Python Program - KLSE Listed Stocks Scraper

Imran Hakim Roslan
imrh@protonmail.com

January 30, 2021

1 Introduction

A Python application to scrape KLSE listed stocks data from web pages. The program will scrape for stock data everyday at time GMT+8 1815 every weekday and the collected data will then be stored in a csv file for reference and monitoring purposes.

2 Goals

The goal for the application are being discussed below:

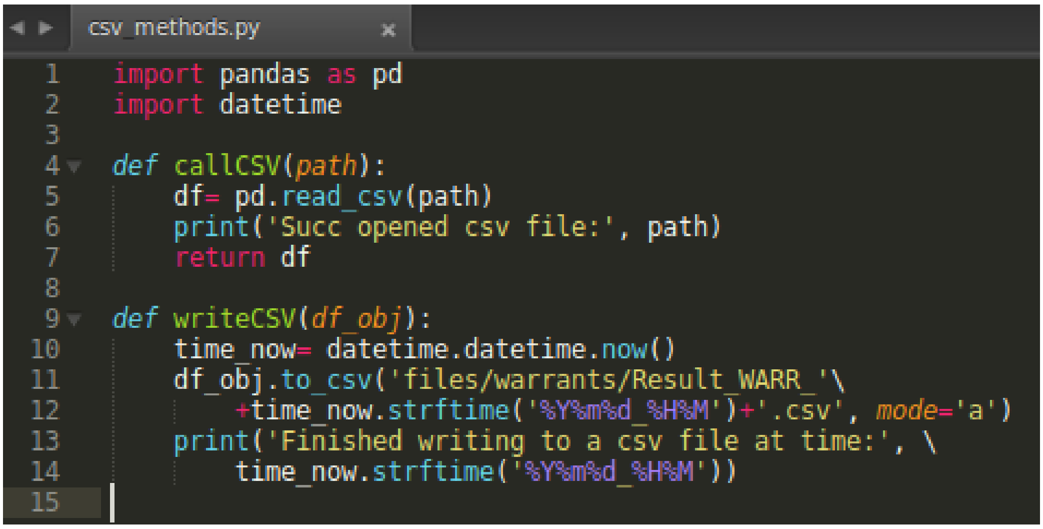
1. To easily identify what stock is currently oversold through scraping counter's Stochastic value.
2. Assists in decision making process by obtaining other counter or stock data such as maturity, RSI, and financial strength to name a few.
3. Compile all scraped data into a csv file and construct the data to be easily perceived and visible.

3 Methods and Solutions

3.1 Problem 1: Obtain KLSE Listed Counters into Python Data Structure

A file named ALL_WARRANTS.csv in /files program directory that has all the listed warrant counters will be read by python pandas module.

Methods and solutions are discussed below:



```python
import pandas as pd
import datetime

def callCSV(path):
    df= pd.read_csv(path)
    print('Succ opened csv file:', path)
    return df

def writeCSV(df_obj):
    time_now= datetime.datetime.now()
    df_obj.to_csv('files/warrants/Result_WARR_'\
        +time_now.strftime('%Y%m%d_%H%M')+'.csv', mode='a')
    print('Finished writing to a csv file at time:', \
        time_now.strftime('%Y%m%d_%H%M'))

```

Figure 1: Python methods to operate CSV document

1. /library folder located in the program default directory is used to saved the required python application services to scrape warrant counters attribute.

2. /library/csv_methods.py contains function methods to handle importing and exporting of a csv file.

