

PROJECT

<코스피, 코스닥 시장 내 기업들의 현재 주가가 분석>

보고서 및 논문 윤리 서약

1. 나는 보고서 및 논문의 내용을 조작하지 않겠습니다.
2. 나는 다른 사람의 보고서 및 논문의 내용을 내 것처럼 무단으로 복사하지 않겠습니다.
3. 나는 다른 사람의 보고서 및 논문의 내용을 참고하거나 인용할 시 참고 및 인용 형식을 갖추고 출처를 반드시 밝히겠습니다.
4. 나는 보고서 및 논문을 대신하여 작성하도록 청탁하지도 청탁받지도 않겠습니다.

나는 보고서 및 논문 작성 시 위법 행위를 하지 않고, 명지인으로서 또한 공학인으로서 나의 양심과 명예를 지킬 것을 약속합니다.



학 과 : 영어영문학과

과 목 : 빅데이터프로그래밍

담당교수 : 권동섭 교수님

반 : 화,목(15:00~16:15)

학 번 : 60152177

이 름 : 김현목

(서명)

<github 주소 - https://github.com/HyunKim24/bigdata_programming_project >

1. 문제 정의

국내 증시는 크게 코스피 시장과 코스닥 시장으로 나뉜다. 각 시장 내에서는 다양한 지표를 제공하며, 여러 개의 지표를 통해 해당 기업에 투자 여부를 평가하게 된다. 하지만 본 프로젝트에서는 다양한 지표 중 '현재가'의 지표를 이용하여 진행한다. '현재가'는 실시간으로 반영되는 지표로서, 해당 지표의 변화 폭을 실시간으로 확인 할 수 있다. 이를 통해 기업의 성장과 정체 그리고 하락 여부를 판단 해본다. 따라서 본 프로젝트에서는 각 시장에서 현재가가 가장 높은 기업 5개와 가장 낮은 기업 5개를 선정하고, 해당 기업들의 날짜별 평균 현재가의 변동을 알아본다. 변동을 통해 해당 기업이 성장하는 기업인지, 정체 및 하락하는 기업인지를 통해 투자자들의 의사결정을 돕는다.

2. 시스템 아키텍처

1) 데이터 수집 - ANACONDA, Python, Windows Task Scheduler



windows task scheduler를 사용하는 이유는 파이썬 코드를 올리면 자동으로 스크래핑을 진행하기 위하여 사용한다. 본 스크래핑을 통해 30분 간격으로 데이터가 저장된다.

2) 데이터 저장

포맷 : CSV and xlsx



확장자가 다른 2개의 동일한 파일을 만든 이유는, 추후 전처리를 진행하고 최종 dataset을 구축 할 때, 한글로 된 데이터가 깨질 것을 우려하였기 때문이다. csv 파일은 한글 데이터가 종종 깨질 수 있다. csv 파일에서 수집한 데이터가 사용할 수 없을 때는 xlsx를 사용하기 위해 2개의 파일 형식을 이용하였다.

3) 데이터 전처리 및 최종 dataset 구성 - Python, csv



스크래핑한 데이터를 전처리하고 모든 데이터를 통합하여 최종 dataset을 구성한다. 데이터 파일이 csv 파일 형식이기 때문에 작업이 용이한 파이썬으로 작업한다. 추후 최종 데이터셋은 csv 파일로 저장한다.

4) 데이터 분석 - HDFS, HIVE



최종 dataset을 HDFS에 업로드하고, HIVE를 통해 분석한다. HIVE를 사용한 이유는 HIVEQL문으로 분석하는 것이 본 프로젝트의 목적을 가장 잘 반영할 수 있다고 생각하였기 때문이다.

3. 데이터 수집 방법

```
# 오늘의 날짜를 가져온다.
import pandas as pd
import datetime
import numpy as np
import time
import os
from bs4 import BeautifulSoup
import requests
from selenium import webdriver

# 주식 데이터 크롤링
def stock_crawling(sosok, market, per_page):
    for page in range(1, per_page):
        url = 'https://finance.naver.com/sise/sise_market_sum.nhn?sosok='+str(sosok)+'&page='+str(page)
        result = requests.get(url)
        soup = BeautifulSoup(result.content, 'html.parser')

        stock_table = soup.find("table", {"class": "type_2"})
        summary_stock = stock_table.find('tbody')
        data_list = summary_stock.find_all('tr')
        data_list = data_list[1:]

        # tmp 리스트에 tr 데이터의 공백을 제거하고 append를 한다.
        tmp = list()
        for tr_data in data_list:
            for i in tr_data:
                try:
                    preprocessing = i.text.strip()
                    if preprocessing != '':
                        tmp.append(i.text.strip())
                except:
                    pass

        # 들어온 데이터는 12개씩 하나의 tr에서 온 데이터다. 각각의 데이터들의 인덱스를 활용하여
        # 해당하는 리스트에 append 해준다.
        index = 0
        for i in tmp:
            if index%12 == 0:
                ranking_list.append(i)
                index+=1

            elif index%12 == 1:
                company_list.append(i)
                index+=1

            elif index%12 == 2:
                price_list.append(i)
                index+=1

            elif index%12 == 3:
                per_yesterday_list.append(i)
                index+=1
```

```

        elif index%12 == 4:
            indecrease_list.append(i)
            index+=1

        elif index%12 == 5:
            acmyeonga_list.append(i)
            index+=1

        elif index%12 == 6:
            siga_total_list.append(i)
            index+=1

        elif index%12 == 7:
            sangjang_stock_list.append(i)
            index+=1

        elif index%12 == 8:
            foreign_part_list.append(i)
            index+=1

        elif index%12 == 9:
            perstock_list.append(i)
            index+=1

        elif index%12 == 10:
            per_list.append(i)
            index+=1

        elif index%12 == 11:
            roe_list.append(i)
            index+=1

    # tr의 갯수만큼 날짜와 시간, 주식시장을 리스트에 넣어준다.
    for i in range(int(len(tmp)/12)):
        date_list.append(today)
        time_list.append(time)
        market_list.append(market)

    ...

if __name__ == "__main__":
    # 오늘의 날짜와 시간을 가져온다.
    todays_date = datetime.datetime.now()
    today = str(todays_date.year) + '-' + str(todays_date.month) + '-' + str(todays_date.day)
    time = str(todays_date.hour) + ':' + str(todays_date.minute)

    # 코스피 목록의 마지막 페이지 넘버를 알아낸다.
    driver1 = webdriver.Chrome('C:\\Users\\HyunMok\\Desktop\\projectforhadoop\\chromedriver_win32\\chromedriver.exe')
    driver1.get('https://finance.naver.com/sise/sise_market_sum.nhn?sosok=0')
    # 맨 마지막 페이지를 클릭한다.
    driver1.find_element_by_css_selector('#contentarea > div.box_type_1 > table.Nnavi > tbody > tr > td.pgRR > a').click()
    # url에서 'page=' 뒤에 있는 숫자를 슬라이싱으로 가져온다.
    reference_num = driver1.current_url.find('page=')
    last_num_kospi = int(driver1.current_url[reference_num+5:])

    driver1.close()

    # 코스닥 목록의 마지막 페이지 넘버를 알아낸다.
    driver2 = webdriver.Chrome('C://Users//HyunMok//Desktop//projectforhadoop//chromedriver_win32//chromedriver.exe')
    driver2.get('https://finance.naver.com/sise/sise_market_sum.nhn?sosok=1')
    # 맨 마지막 페이지를 클릭한다.
    driver2.find_element_by_css_selector('#contentarea > div.box_type_1 > table.Nnavi > tbody > tr > td.pgRR > a').click()
    # url에서 'page=' 뒤에 있는 숫자를 슬라이싱으로 가져온다.
    reference_num2 = driver2.current_url.find('page=')
    last_num_kosdac = int(driver2.current_url[reference_num+5:])

    driver2.close()

