Data preparation

In this blog post, we will introduce how we prepare the financial and social data using web scrapers and APIs.

Snscrape

Snscrape is a Python package that allows you to scrape data from various social media platforms such as Twitter, Instagram, Reddit, and more. It is an alternative to the official APIs provided by these platforms, and it uses web scraping techniques to extract data directly from the HTML source code of the websites.

One advantage of using snscrape is that it can help prevent rate limit errors, which occur when you make too many requests to an API within a certain period of time. Many social media platforms have rate limits in place to prevent abuse and ensure the stability of their services.

Snscrape can prevent rate limit errors by using a combination of features such as rate limiting, backoff strategies, and proxies. Rate limiting involves slowing down the rate at which you make requests to an API to stay within the platform's limits. Backoff strategies involve waiting for a certain period of time before making another request after receiving a rate limit error. Proxies involve using a different IP address for each request to avoid triggering rate limit restrictions.

Overall, snscrape can be a useful tool for scraping data from social media platforms while avoiding rate limit errors.

Extract data from Twitter

To scrape data from Twitter using snscrape, we first define the cashtag (symbol for a stock or cryptocurrency) we want to search for. Then, we use the TwitterSearchScraper method from snscrape to scrape tweets that contain the cashtag. After removing special characters, we store the scraped tweets in a pandas DataFrame and export it to a csv file.

```
import snscrape.modules.twitter as sntwitter

# Creating list to append tweet data to
tweets_list = []

# Using TwitterSearchScraper to scrape data and append tweets to list
for s in stocks_symbol:
    for i,tweet in enumerate(sntwitter.TwitterSearchScraper('${} (from:jimcramer)
since:1990-01-01 until:2023-12-31'.format(s)).get_items()):
        content = tweet.content.replace('\n', '')
        content = content.replace('\r', '')
        tweets_list.append([tweet.date, s, content, tweet.retweetedTweet])
    break

# Creating a dataframe from the tweets list above
tweets_df = pd.DataFrame(tweets_list, columns=['Datetime', 'Symbol', 'Text',
'Retweeted'])
```

YFinance

Yfinance is a Python package that provides a simple and convenient way to download historical stock price data from Yahoo Finance. It allows users to easily retrieve historical stock data and financial statements, as well as real-time stock price data.

One of the advantages of yfinance is its comprehensive data coverage, which makes it a valuable tool for investors and researchers who need to analyze the performance of different financial markets and asset classes. With yfinance, users can easily download historical stock data for individual companies or for entire stock market indices, such as the S&P 500 or Dow Jones Industrial Average.

Moreover, yfinance uses multithreading to download data in parallel, which can significantly improve performance. This allows users to download large amounts of historical stock data quickly and efficiently. Additionally, yfinance's ability to download data in parallel makes it possible to retrieve data for multiple stocks or assets at once, further improving performance.

Financial data extraction

To extract the corresponding financial data from yfinance, we used the ticker.history method to trace back the history of a given stock symbol by passing in the start and end dates as parameters. For example, `yfinance.Ticker("MSFT").history(start="2020-01-01", end="2022-01-31")` returns us the historical data of Microsoft in the whole January in 2020, including daily trading volume and closing prices. In particular, we extract the closing prices of the stocks on the 0th day (the day when Jim Cramer post the tweet), 30th day, 90th day and 180th day.

Difficulties

Sometimes that the target day may be a non-trading day (public hollidays or weekends). If no data is available on that day, the API call will return an error `No data found, symbol may be delisted`. To solve this problem, we extended the request date by 1 day until the data is successfully fetched.

```
from datetime import datetime, timedelta, date
import numpy as np
import yfinance as yf

# get date string after n days
def addDate(date, day):
    date1 = datetime.strptime(date, '%Y-%m-%d').date()
    date2 = date1 + timedelta(days=day)
    return str(date2)

# get stock price given the date
def yfSearch(row, days):
    today = date.today()
    symbol = 'BRK-B' if row['Symbol'] == 'BRK.B' else row['Symbol']
    if stocks_status[stocks_status['symbol'] == symbol]['ipoDate'].values[0] >=
row['Datetime'][:10]:
        return np.nan
    ticker = yf.Tickers(symbol)
    adj = 1
```

```
    while True:
        # return nan if the request date is later than today
        if addDate(row['Datetime'][:10], days+adj) >= today.strftime("%Y-%m-%d"):
            return np.nan
        try:
            return ticker.tickers[symbol].history(start=addDate(row['Datetime']
[:10], days),

end=addDate(row['Datetime'][:10], days+adj),
                                                        raise_errors=True)
['Close'].values[0]
        # extend 1 request date if no data on that day
        except:
            adj += 1
```

Conclusion

Through this journet, we have learned the importance of selecting the right tools for the task at hand. We found snscrape to be a useful tool for scraping data from social media platforms, as it can help prevent rate limit errors and avoid triggering restrictions. Additionally, we found yfinance to be a convenient and efficient way to download historical stock price data, with its comprehensive data coverage and ability to download data in parallel.

Through this experience, we have also encountered some difficulties, such as non-trading days and the need to extend the request date to fetch data successfully. However, we have found that these challenges can be overcome with careful planning and attention to detail.