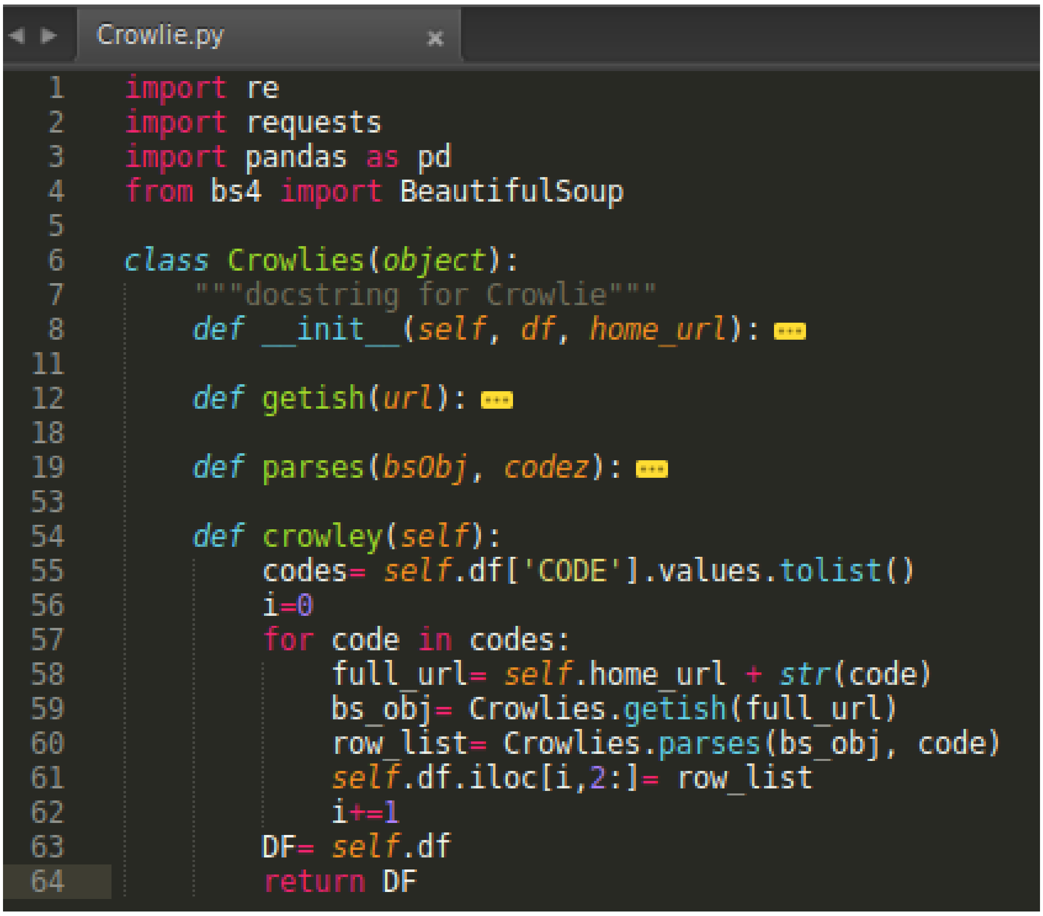
3. callCSV() method from /library/csv_methods.py will read the ALL_WARRANTS.csv contents and returns into python dataframe data structure.

4. The warrant parent counters will be using scraping logic from folder /library2 instead and the processes of importing and exporting csv document are similar.

3.2 Problem 2: Scraping Counter Data From Website

A website *https://www.klsescreener.com* is used to be the location for the python application to execute the data scraping procedures.

Methods and solutions are discussed below:



```python
import re
import requests
import pandas as pd
from bs4 import BeautifulSoup

class Crowlies(object):
    """docstring for Crowlie"""
    def __init__(self, df, home_url): ...

    def getish(url): ...

    def parses(bsObj, codez): ...

    def crowley(self):
        codes= self.df['CODE'].values.tolist()
        i=0
        for code in codes:
            full_url= self.home_url + str(code)
            bs_obj= Crowlies.getish(full_url)
            row_list= Crowlies.parses(bs_obj, code)
            self.df.iloc[i,2:]= row_list
            i+=1
        DF= self.df
        return DF
```

Figure 2: Scraping logic of the Python application

1. From /library folder, the python file Crowlie.py contains a python class object named Crowlies with the purpose of carrying the duties of scraping processes.

2. getish() method from the Crowlies class is executed for HTML GET request.

3. parses() method accepts a beautifulsoup object and counter code and returns a list containing the counter scraped attributes.