```

```

# 필요한 변수 선언
date_list = list()
time_list = list()
ranking_list = list()
market_list = list()
company_list = list()
price_list = list()
per_yesterday_list = list()
indecrease_list = list()
acmyeonga_list = list()
siga_total_list = list()
sangjang_stock_list = list()
foreign_part_list = list()
perstock_list = list()
per_list = list()
roe_list = list()

# 크롤링 함수로 크롤링을 실행한다.
stock_crawling(0, '코스피', last_num_kospi+1)
stock_crawling(1, '코스닥', last_num_kosdac+1)

# 최종 데이터 프레임을 만들고 데이터를 넣는다.
today_stock_list = {'날짜':date_list, "시간":time_list, "시장":market_list, 'NO':ranking_list, "종목명":company_list,
                    "등락률":indecrease_list, "액면가":acmyeonga_list, '시가총액':siga_total_list, '상장주식수':sangjang_stock_list,
                    '거래량':perstock_list, 'PER':per_list, 'ROE':roe_list}
today_summary = pd.DataFrame(today_stock_list)

# 데이터 프레임을 csv와 excel로 저장한다.
# 2가지 파일 형태로 저장하는 이유 - csv로 저장시 한글이 깨질 수도 있음 -> 만약 csv에서 한글이 깨질경우
# 엑셀파일을 이용하기 위해 2가지 파일로 저장함
today_summary.to_excel('stock_'+str(today_date.year) + str(today_date.month)+str(today_date.day)+'_'+str(today_date.day)+'.xlsx')
today_summary.to_csv('stock_'+str(today_date.year) + str(today_date.month)+str(today_date.day)+'_'+str(today_date.day)+'.csv')

```

파이썬으로 작성된 코드를 바탕으로 네이버에서 제공하는 코스피에 상장된 기업들의 정보와 코스닥에 상장된 기업들의 정보를 스크래핑한다. 스크래핑은 BeautifulSoup을 이용한다. 해당 라이브러리를 사용하는 이유는 스크래핑 할 사이트의 URL이 동일한 패턴을 지녔기 때문이다. 가령 코스피는 'https://finance.naver.com/sise/sise_market_sum.nhn?sosok=0&page=1' 라는 url을 가지며 코스피에 상장된 기업들의 정보는 다음 화면과 같이 보인다.

finance.naver.com/sise/sise_market_sum.nhn?sosok=0&page=1

37	넷마블	90,300	▲ 1,100	+1.23%	100	77,426	85,743	22.63
38	KODEX 200	29,050	▲ 445	+1.56%	0	73,148	251,800	10.23
39	기업은행	12,400	▲ 300	+2.48%	5,000	71,567	577,157	20.22
40	KT	26,950	▲ 50	+0.19%	5,000	70,370	261,112	47.93
41	아모레퍼	80,500	▲ 1,700	+2.16%	500	66,379	82,458	22.04
42	웅진코웨이	88,900	▼ 500	-0.56%	500	65,608	73,800	59.51
43	강원랜드	30,650	▲ 650	+2.17%	500	65,573	213,940	30.18
44	LG유플러스	13,900	0	0.00%	5,000	60,689	436,611	36.73
45	한온시스템	11,100	0	0.00%	100	59,252	533,800	19.22
46	LG디스플레이	15,950	▼ 100	-0.62%	5,000	57,072	357,816	21.25
47	현대엘리베이터	149,000	▲ 2,000	+1.36%	500	55,875	37,500	37.00
48	현대중공업지주	342,500	▲ 500	+0.15%	5,000	55,782	16,287	19.44
49	미래에셋대우	7,720	▲ 260	+3.49%	5,000	50,822	658,316	15.26
50	현대건설	43,900	▲ 350	+0.80%	5,000	48,885	111,356	22.65

