Python Code Documentation

Version of python:

```
>> Python 3.6.8
```

Version of scrapy

```
>> Scrapy 2.5.0
```

Command line to download scrapy

```
>> sudo pip3 install scrapy
```

Version of chromedriver

```
>> ChromeDriver 91.0.4472.101
```

the download link is: https://chromedriver.chromium.org/downloads)

Some potential error might encounter while calling webdriver in python:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/selenium/webdriver/chrome/webdriver.py", line 73, in __init__
    self.service.start()
  File "/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/selenium/webdriver/common/service.py", line 104, in start
    raise WebDriverException("Can not connect to the Service %s" % self.path)
selenium.common.exceptions.WebDriverException: Message: Can not connect to the Service <the directory of chromedriver>  (i.e./Users/xuhuili/Desktop/ccm_project/chromedriver)
```

Try type the following line in terminal:

```
>> xattr -d com.apple.quarantine <the directory of chromedriver>
(i.e./Users/xuhuili/Desktop/ccm_project/chromedriver)
```

To obtain the csv files of each pivot carbon market, we firstly need to go to the directory of each carbon market. Let's take Beijing carbon market as an example.

In the terminal, type the following command lines:

```
>> cd /Users/xuhuili/Desktop/ccm_project/web_scraping/beijing_spider/beijing_spider/spiders
```

The actual part we need to change is the text that we highlighted in red. Other than beijing_spider, we also have tianjin_spider, guangdong_spider, hubei_spider, shenzhen_spider, chongqing_spider, fujian_spider, shanghai_spider, which indicates different pivot carbon market.

To scrape the data from the website, we then type the following command

```
>> scrapy crawl beijing -o beijing.csv
```

It should be noted that the command lines are different for each carbon market, they are listed below:

Beijing

```
>> scrapy crawl beijing -o beijing.csv
```

Tianjin

```
>> scrapy crawl tianjin -o tianjin.csv
```

Guangdong

```
>> scrapy crawl guangdong
```

Hubei

```
>> scrapy crawl hubei -o hubei.csv
```

Chongqing

```
>> scrapy crawl chongqing
```

Fujian

```
>> scrapy crawl fujian
```

Shenzhen

```
>> scrapy crawl shenzhen -o shenzhen.csv
```

Shanghai

```
>> scrapy crawl shanghai
```

We then obtain the raw data table for each carbon market. The next step is to clean the data, let's assume that we are still in the ./spiders directory. We type the following command to clean the data:

```
>> scrapy clean/clean.py
```

In this step, we will obtain the total return time series for each carbon market, labelled as ‘Return’. We will also obtain features for each carbon market such as trading index, settlement price, settlement volume and turnover, etc. The output CSV files of this step would be a string such as ‘beijing_BEA.csv’ (‘city name’ + ‘_’ + ‘trading index + ‘.csv’)

After cleaning the table for each carbon market, we will then obtain metrics for each carbon markets such as the annualised rates of return and standard deviation, and correlation of each carbon market to the main equity and bond indices over multiple periods we specify.

Take beijing as an example, we type the following command line in terminal (assuming that now we are in the ./web_scraping directory):