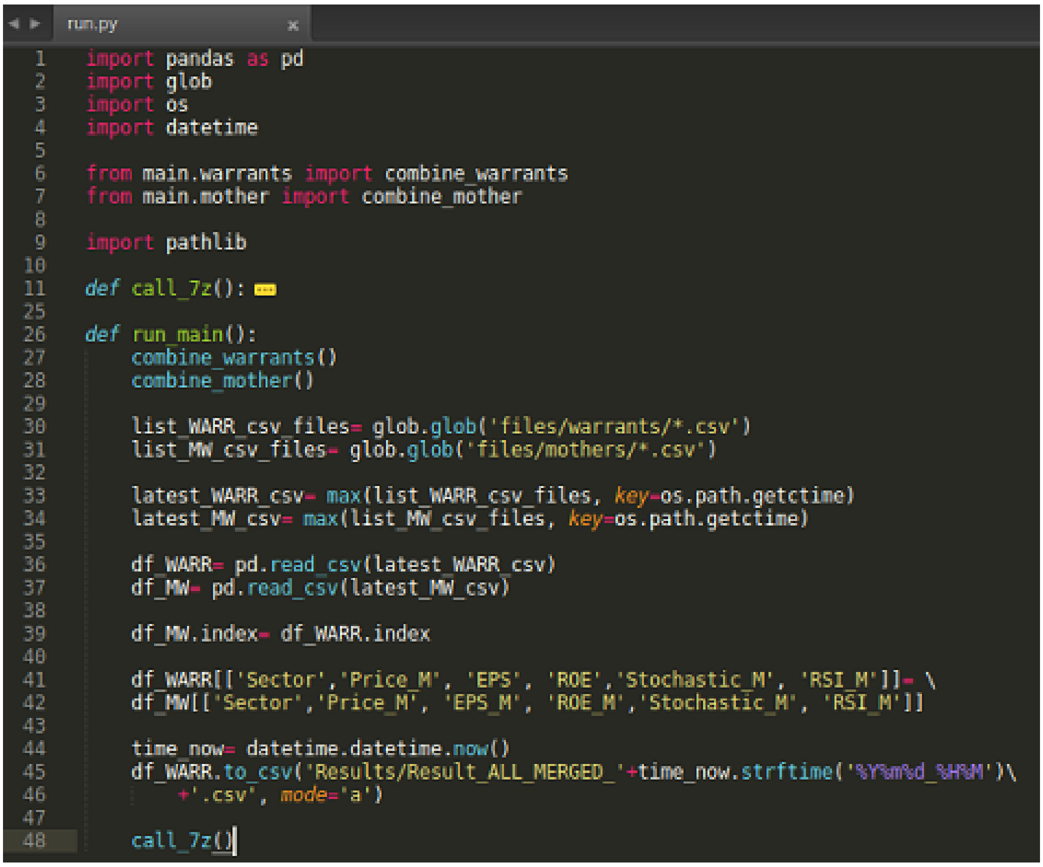
4. crowley() method is being a role of iterating obtained counter codes that was initialized during the instantiate of Crowlies class object.

5. The warrant parent counters will be using Crowlies class from folder /library2 instead for data scraping due to slightly different attributes and simplicity.

3.3 Problem 3: Combine Both Warrant and Parent Counters in One CSV Document

run.py in the /main folder will combine both scraped warrant and parent counters data into one csv document. To encrypt the gathered data inside the csv document, python os.system() method has been utilized to import bash command with call to 7-zip application for the encryption process.

The application produces two files, a csv and an encrypted zip document files which then will be located in /Results folder with the naming scheme Result_ALL_MERGED_date_time followed by file extension .csv and .zip respectively.