1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 다음 > | 맨뒤 >>

여기서 페이지 번호를 눌러 해당 페이지로 이동하면 해당 url에서 'page=' 뒤 부분의 숫자가 내가 클릭한 페이지 번호와 동일해진다.

finance.naver.com/sise/sise_market_sum.nhn?sosok=0&page=2

87	CJ	96,300	▲ 300	+0.31%	5,000	28,097	29,177	18.48
88	BGF리테일	161,000	▲ 2,000	+1.26%	1,000	27,827	17,284	34.49
89	현대해상	29,700	▲ 750	+2.59%	500	26,552	89,400	44.81
90	대한항공	27,350	▲ 450	+1.67%	5,000	25,940	94,845	18.98
91	BNK금융지주	7,870	▲ 270	+3.55%	5,000	25,651	325,935	52.16
92	GS건설	32,000	▲ 500	+1.59%	5,000	25,576	79,924	30.41
93	KODEX 레버리지	13,720	▲ 455	+3.43%	0	24,586	179,200	0.28
94	오션저라이트	29,900	▲ 1,000	+3.46%	1,000	24,518	82,000	27.05
95	한미사이언스	37,450	▲ 900	+2.46%	500	24,246	64,743	2.63
96	포스코인터내셔널	19,600	▲ 200	+1.03%	5,000	24,182	123,375	15.38
97	한진칼	40,700	▼ 50	-0.12%	2,500	24,082	59,170	12.37
98	금호석유	78,800	▲ 2,500	+3.28%	5,000	24,009	30,468	34.15
99	KCC	227,000	▲ 9,000	+4.13%	5,000	23,963	10,557	20.42
100	샘오션	4,450	▲ 80	+1.83%	1,000	23,788	534,569	11.03

<< 맨앞 < | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 다음 > | 맨뒤 >>

따라서 마지막 페이지 번호만을 알아내어 BeautifulSoup으로 해당 데이터를 스크래핑한다.

(코스닥의 경우 위 url에서 sosok=1로 변경되는 것 외에 동일한 주소를 지님)

스크래핑을 할 때는 데이터가 있는 테이블에서 tr tag만을 가져온다. 여기서 주의해야 할 점

```
# tmp 리스트에 tr 데이터의 공백을 제거하고 append를 한다.
tmp = list()
for tr_data in data_list:
    for i in tr_data:
        try:
            preprocessing = i.text.strip()
            if preprocessing != '':
                tmp.append(i.text.strip())
        except:
            pass
```

은 본 사이트는 5개의 목록 후 빈 tr 태그가 3개가 있는 패턴이 반복되어있다. 빈 tr태그는 데이터와는 상관이 없으므로 빈 tr 태그가 아닌 것들만 따로 리스트에 append 해준다.

```
# 코스닥 목록의 마지막 페이지 넘버를 알아낸다.
driver1 = webdriver.Chrome('C:\\Users\\HyunMok\\Desktop\\projectforhadoop\\chromedriver_win32\\chromedriver.exe')
driver1.get('https://finance.naver.com/sise/sise_market_sum.nhn?sosok=0')
# 맨 마지막 페이지를 클릭한다.
driver1.find_element_by_css_selector('#contentarea > div.box_type_1 > table.Nnavi > tbody > tr > td.pgRR > a').click()
# url에서 'page=' 뒤에 있는 숫자를 슬라이싱으로 가져온다.
reference_num = driver1.current_url.find('page=')
last_num_kospi = int(driver1.current_url[reference_num+5:])

driver1.close()

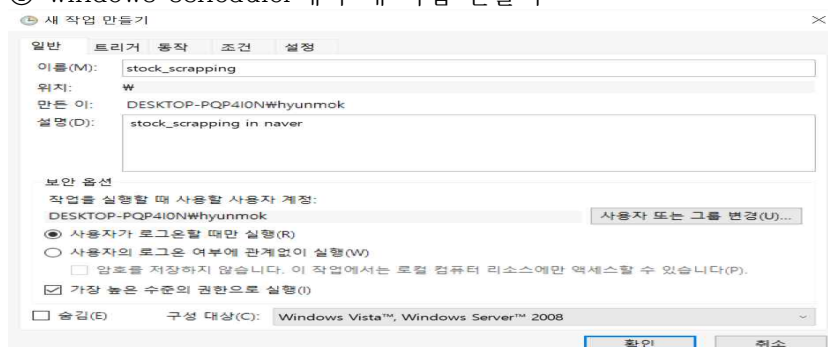
# 코스닥 목록의 마지막 페이지 넘버를 알아낸다.
driver2 = webdriver.Chrome('C:\\Users\\HyunMok\\Desktop\\projectforhadoop\\chromedriver_win32\\chromedriver.exe')
driver2.get('https://finance.naver.com/sise/sise_market_sum.nhn?sosok=1')
# 맨 마지막 페이지를 클릭한다.
driver2.find_element_by_css_selector('#contentarea > div.box_type_1 > table.Nnavi > tbody > tr > td.pgRR > a').click()
# url에서 'page=' 뒤에 있는 숫자를 슬라이싱으로 가져온다.
reference_num2 = driver2.current_url.find('page=')
last_num_kosdac = int(driver2.current_url[reference_num+5:])

driver2.close()
```

마지막 페이지를 알아내기 위해서는 selenium을 사용한다. 해당 페이지를 selenium으로 접근한 뒤, 페이지 번호 가장 우측에 맨뒤 버튼을 누르면 마지막 페이지로 이동한다. 마지막 페이지도 역시 url에 'page=' 뒤에 해당 페이지 번호가 나온다. 해당 url을 current_url 함수로 받아오고 'page='에서 p의 인덱스를 알아낸다. p의 인덱스에 5를 더하면 숫자가 시작하는 부분의 인덱스가 나오기 때문에 그 위치에서 끝까지 슬라이싱을 하고, 형 변환을 통해 int형으로 변경해주면 마지막페이지 번호를 얻을 수 있다.

