YZV211E - Introduction to Data Science and Engineering

Emircan EROL, 150200324, Istanbul Technical University

Bitcoin Data

I could not manage to use an API because historical data is not free and after research I found a website and downloaded historical data(I did not want to use ready data). Also I wrote code for Cryto Compare API.

Tweet Data

My TwitterAPI application is under review, twint is used in this project. In order to initialize twint configurations are declared and because of twitter a problem in twint which is global and recent. Code automatically detects last date and goes on. At the end file is saved as space separated values.

Code Snippet 1: get tweets

```
# allows nested use of asyncio.run
nest_asyncio.apply()

c = twint.Config()
c.Username = 'elonmusk'
c.Since = '2019-09-08 17:57:00'

with open('elonMuskTweets.csv') as f:
    for line in f:
        pass
    last_line = line
date = last_line[20:39]

c.Until = date
c.Output_pandas = True
c.Output = 'elonMuskTweets.csv'
c.Show_cashtags = True
c.Show_hashtags = True

# twint is searching
twint.run.Search(c)
```

Beautify Tweets

Because tweets have space characters and file is also space separated, division is a must(5th and 6th rows). Timezone is changed from UTC+03:00 to UTC±00:00. Unix time is created and data frame is returned.

Code Snippet 2: edit tweet

```
with open('elonMuskTweets.csv', 'r') as f:
    titles = f.readline().strip().split(' ')
    df = pd.DataFrame(columns=["unix", "time", "tweet"])
    for line in f:
        infos = line[:56].split(' ')
        tweet = line[56:].strip()

        date = infos[1]
        hour = infos[2]

        three_hours = datetime.timedelta(hours=3)
        time_obj = datetime.datetime.strptime(date + " " + hour, '%Y-%↩
            m-%d %H:%M:%S') - three_hours

        # unix timestamp is created
        unix = time.mktime(time_obj.timetuple())

        # new rows are added to df
        df = df.append({'unix': unix, 'time': time_obj, 'tweet': tweet↩
            }, ignore_index=True)
```

Control Missed Value

If values are not ordered (missing values) this function warns and prints first distorted unix timestamp. It does not effect anything just control if there is some missing value.

Code Snippet 3: edges

```
def controlMissedValue(btc:pd.DataFrame):
    i = int(btc['unix'][0] / 60)
    for x in btc['unix']:
        x = int(x/60)
        if i != x:
            print(x*60)
            break
        i -= 1
```

Fill Empty Data

Code Snippet 4: fill empyt data

```
    with open('BTCUSDnew.csv', 'r') as f:
        # file created to write
        not_empty = open('BTCUSD_min.csv', 'w')
        titles = f.readline().strip().split(',')
        prev_line = f.readline().strip().split(',')
        not_empty.write(",".join(titles[:-1]) + '\n' + ",".join(↩
            prev_line) + '\n')
        prev_unix = prev_line[0]
        for line in f:
            line = line.strip().split(',')
            unix = line[0]
            if int(unix) == (int(prev_unix) - 60):
                prev_line = line
                prev_unix = unix
                not_empty.write(",".join(line) + '\n')
                continue

            space_len = int((int(prev_unix) - int(unix)) / 60) - 1
            for i in range(space_len, 0, -1):
                new_unix = int(unix) + (60 * i)
                new_line = prev_line
                new_line[0] = str(new_unix)
                not_empty.write(",".join(new_line) + '\n')
            prev_unix = unix
            prev_unix = unix
            not_empty.write(",".join(line) + '\n')

    not_empty.close()
```

Join Tweet and Prices

My choice is to reach price values by their indices. Location on prices data frame created based on unix data and index is calculated. Hourly, day start, day end unix timestamps are also derived indices and unix timestamp.

Code Snippet 5: join tweet prices

```
cols = ['unix', 'date', 'tweet', 'day start', 'tweet time price', 'ten↩
    minute', 'hour', 'day end']
    df = pd.DataFrame(columns=cols) # create a dataframe to save data
```

```
    for line in tw.iloc: # every line in tweets DataFrame .iloc is

        temp = line['unix'] # to save time value because t will change
        t=line['unix'] # better for calculations
        t = t-t%60 # round to minute
        date = line[1] # better to represent as date
        tweet = line['tweet']

        int_location = int((btc['unix'][0] - t) / 60) # calculate ↩
            where the value is
        ds = t - t%86400 # day start

        #prices
        nw = btc['unix'][int_location] # get open price for tweet time
        nw_open = btc['open'][int_location]
        ten_minute = btc['open'][int_location - 10] # int_loc + 10 ↩
            minutes
        hour = btc['open'][int_location - 60]
        day_loc = int((btc['unix'][0]-ds)/60)
        day_start = btc['open'][day_loc]
        day_end = btc['open'][day_loc - 1440] # day_loc - 1440: go to ↩
            1 day later

        # create a temporary dataframe and append to df
        final_line = pd.DataFrame({'unix':[temp],
                                   'date':[date],
                                   'tweet':[tweet],
                                   'day start':[day_start],
                                   'tweet time price':[nw_open],
                                   'ten minute':[ten_minute],
                                   'hour':[hour],
                                   'day end':[day_end]})

        df = df.append(final_line, ignore_index=True)
```