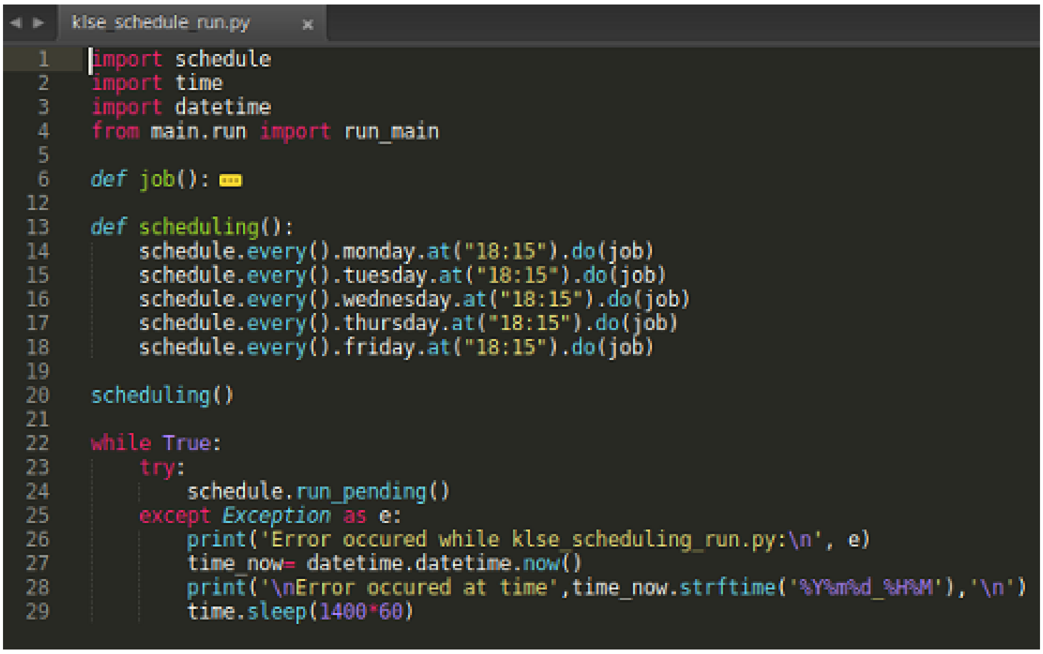
```
import pandas as pd
import glob
import os
import datetime

from main.warrants import combine_warrants
from main.mother import combine_mother

import pathlib

def call_7z(): ...

def run_main():
    combine_warrants()
    combine_mother()

    list_WARR_csv_files= glob.glob('files/warrants/*.csv')
    list_MW_csv_files= glob.glob('files/mothers/*.csv')

    latest_WARR_csv= max(list_WARR_csv_files, key=os.path.getctime)
    latest_MW_csv= max(list_MW_csv_files, key=os.path.getctime)

    df_WARR= pd.read_csv(latest_WARR_csv)
    df_MW= pd.read_csv(latest_MW_csv)

    df_MW.index= df_WARR.index

    df_WARR[['Sector','Price_M', 'EPS', 'ROE','Stochastic_M', 'RSI_M']]= \
    df_MW[['Sector','Price_M', 'EPS_M', 'ROE_M','Stochastic_M', 'RSI_M']]

    time_now= datetime.datetime.now()
    df_WARR.to_csv('Results/Result_ALL_MERGED_'+time_now.strftime('%Y%m%d_%H%M')\
        +'.csv', mode='a')

    call_7z()
```

Figure 3: run.py to compile the result data

3.4 Problem 4: Daily Scraping Run Schedule

Utilizing a python schedule module, the scraping program can be specified of its running time frame. Since market closed at time 1700 every weekday, the program has been set to run at time 1815 to make room for the website to finish updating its counters data.

The scheduling methods are being handled by klse_schedule_run.py file of which will be the starting point of the python program.

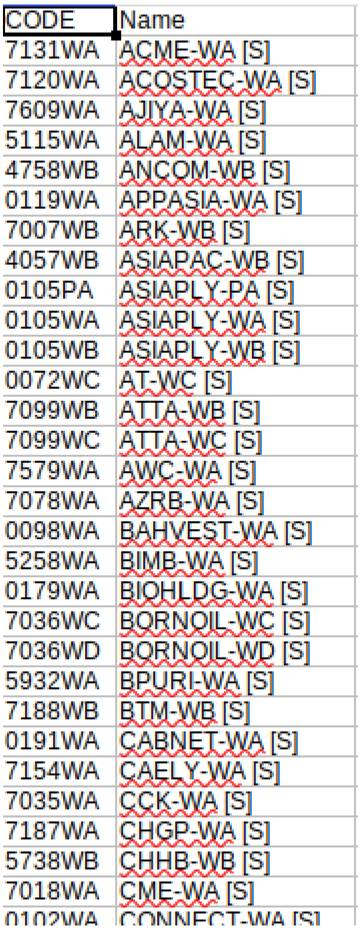


```python
import schedule
import time
import datetime
from main.run import run_main

def job():

def scheduling():
    schedule.every().monday.at("18:15").do(job)
    schedule.every().tuesday.at("18:15").do(job)
    schedule.every().wednesday.at("18:15").do(job)
    schedule.every().thursday.at("18:15").do(job)
    schedule.every().friday.at("18:15").do(job)

scheduling()

while True:
    try:
        schedule.run_pending()
    except Exception as e:
        print('Error occured while klse_scheduling_run.py:\n', e)
        time_now= datetime.datetime.now()
        print('\nError occured at time',time_now.strftime('%Y%m%d_%H%M'),'\n')
        time.sleep(1400*60)
```