데이터를 수집 할 때는 30분 간격으로 자동으로 실행되도록 설정한다. 여기서 windows에서 제공하는 task scheduler를 이용하여 자동화를 만들어준다. 자동화를 사용하는 이유는 주식 데이터는 실시간으로 업데이트된다. 따라서 주기적으로 변화하는 데이터를 모으기 위하여 사용한다. task scheduler를 작업하는 과정은 아래와 같다.

① windows scheduler에서 새 작업 만들기



② 트리거 설정하기

새 트리거 만들기

작업 시작(G): 예약 상태

설정

☒ 한 번(N) ☐ 매일(D) ☐ 매주(W) ☐ 매월(M)

시작(S): 2019-12-10 오전 9:30:00 ☐ 표준 시간대 간 동기화(Z)

고급 설정

☐ 작업이 지연되는 최대 시간(일의 지연)(K): 1 시간

☒ 작업 반복 간격(P): 30 분 ☐ 반복 기간이 종료될 때 실행 중인 모든 작업 중지(I) 기간(F): 무기한으로

☐ 다음 기간 이상 실행되는 작업 중지(L): 3 일

☐ 만료(O): 2020-12-15 오후 4:27:11 ☐ 표준 시간대 간 동기화(E)

☒ 사용(B)

확인 취소

③ 동작 만들기

새 동작 만들기

실행할 작업을 지정해야 합니다.

동작(D): 프로그램 시작

설정

프로그램/스크립트(P): C:\Users\Whyunmok\Anaconda3\pythonw.exe

인수 추가(옵션)(A): stock_crawling.py

시작 위치(옵션)(I): ok\WDesktop\stock_data

확인 취소

④ 해당 작업 확인하기

stock_scrapp...	준비	2019-12-10 오전 9:30에 - 트리거된 후 무기한으로 30 분마다 반복합니다.
Update Chec...	준비	여러 개의 트리거가 정의되었습니다.
User_Feed_S...	준비	매일 오후 9:41에 - 2019-12-15 오후 9:41:51에 트리거가 만료됩니다.

만들어진 스케줄러를 이용하면 12월 10일 오전 9시 30분부터 30분 간격으로 데이터가 수집되어 아래와 같은 csv 파일 형식과 xlsx 파일 형식으로 2개의 동일한 파일이 저장된다. 자동으로 수집된 데이터는 csv 파일 기준 16185KB, 165326 Row, 14 Columns 이다.

<파일이 저장된 화면 예시>

stock_20191210_9시30분	2019-12-10 오전 9:30	Microsoft Excel 실행...
stock_20191210_9시30분	2019-12-10 오전 9:30	Microsoft Excel 워크...
stock_20191210_10시0분	2019-12-10 오전 10:00	Microsoft Excel 실행...
stock_20191210_10시0분	2019-12-10 오전 10:00	Microsoft Excel 워크...
stock_20191210_10시30분	2019-12-10 오전 10:30	Microsoft Excel 실행...
stock_20191210_10시30분	2019-12-10 오전 10:30	Microsoft Excel 워크...
stock_20191210_11시0분	2019-12-10 오전 11:00	Microsoft Excel 실행...
stock_20191210_11시0분	2019-12-10 오전 11:00	Microsoft Excel 워크...
stock_20191210_11시30분	2019-12-10 오전 11:31	Microsoft Excel 실행...
stock_20191210_11시30분	2019-12-10 오전 11:31	Microsoft Excel 워크...
stock_20191210_12시0분	2019-12-10 오후 12:00	Microsoft Excel 실행...
stock_20191210_12시0분	2019-12-10 오후 12:00	Microsoft Excel 워크...
stock_20191210_12시30분	2019-12-10 오후 12:30	Microsoft Excel 실행...
stock_20191210_12시30분	2019-12-10 오후 12:30	Microsoft Excel 워크...
stock_20191210_13시0분	2019-12-10 오후 1:01	Microsoft Excel 실행...
stock_20191210_13시0분	2019-12-10 오후 1:01	Microsoft Excel 워크...
stock_20191210_13시30분	2019-12-10 오후 1:30	Microsoft Excel 실행...
stock_20191210_13시30분	2019-12-10 오후 1:30	Microsoft Excel 워크...