```
>> python3.6
>> from stats.stats import get_stats
>> beijing = Stats(csv_file =
['/Users/xuhuili/Desktop/ccm_project/web_scraping/beijing_spider/beijing_spider/spiders/beijin
g_BEA.csv','/Users/xuhuili/Desktop/ccm_project/Comparator_indices_07062021.csv'],
trading_index = 'beijing', risk_free
= 0.02388, start_date = '2016-01-01', end_date = '2020-12-31', to_csv = True)
>> beijing.get_stats()
```

Note:

1. This step could also be done in a Jupyter Notebook, where some examples are provided.
2. Comparator_indices_07062021.csv is cleaned and converted from an excel file to a csv file.

We have five parameters in Stats class, they are:

- csv_file: [carbon market csv, comparator csv]
- trading_index: name of trading index
- risk_free: risk free rate (The China 1 Year Government Bond yield)
- start_date: it depends on the length of timeframe for each carbon market csv, we can have ‘2016-01-01’
- end_date: it depends on the length of timeframe for each carbon market csv, we can have ‘2020-12-31’
- to_csv: whether convert df, aggregated_df, correlation_df to CSV files

This function will provide the following metrics:

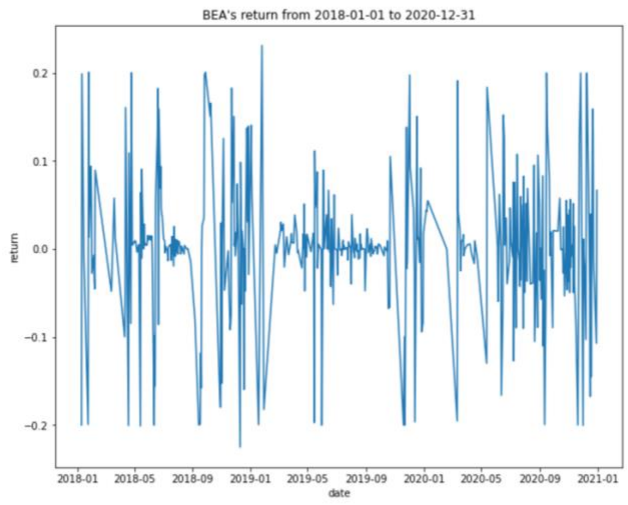
- Total return
- Annualised return over months
- Annualised return over years
- Standard deviation
- Annualised volatility (Annualised standard deviation)
- Sharpe Ratio
- Skewness
- Kurtosis
- Covariance matrix of each carbon market to the main equity and bond indices.

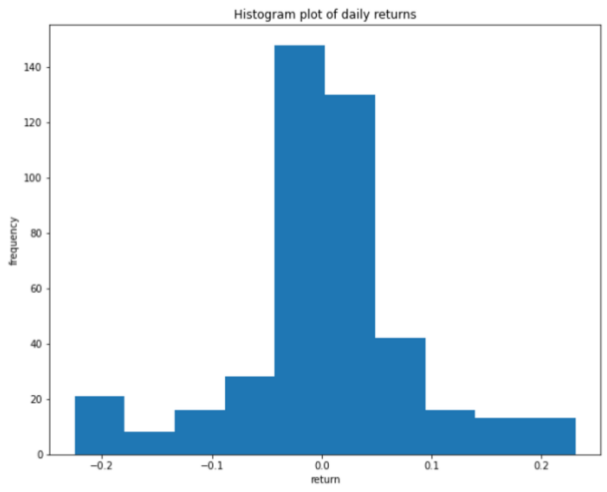
It will also produce the following graphs:

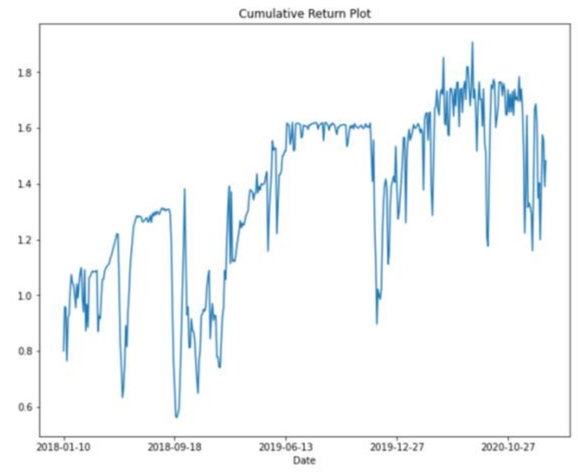
- Return plot over time
- Histogram plot of daily returns
- Cumulative return plot
- Maximum daily drawdown (window length = 180 days)

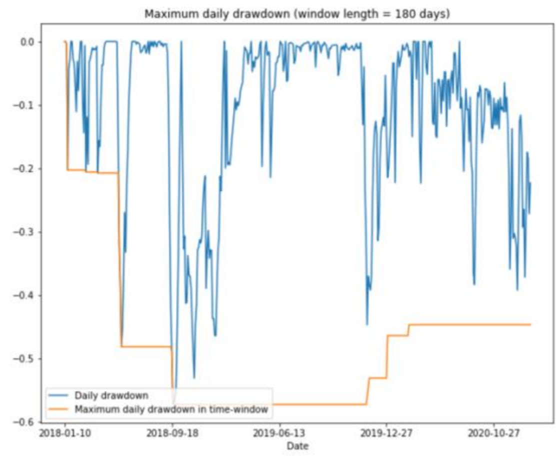
The output of the get_stats function in Jupyter Notebook
(Beijing BEA)