Figure 4: Scheduling the python application

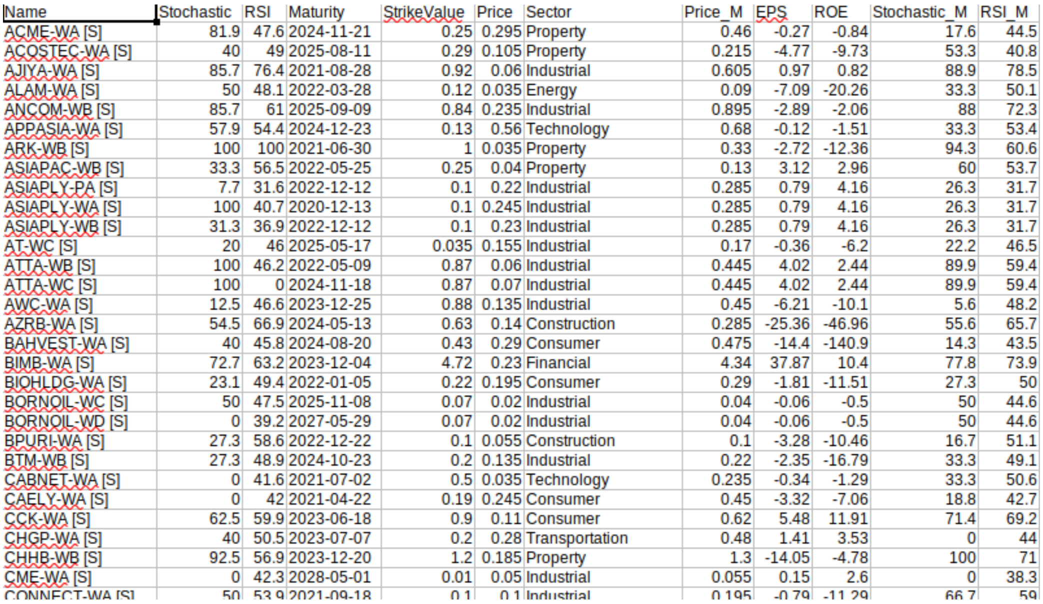
4 Result Example

The result obtained from the KLSE Listed Stocks Scraper Python application can be seen below:



CODE	Name
7131WA	ACME-WA [S]
7120WA	ACOSTEC-WA [S]
7609WA	AJIYA-WA [S]
5115WA	ALAM-WA [S]
4758WB	ANCOM-WB [S]
0119WA	APPASIA-WA [S]
7007WB	ARK-WB [S]
4057WB	ASIAPAC-WB [S]
0105PA	ASIAPLY-PA [S]
0105WA	ASIAPLY-WA [S]
0105WB	ASIAPLY-WB [S]
0072WC	AT-WC [S]
7099WB	ATTA-WB [S]
7099WC	ATTA-WC [S]
7579WA	AWC-WA [S]
7078WA	AZRB-WA [S]
0098WA	BAHVEST-WA [S]
5258WA	BIMB-WA [S]
0179WA	BIOHLDG-WA [S]
7036WC	BORNOIL-WC [S]
7036WD	BORNOIL-WD [S]
5932WA	BPURI-WA [S]
7188WB	BTM-WB [S]
0191WA	CABNET-WA [S]
7154WA	CAELY-WA [S]
7035WA	CCK-WA [S]
7187WA	CHGP-WA [S]
5738WB	CHHB-WB [S]
7018WA	CME-WA [S]
0102WA	CONNECT-WA [S]

