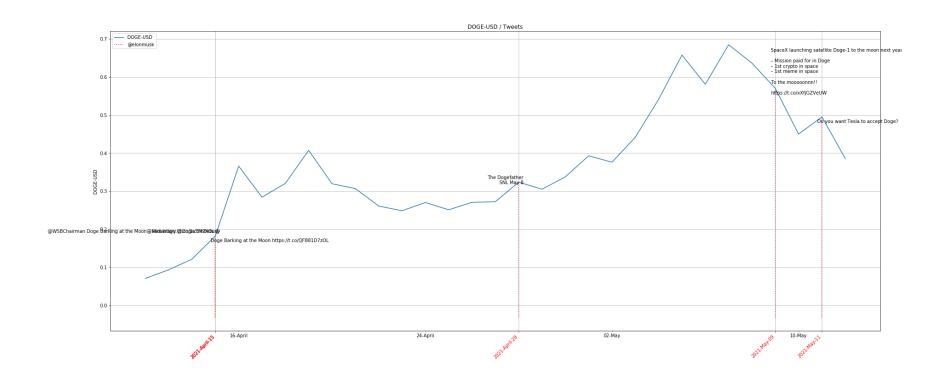
```
#ax.vlines(tweets_date, min_price - range_price/4, prices_at_tweet, color="tab:red", linestyles =
"dotted")
   ax.vlines(tweets_date, min_price - range_price/6, prices_at_tweet, color="tab:red", linestyles = "dotted")
   # annotate lines
   alternate_position = 1
   for date, price_level, tweet_text in zip(tweets_date, prices_at_tweet, tweets_text):
     #tweet_text = textwrap.shorten(tweet_text, 20)
     ax.annotate(tweet_text, xy = (date, price_level), xytext = (10 * alternate_position, 15 *
alternate_position), textcoords="offset points", horizontalalignment = "right" if alternate_position>0 else
alternate_position = alternate_position * -1
   print(2)
   #Set Major locator tick
   ax.xaxis.set_major_locator(ticker.FixedLocator([mdates.date2num(z) for z in tweets_date] ))
   ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%B-%d'))
   ax.tick_params(axis="x",colors='red', which='major')
   #Set Minor locator tick
   ax.xaxis.set_minor_locator(ticker.MaxNLocator(5))
   ax.xaxis.set_minor_formatter(mdates.DateFormatter('%d-%B'))
   labels = ax.get_xticklabels()
   plt.setp(labels, rotation=45, horizontalalignment='right')
   #legend
   ax.set_ylabel(ticker_symbol)
   ax.set_title("{0} / {1}".format(ticker_symbol, "Tweets"))
   ax.legend([ticker_symbol, "@{0}".format(tweets_username)], loc='upper left')
   ax.grid()
   plt.show()
```

plt.clf()

Draw

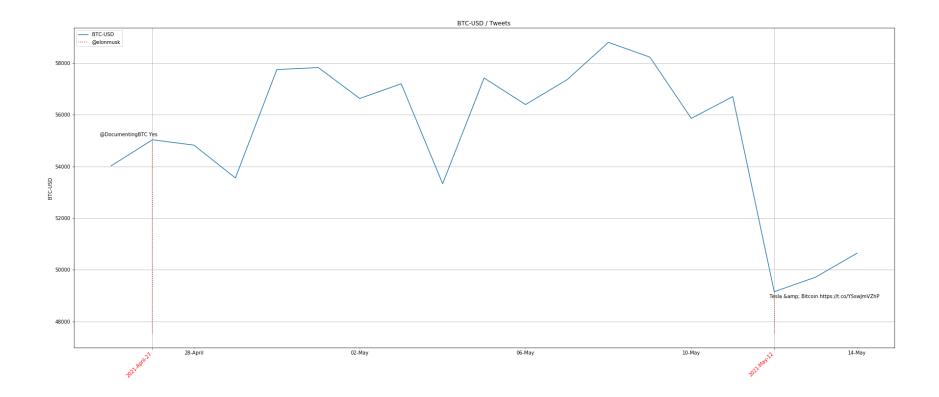
DOGE vs Elon Musk

```
# Selected ticker symbol
SELECTED_TICKER_SYMBOL = "DOGE-USD"
#TWITTER HANDLER
TWITTER_USER = "elonmusk"
#Date parameters to override
OVERRIDE_MIN_DATE = "2021-04-15"
OVERRIDE_MAX_DATE = "2021-05-12"
filtered_tweets, price_history = get_filtered_tweets_and_prices(
    SELECTED_TICKER_SYMBOL,
   TICKER_TWEET_PARAM_DF,
   get_tweets_from_mongo(TWITTER_USER),
   OVERRIDE_MIN_DATE,
   OVERRIDE_MAX_DATE)
render_visual(
    # Dataframe with ticker prices
    price_history,
    #ticker symbol
   SELECTED_TICKER_SYMBOL,
    #Ticker column. Usually "Close" as closing price
    # is the one most commonly rendered in daily charts
    "Close",
    # Dataframe with relevant tweets to the ticker
   filtered_tweets,
    # twitter username
   TWITTER_USER
```



BTC vs Elon Musk

```
# Selected ticker symbol
SELECTED_TICKER_SYMBOL = "BTC-USD"
#TWITTER HANDLER
TWITTER_USER = "elonmusk"
#Date parameters to override
OVERRIDE_MIN_DATE = "2021-04-15"
OVERRIDE_MAX_DATE = "2021-05-15"
filtered_tweets, price_history = get_filtered_tweets_and_prices(
    SELECTED_TICKER_SYMBOL,
   TICKER_TWEET_PARAM_DF,
   get_tweets_from_mongo(TWITTER_USER),
   OVERRIDE_MIN_DATE,
   OVERRIDE_MAX_DATE)
render_visual(
    # Dataframe with ticker prices
   price_history,
    #ticker symbol
   SELECTED_TICKER_SYMBOL,
    #Ticker column. Usually "Close" as closing price
    # is the one most commonly rendered in daily charts
    "Close",
    # Dataframe with relevant tweets to the ticker
   filtered_tweets,
    # twitter username
   TWITTER_USER
    )
```



ETH vs Vitalik Buterin