4. 데이터 분석 방법

① 데이터 전처리

```
import os
import pandas as pd

def stock_preprocessing():
    # 파일 목록 가져오기
    path = "C:\\Users\\hyunmok\\Desktop\\stock_data"
    file_list = os.listdir(path)

    # 파일 목록 중 확장자가 csv인 파일만 따로 저장하기
    stock_csv = [file for file in file_list if file.endswith(".csv")]

    # 주가가 변동하는 시간에 크롤링된 데이터만 추출한다.
    time = ['_9시30분', '_10시0분', '_10시30분', '_11시0분', '_11시30분', '_12시0분',
            '_12시30분', '_13시0분', '_13시30분', '_14시0분', '_14시30분', '_15시0분', '_15시30분', '_16시0분']

    stock_time_csv = list()
    for i in stock_csv:
        for j in time:
            if j in i:
                stock_time_csv.append(i)

    # 수집한 데이터의 컬럼 목록을 가져온다.
    tmp_columns = pd.read_csv('C:\\Users\\hyunmok\\Desktop\\stock_data\\'+stock_csv[0]).columns

    # 최종 데이터셋을 구성할 데이터프레임 선언
    final_dataset = pd.DataFrame(columns=tmp_columns)

    # final_dataset에 모든 데이터를 병합한다.
    for i in stock_time_csv:
        data = pd.read_csv('C:\\Users\\hyunmok\\Desktop\\stock_data\\'+i)
        final_dataset = pd.concat([final_dataset, data])

    # 불필요한 열을 제거한다.
    final_dataset.drop(['Unnamed: 0', 'NO'], axis=1, inplace=True)

    # 계산에 필요한 컬럼들은 천의자리의 콤마를 지우고 데이터 타입은 numeric으로 변경
    final_dataset['현재가'] = final_dataset.현재가.str.replace(',', '').astype('int64')
    final_dataset['전일비'] = final_dataset.전일비.str.replace(',', '').astype('int64')
    final_dataset['액면가'] = final_dataset.액면가.str.replace(',', '').astype('int64')
    final_dataset['시가총액'] = final_dataset.시가총액.str.replace(',', '').astype('int64')
    final_dataset['거래량'] = final_dataset.거래량.str.replace(',', '').astype('int64')
    final_dataset['상장주식수'] = final_dataset.상장주식수.str.replace(',', '').astype('int64')

    final_dataset['외국인비율'] = final_dataset.외국인비율.astype('float')
    final_dataset['PER'] = final_dataset.PER.str.replace(',', '').astype('float')
    final_dataset['ROE'] = final_dataset.ROE.str.replace(',', '').astype('float')

    # 시간 데이터를 포맷팅해준다.
    time = final_dataset['시간']
    new_time = list()
    for i in time:
        t = i.split(':')
        if len(t[0]) != 2:
            t[0] = '0'+t[0]
            new_time.append(":".join(t))
        elif len(t[1]) != 2:
            t[1] = t[1]+'0'
            new_time.append(":".join(t))
        else:
            new_time.append(":".join(t))

    final_dataset['시간'] = new_time
```



```

# 날짜와 시간으로 정렬을 해주고 인덱스를 재설정 해준다.
final_dataset.sort_values(['날짜','시간'],ascending=True,inplace = True)
final_dataset.reset_index(drop=True,inplace=True)

# 최종 파일을 csv 형태로 저장해준다.
final_dataset.to_csv('final_dataset.csv',encoding = 'utf-8')

if __name__ == "__main__":
    stock_preprocessing()

```

데이터를 전처리하기 위해 먼저 주식시장이 개장하는 시간과 변동이 있는 시간에 스크래핑 된 데이터 파일들의 이름 목록을 리스트에 추가해준다. 그 후 최종적으로 구성될 데이터프레임을 위한 변수를 선언하고, 해당 데이터프레임에 컬럼을 부여한다. 그 후 파일들의 이름이 있는 리스트를 반복하며 계속하여 밑으로 추가해준다. 다 추가한 뒤, 불필요한 열은 제거하고, 계산에 사용될 데이터들은 int와 float형으로 변환한다. 그 후 시간을 같은 형식으로 맞춘다. 가령, 9:30이나 10:00으로 저장된 시간들이 있다. 시간별로 정렬을 해야 할 경우, 오전 9시 30분이 가장 앞에 있어야 하는데 이와 같은 경우는 가장 뒤로 간다. 따라서 '시'도 2자리로, '분'도 2자리로 통일 시켜준다. 이렇게 전처리된 파일은 최종 final_dataset.csv 파일로 저장한다.

② HDFS에 파일 업로드 및 HIVE를 위한 테이블 만들기

해당 파일을 HDFS에 업로드 한 후, HIVE QL문을 이용하여 해당 파일들의 데이터를 테이블로 저장한다.

```

DROP TABLE stock;
CREATE TABLE IF NOT EXISTS stock(
    no INT,
    date_yyyy_mm_dd STRING,
    time STRING,
    market STRING,
    company_name STRING,
    now_price INT,
    per_yesterday INT,
    in_decrease STRING,
    per_value INT,
    market_cap INT,
    listed_stock INT,
    foreing_rate DOUBLE,
    trading_volume INT,
    per DOUBLE,
    roe DOUBLE
)
row format delimited fields terminated BY ',' lines terminated BY '\n'
tblproperties("skip.header.line.count"="1");

LOAD DATA INPATH '/user/maria_dev/stock/final_dataset.csv' OVERWRITE INTO TABLE stock;

SELECT * FROM stock LIMIT 30;