```
BEA carbon market metrics from 2018-01-01 to 2020-12-31
>> Total Return:  85.1852%
>> Annualised return over months:  22.801%
>> Annualised return over years:  36.0828%
>> Standard Deviation:  0.0786
>> Annualised volatility:  1.2432
>> Sharpe ratio:  0.271
>> Skewnesss:  -0.2239
>> Kurtosis:  1.6164
>> Maximum drawdown:  -57.24%
```









After calling the get_stats function, the aggregated_df, correlation_df and result_df will be stored in stats_output folder as CSV file if defining to_csv as True.

We can also obtain the correlation matrices of Chinese carbon markets internally. To do that, type the following command lines in terminal (or Jupyter Notebook):

```
>> python3.6
>> from stats.stats import get_stats
>> Correlation(csv_files = ['stats_output/BEA_aggregated_df_2018-01-01_2020-12-31.csv',\
                          'stats_output/CQEA_aggregated_df_2018-01-01_2020-12-31.csv', \
                         'stats_output/FJEA_aggregated_df_2018-01-01_2020-12-31.csv', \
                         'stats_output/GDEA_aggregated_df_2018-01-01_2020-12-31.csv', \
                         'stats_output/HBEA_aggregated_df_2018-01-01_2020-12-31.csv', \
                         'stats_output/SHEA_aggregated_df_2018-01-01_2020-12-31.csv', \
                         'stats_output/SZA_2015_aggregated_df_2018-01-01_2020-12-31.csv', \
                         'stats_output/TJEA_aggregated_df_2018-01-01_2020-12-31.csv'],
start_date = '2018-01-01', end_date = '2020-12-31', to_csv = True)
>> cm_corr_2018_01_01_2020_12_31.get_correlation()
```

We have four parameters in Correlation class, they are:

- csv_file: [carbon market csv1, carbon market csv2, …, carbon market csv8], a list of 8 Chinese carbon market csv files in the following order: ‘Beijing’, ‘Chongqing’, ‘Fujian’, ‘Guangdong’, ‘Hubei’, ‘Shanghai’, Shenzhen’, ‘Tianjin’
- start_date: it depends on the length of timeframe for each carbon market csv, we can have ‘2016-01-01’
- end_date: it depends on the length of timeframe for each carbon market csv, we can have ‘2020-12-31’
- to_csv: whether convert correlation matrix into CSV file

Assuming that we are in the web_scraping directory, we can merge all dataframes of Chinese carbon markets along with the comparator dataframe into a single dataframe in a long format by typing the following commands in terminal:

```
>> python3.6
>> from merge_df.merge_df import merge_df
>> merge_df(csv_files =  csv_files, comparator_csv =
'/Users/xuhuili/Desktop/ccm_project/Comparator_indices_07062021.csv',
 to_csv = True)
```

We have four parameters in merge_df function, they are:

- csv_file: [carbon market csv1, carbon market csv2, …, carbon market csv8], a list of 8 Chinese carbon market csv -> list
- comparator_csv: comparator csv file
- to_csv: whether convert the merged df into CSV file