Figure 5: ALL_WARRANT.csv document containing all listed counter



Name	Stochastic	RSI	Maturity	StrikeValue	Price	Sector	Price_M	EPS	ROE	Stochastic_M	RSI_M
ACME-WA [S]	81.9	47.6	2024-11-21	0.25	0.295	Property	0.46	-0.27	-0.84	17.6	44.5
ACOSTEC-WA [S]	40	49	2025-08-11	0.29	0.105	Property	0.215	-4.77	-9.73	53.3	40.8
AJIYA-WA [S]	85.7	76.4	2021-08-28	0.92	0.06	Industrial	0.605	0.97	0.82	88.9	78.5
ALAM-WA [S]	50	48.1	2022-03-28	0.12	0.035	Energy	0.09	-7.09	-20.26	33.3	50.1
ANCOM-WB [S]	85.7	61	2025-09-09	0.84	0.235	Industrial	0.895	-2.89	-2.06	88	72.3
APPASIA-WA [S]	57.9	54.4	2024-12-23	0.13	0.56	Technology	0.68	-0.12	-1.51	33.3	53.4
ARK-WB [S]	100	100	2021-06-30	1	0.035	Property	0.33	-2.72	-12.36	94.3	60.6
ASIAPAC-WB [S]	33.3	56.5	2022-05-25	0.25	0.04	Property	0.13	3.12	2.96	60	53.7
ASIAPLY-PA [S]	7.7	31.6	2022-12-12	0.1	0.22	Industrial	0.285	0.79	4.16	26.3	31.7
ASIAPLY-WA [S]	100	40.7	2020-12-13	0.1	0.245	Industrial	0.285	0.79	4.16	26.3	31.7
ASIAPLY-WB [S]	31.3	36.9	2022-12-12	0.1	0.23	Industrial	0.285	0.79	4.16	26.3	31.7
AT-WC [S]	20	46	2025-05-17	0.035	0.155	Industrial	0.17	-0.36	-6.2	22.2	46.5
ATTA-WB [S]	100	46.2	2022-05-09	0.87	0.06	Industrial	0.445	4.02	2.44	89.9	59.4
ATTA-WC [S]	100	0	2024-11-18	0.87	0.07	Industrial	0.445	4.02	2.44	89.9	59.4
AWC-WA [S]	12.5	46.6	2023-12-25	0.88	0.135	Industrial	0.45	-6.21	-10.1	5.6	48.2
AZRB-WA [S]	54.5	66.9	2024-05-13	0.63	0.14	Construction	0.285	-25.36	-46.96	55.6	65.7
BAHVEST-WA [S]	40	45.8	2024-08-20	0.43	0.29	Consumer	0.475	-14.4	-140.9	14.3	43.5
BIMB-WA [S]	72.7	63.2	2023-12-04	4.72	0.23	Financial	4.34	37.87	10.4	77.8	73.9
BIOHLDG-WA [S]	23.1	49.4	2022-01-05	0.22	0.195	Consumer	0.29	-1.81	-11.51	27.3	50
BORNOIL-WC [S]	50	47.5	2025-11-08	0.07	0.02	Industrial	0.04	-0.06	-0.5	50	44.6
BORNOIL-WD [S]	0	39.2	2027-05-29	0.07	0.02	Industrial	0.04	-0.06	-0.5	50	44.6
BPURI-WA [S]	27.3	58.6	2022-12-22	0.1	0.055	Construction	0.1	-3.28	-10.46	16.7	51.1
BTM-WB [S]	27.3	48.9	2024-10-23	0.2	0.135	Industrial	0.22	-2.35	-16.79	33.3	49.1
CABNET-WA [S]	0	41.6	2021-07-02	0.5	0.035	Technology	0.235	-0.34	-1.29	33.3	50.6
CAELY-WA [S]	0	42	2021-04-22	0.19	0.245	Consumer	0.45	-3.32	-7.06	18.8	42.7
CCK-WA [S]	62.5	59.9	2023-06-18	0.9	0.11	Consumer	0.62	5.48	11.91	71.4	69.2
CHGP-WA [S]	40	50.5	2023-07-07	0.2	0.28	Transportation	0.48	1.41	3.53	0	44
CHHB-WB [S]	92.5	56.9	2023-12-20	1.2	0.185	Property	1.3	-14.05	-4.78	100	71
CME-WA [S]	0	42.3	2028-05-01	0.01	0.05	Industrial	0.055	0.15	2.6	0	38.3
CONNECT-WA [S]	50	53.9	2021-09-18	0.1	0.1	Industrial	0.195	-0.79	-11.29	66.7	59

Figure 6: Result data obtained from the Python application

5 Appendix

A Modules Dependency

Below are extra python modules required for the application to run. Those can be installed using python pip module.
1. requests
2. BeautifulSoup4
3. pandas
4. schedule

B Improvements

Improvements for the application are mentioned below:

1. Scraping recent news data and utilizing machine learning to better identify current market trend.

2. Develop a graph to visualize historical scrapped data. This can be done using matplotlib python module.

3. Store scrapped data in a database for more concise data compilation.