```

<실행 결과>

stock.no	stock.date_yyyy_mm_dd	stock.time	stock.market	stock.company_name	stock.now_price	stock.per_yesterday	stock.in_decrease	stock.per_value	stock.market_cap	stock.listed_stock	stock.foreing_rate	stock.trading_volume	stock.per	stock.roe
0	2019-12-10	09:30	코스피	삼성전자	50900	300	-0.59%	100	3038619	5969783	56.97	1083521	8.45	19.63
1	2019-12-10	09:30	코스피	SK하이닉스	80100	600	-0.74%	5000	583130	728002	49.98	335388	3.75	38.53
2	2019-12-10	09:30	코스피	삼성전자우	41800	100	-0.24%	100	343967	822887	92.2	90090	6.94	null
3	2019-12-10	09:30	코스피	NAVER	173500	1500	-0.86%	100	289951	164813	58.68	32597	44.07	12.97
4	2019-12-10	09:30	코스피	현대차	120000	500	+0.42%	5000	256402	213668	41.58	50625	22.42	2.2
5	2019-12-10	09:30	코스피	삼성바이오로직스	385500	2500	-0.64%	2500	255066	66165	9.77	10729	113.82	5.51
6	2019-12-10	09:30	코스피	현대모비스	254000	2000	+0.79%	5000	242079	95307	47.74	26055	13.09	6.3
7	2019-12-10	09:30	코스피	셀트리온	167500	500	-0.30%	1000	214966	128338	20.04	69066	81.75	10.84
8	2019-12-10	09:30	코스피	LG화학	298500	3500	+1.19%	5000	210718	70592	37.95	36863	15.87	8.86
9	2019-12-10	09:30	코스피	신한지주	43750	350	-0.79%	5000	207462	474200	64.77	90553	6.57	9.21
10	2019-12-10	09:30	코스피	POSCO	231500	1500	-0.64%	5000	201838	87187	51.9	14080	11.94	3.88
11	2019-12-10	09:30	코스피	KB금융	48050	0	0.00%	5000	200903	418112	66.7	100855	6.56	8.78
12	2019-12-10	09:30	코스피	LG생활건강	1264000	3000	-0.24%	5000	197414	15618	45.15	1864	32.8	20.98
13	2019-12-10	09:30	코스피	SK물류	238000	1000	-0.42%	500	192175	80746	37.58	10815	6.14	15.52
14	2019-12-10	09:30	코스피	삼성물산	99000	800	-0.80%	100	187793	189690	13.59	33109	11.06	8.06
15	2019-12-10	09:30	코스피	한국전력	28600	0	0.00%	5000	183602	641964	25.15	196796	-13.96	-1.86
16	2019-12-10	09:30	코스피	SK	260000	0	0.00%	200	182937	70360	25.33	5134	8.18	14.88
17	2019-12-10	09:30	코스피	기아차	43850	500	+1.15%	5000	177752	405363	42.26	65534	15.38	4.27
18	2019-12-10	09:30	코스피	삼성SDI	223000	0	0.00%	5000	153345	68765	43.49	18234	22.39	6.05
19	2019-12-10	09:30	코스피	삼정엔지니어스	191000	1500	-0.78%	500	147792	77378	12.77	2858	23.48	10.91
20	2019-12-10	09:30	코스피	삼성생명	72700	300	-0.41%	500	145400	200000	15.7	15590	8.74	5.95
21	2019-12-10	09:30	코스피	SK이노베이션	146000	500	+0.34%	5000	135000	92466	35.02	28496	8.29	9.12
22	2019-12-10	09:30	코스피	KT&G	95500	0	0.00%	5000	131114	137292	47.45	40219	14.54	11.38
23	2019-12-10	09:30	코스피	카카오	149000	2500	-1.65%	500	128396	86172	30.33	50548	243.07	1.05
24	2019-12-10	09:30	코스피	LG	72600	0	0.00%	5000	125276	172557	34.48	17472	6.85	10.96
25	2019-12-10	09:30	코스피	엔씨소프트	539000	1000	-0.19%	500	118332	21954	49.81	9443	28.28	16.44
26	2019-12-10	09:30	코스피	LG전자	70000	900	+1.30%	5000	114553	163648	33.91	50055	10.21	9.03
27	2019-12-10	09:30	코스피	삼성화재	234000	2000	-0.85%	500	110857	47375	47.66	3019	11.05	8.81
28	2019-12-10	09:30	코스피	아모레퍼시픽	189500	3500	+1.88%	500	110779	58458	31.71	24142	39.37	7.75
29	2019-12-10	09:30	코스피	하나금융지주	36500	250	-0.68%	5000	109588	300242	67.11	110156	4.89	8.88

③ 최초 수집 날짜와 수집 시간에 코스피 시장에서 현재가가 가장 높은 기업 5개를 찾고 결과 테이블 CSV로 저장한다.

```
SELECT best_top5.company_name
FROM(SELECT kospi.company_name,kospi.now_price
FROM(SELECT stock.company_name,stock.date_yyyy_mm_dd,stock.time,stock.now_price
FROM stock
WHERE stock.market == '코스피') AS kospi
WHERE kospi.date_yyyy_mm_dd == '2019-12-10' and kospi.time == '09:30'
ORDER BY kospi.now_price DESC LIMIT 5) AS best_top5
```

Filter columns
✕

best_top5.company_name

LG생활건강

태광산업

LG생활건강우

영풍

엔씨소프트

SAVE TO HDFS

DOWNLOAD AS CSV

④ 최초 수집 날짜와 수집 시간에 코스피 시장에서 현재가가 가장 낮은 기업 5개를 찾고 결과 테이블 CSV로 저장한다.

```
SELECT worst_top5.company_name
FROM(SELECT kospic.company_name,kospic.now_price
      FROM(SELECT stock.company_name,stock.date_yyyy_mm_dd,stock.time,stock.now_price
            FROM stock
            WHERE stock.market == '코스피') AS kospic
      WHERE kospic.date_yyyy_mm_dd == '2019-12-10' and kospic.time == '09:30'
      ORDER BY kospic.now_price ASC LIMIT 5) AS worst_top5
```

Filter columns ✕

best_top5.company_name

미래산업

키위미디어그룹

서울식품

이아이디

KR모터스

SAVE TO HDFS

DOWNLOAD AS CSV

⑤ 코스닥의 경우도, 위와 같이 현재가가 가장 높은 기업과, 낮은 기업 5군데를 찾아서 csv로 저장한 뒤, 해당 csv 파일을 HIVE에서 직접 테이블로 만든다.

```
SELECT best_top5.company_name
FROM(SELECT koscac.company_name,koscac.now_price
      FROM(SELECT stock.company_name,stock.date_yyyy_mm_dd,stock.time,stock.now_price
            FROM stock
            WHERE stock.market == '코스닥') AS koscac
      WHERE koscac.date_yyyy_mm_dd == '2019-12-10' and koscac.time == '09:30'
      ORDER BY koscac.now_price DESC LIMIT 5) AS best_top5
```

best_top5.company_name

휴젤

메디톡스

펄어비스

SK머티리얼즈

CJ ENM

SAVE TO HDFS

DOWNLOAD AS CSV

```
SELECT worst_top5.company_name
FROM(SELECT koscac.company_name,koscac.now_price
      FROM(SELECT stock.company_name,stock.date_yyyy_mm_dd,stock.time,stock.now_price
            FROM stock
            WHERE stock.market == '코스닥') AS koscac
      WHERE koscac.date_yyyy_mm_dd == '2019-12-10' and koscac.time == '09:30'
      ORDER BY koscac.now_price ASC LIMIT 5) AS worst_top5
```

worst_top5.company_name

모다

이에스브이

솔고바이오

이화전기





퓨전

SAVE TO HDFS

DOWNLOAD AS CSV

⑥ 저장한 파일들을 로컬에서 바로 hive에 올려서 각각 테이블로 만들어준다. csv로 저장한 뒤 테이블을 형성하는 이유는 hive에는 where in 구조의 문법을 사용 할 수 없기 때문이다. where in 문법을 대체하기 위해서는 hive에서는 left semi join이라는 문법을 사용해야 한다. 하지만 본 문법을 사용하기 위해서는 미리 만들어진 테이블이 있어야만 한다. 그렇기 때문에 각각 csv로 저장한 뒤 해당 파일을 테이블로 형성하였다.

<테이블 만든 결과>

 kospi_best_top5
 kospi_worst_top5
 kosdac_best_top5
 kosdac_worst_top5
 stock

5. 데이터 분석 결과

1) 코스피 시장에서 처음으로 수집된 데이터의 현재가가 가장 높은 5개의 기업들의 일 평균 현재가 분석

```
SELECT stock.company_name,stock.date_yyyy_mm_dd,AVG(stock.now_price)
FROM stock LEFT SEMI JOIN kospi_best_top5 on (stock.company_name = kospi_best_top5.best_company_name)
GROUP BY stock.company_name,stock.date_yyyy_mm_dd
```

LG생활건강	2019-12-10	1265357.142857143
LG생활건강	2019-12-11	1251142.857142857
LG생활건강	2019-12-12	1264285.7142857143
LG생활건강	2019-12-13	1244428.5714285714
LG생활건강우	2019-12-10	734642.8571428572
LG생활건강우	2019-12-11	735214.2857142857
LG생활건강우	2019-12-12	748571.4285714285
LG생활건강우	2019-12-13	743857.1428571428
엔씨소프트	2019-12-10	538785.7142857143
엔씨소프트	2019-12-11	544785.7142857143
엔씨소프트	2019-12-12	539214.2857142857
엔씨소프트	2019-12-13	536071.4285714285
영풍	2019-12-10	628642.8571428572
영풍	2019-12-11	632428.5714285715
영풍	2019-12-12	655142.8571428572
영풍	2019-12-13	639500.0
태광산업	2019-12-10	983071.4285714285
태광산업	2019-12-11	986571.4285714285
태광산업	2019-12-12	989642.8571428572
태광산업	2019-12-13	987000.0

=> 분석 결과

‘현재가’를 기준으로 할 때, 코스피 시장에서는 LG생활건강이 강세를 보이고 있다. 타 기업들과 비교해도, LG생활건강의 현재가는 높으며, 5개의 기업 중 1위와 2위가 LG생활건강임을 알 수 있다.

각 기업들은 현재가가 꾸준히 상승하거나 꾸준히 하락하는 기업이 없다. 지속적으로 현재가를 비슷한 범위에서 유지하고 있다. 이 점은 성장이나 하락하는 기업이 아니라, 어느 정도 안정궤도로 접어든 기업으로 추측된다. 이미 현재가가 높기 때문에 발전보다는 안정을 추구하는 것으로 보인다.

2) 코스피 시장에서 처음으로 수집된 데이터의 현재가가 가장 낮은 5개의 기업들의 일 평균 현재가 분석

```
SELECT stock.company_name,stock.date_yyyy_mm_dd,AVG(stock.now_price)
FROM stock LEFT SEMI JOIN kospi_worst_top5 on (stock.company_name = kospi_worst_top5.worst_company_name)
GROUP BY stock.company_name,stock.date_yyyy_mm_dd
```

KR모터스	2019-12-10	270.64285714285717
KR모터스	2019-12-11	273.2857142857143
KR모터스	2019-12-12	283.7142857142857
KR모터스	2019-12-13	295.42857142857144
미래산업	2019-12-10	143.64285714285714
미래산업	2019-12-11	143.07142857142858
미래산업	2019-12-12	143.57142857142858
미래산업	2019-12-13	151.28571428571428
서울식품	2019-12-10	159.28571428571428
서울식품	2019-12-11	161.07142857142858
서울식품	2019-12-12	163.5
서울식품	2019-12-13	169.85714285714286
이아이디	2019-12-10	226.14285714285714
이아이디	2019-12-11	228.5
이아이디	2019-12-12	232.21428571428572
이아이디	2019-12-13	245.71428571428572
키위미디어그룹	2019-12-10	150.0
키위미디어그룹	2019-12-11	150.0
키위미디어그룹	2019-12-12	150.0
키위미디어그룹	2019-12-13	150.0

=> 분석 결과

전반적으로 현재가가 낮은 기업들은 증가하는 추세가 나타난다. 이는 현재가가 낮은 기업들은 성장하고 있는 기업임을 보여 준다. '키위미디어 그룹'을 제외하고는 첫날에 비해 모든 기업들이 현재가가 증가했음을 알 수 있다.

3) 코스닥 시장에서 처음으로 수집된 데이터의 현재가가 가장 높은 5개의 기업들의 일 평균 현재가 분석

```
SELECT stock.company_name,stock.date_yyyy_mm_dd,AVG(stock.now_price)
FROM stock LEFT SEMI JOIN kosdac_best_top5 on (stock.company_name = kosdac_best_top5.best_company_name)
GROUP BY stock.company_name,stock.date_yyyy_mm_dd
```

CJ ENM	2019-12-10	149678.57142857142
CJ ENM	2019-12-11	152285.7142857143
CJ ENM	2019-12-12	152578.57142857142
CJ ENM	2019-12-13	152921.42857142858
SK머티리얼즈	2019-12-10	167521.42857142858
SK머티리얼즈	2019-12-11	169214.2857142857
SK머티리얼즈	2019-12-12	172792.85714285713
SK머티리얼즈	2019-12-13	180221.42857142858
메디톡스	2019-12-10	291157.14285714284
메디톡스	2019-12-11	290764.28571428574
메디톡스	2019-12-12	293685.71428571426
메디톡스	2019-12-13	298664.28571428574
필터비스	2019-12-10	187035.7142857143
필터비스	2019-12-11	185635.7142857143
필터비스	2019-12-12	189500.0
필터비스	2019-12-13	192785.7142857143
휴젤	2019-12-10	359835.71428571426
휴젤	2019-12-11	369085.71428571426
휴젤	2019-12-12	368428.5714285714
휴젤	2019-12-13	382785.71428571426

=> 분석결과

코스닥에서 현재가가 높은 기업들 5개 중 대부분은 현재가가 상승하고 있다. 모든 기업들이 모두 꾸준한 상승을 하는 것은 아니지만, 일시적으로 주춤했다가 다시 현재가가 올라가는 양상을 보여주고 있다.

4) 코스닥 시장에서 처음으로 수집된 데이터의 현재가가 가장 낮은 5개의 기업들의 일 평균 현재가 분석

```
SELECT stock.company_name, stock.date_yyyy_mm_dd, AVG(stock.now_price)
FROM stock LEFT SEMI JOIN kosdac_worst_top5 on (stock.company_name = kosdac_worst_top5.worst_company_name)
GROUP BY stock.company_name, stock.date_yyyy_mm_dd
```

모다	2019-12-10	155.0
모다	2019-12-11	155.0
모다	2019-12-12	155.0
모다	2019-12-13	155.0
솔고바이오	2019-12-10	199.78571428571428
솔고바이오	2019-12-11	199.28571428571428
솔고바이오	2019-12-12	200.14285714285714
솔고바이오	2019-12-13	199.64285714285714
이에스브이	2019-12-10	172.85714285714286
이에스브이	2019-12-11	172.28571428571428
이에스브이	2019-12-12	174.78571428571428
이에스브이	2019-12-13	181.07142857142858
이화전기	2019-12-10	209.0
이화전기	2019-12-11	208.85714285714286
이화전기	2019-12-12	210.85714285714286
이화전기	2019-12-13	215.14285714285714
퓨전	2019-12-10	221.0
퓨전	2019-12-11	221.0
퓨전	2019-12-12	221.0
퓨전	2019-12-13	221.0

=> 분석결과

본 5개의 기업 중 '이에스브이'를 제외하고 꾸준한 상승이나 꾸준한 하락보다는 어느 특정 범위 내에서 움직이고 있다. 이는 해당 기업들이 성장하고 있지 않다는 것을 보여준다. 만약 현재가가 높은 기업들이라면 안정적인 운영이라고 할 수 있지만, 현재가가 낮은 기업들이기 때문에 정체되었다고 해석을 할 수도 있을 것 같다.

6. 한계 및 추가적인 확장 가능성

본 프로젝트의 한계점은 데이터를 수집 할 수 있는 기간이 매우 짧았다는 것이다. 4일간의 현재가의 변화로는 시계열 분석을 통해 추세를 알아볼 수 없다. 물론 매일 30분 단위로 크롤링을 했지만, 주식 시장을 분석하기에는 짧은 시간이라고 생각한다. 두 번째로, 단순한 지표만을 통해 기업의 가치를 평가하기에는 역부족이다. 현재가는 실시간으로 변화하는 가격이기 때문에 기업의 등락을 실시간으로 평가하는데 지표가 될 수는 있지만, 해당 기업이 보유한 다양한 자원을 통합적으로 평가해야 해당 기업의 가치를 진정으로 평가 할 수 있다고 생각한다.

그럼에도 불구하고 본 프로젝트를 확장한다면, 좋은 프로젝트가 될 것임은 분명하다. 먼저, 데이터를 수집하는 기간을 늘리고, 30분 간격이 아닌, 최소 5분간격으로 주식시장의 다양한 지표를 함께 수집한다면, 폭넓은 빅데이터로 발전 할 수 있을 것으로 생각한다.

두 번째로 기업 신용평가 모델과 결합하면 그 가치가 높아질 것이라고 생각한다. 기업들의 신용정보를 바탕으로 해당 기업들의 신용을 평가하고 등급을 매기는 신용평가기업들이 존재한다. 만약 본 프로젝트가 확장되어 빅데이터 규모의 데이터가 축적된다면, 주식 시장의 다양한 지표와 신용평가기업들이 가진 정보를 결합하여, 추후 기업을 평가하는 데 더 정확한 평가를 할 수 있을 것이라고 생각한다.

<출처>

네이버 업종별 시가총액

https://finance.naver.com/sise/sise_market_sum.nhn?sosok=0&page=1

https://finance.naver.com/sise/sise_market_sum.nhn?sosok=1&page=1

아나콘다 로고

<https://bradbury.tistory.com/60>

파이썬 로고

<https://ko.wikipedia.org/wiki/%ED%8C%8C%EC%9D%BC:Python-logo-notext.svg>

스케줄러 로고

<http://www.techwork.dk/servers/how-to-schedule-a-reboot-of-windows-server/attachment/task-scheduler-icon>

엑셀 로고

<https://edmond.mpg.de/imeji/collection/MTBbtaulvOibV4q/item/EYrnkRP7fEeRvH7C?q=&fq=&filter=&pos=12>

CSV 로고

https://www.iconfinder.com/icons/3381493/csv_file_icon_plano_icon