제목: 금융 재무 데이터 크롤링을 통한 pandas 분석

학과: 경영학부, 학년: 4학년, 학번: 2016508113, 이름: 윤양현

**과제 개요: (어떤 내용을 할 것인지를 간략하게 요약합니다)**

네이버 금융, fnguide, DART 전자공시 API 중 재무 데이터를 크롤링하여 투자에 있어 필요한 정보들을 정리합니다(시가총액, 분기별 순이익, 부채비율, PER, 산업종, etc…). 이 크롤링한 정보들을 보기 쉽게 pandas 라이브러리를 이용합니다.

**문제 정의와 과제의 필요성 (왜 이 과제를 하려고 하는지, 어떤 불편한 점을 개선하기 위한 것인지, 이것을 이용하면 어떻게 좋은지를 개괄적으로 설명합니다)**

네이버 금융을 살펴보면 각 종목에 대한 정보들은 설명이 잘 되어 있어 주식 초보자가 보기에도 쉽게 이해할 수 있지만, 기업 대 기업의 비교라던지 혹은 산업별 비교에 대해서는 상당히 잘 처리된 정보들을 찾아보기 쉽지 않습니다. 따라서 이 데이터 크롤링을 통하여 비교하고 싶은 기업의 시가총액, 혹은 당기순이익, 영업이익 등등 여러가지를 비교분석할 수 있습니다.

이 데이터들을 잘 이용하면 상대적으로 우위에 있는 가치주들을 발굴해 내기 쉽고, 이른바 퀀트투자에 도움이 될 만한 정보들을 구성해줄 것입니다.

**개발 개요 (과제의 구체적인 내용을 서술합니다. 필요한 기술과 처리 절차 등을 가능하면 구체적으로 기술합니다.)**

웹 크롤링을 하기 위한 라이브러리인 beautifulsoup 와 selenium 라이브러리의 사용이 필수입니다.

금융 데이터 수집을 편리하게 해주는 financedatareader의 라이브러리, 혹은 DART 전자공시 API에서 제공하는 OpenDartReader 라이브러리에서 데이터를 끌어올 수 있습니다.

코스피, 코스닥 기업을 합쳐서 대략 3천개의 재무데이터를 크롤링한 자료들을 쉽게 구분할 수 있게 pandas 라이브러리를 사용하여 볼 수 있도록 정리할 것입니다. 이 자료들은 엑셀로도 저장하여 볼 수 있게끔 개발할 예정입니다.

데이터를 수집하는 데 for문과 if문, try/except문, 간단한 함수들을 만들어서 크롤링 자동화 과정을 거친 후, 자료를 만들 것입니다.

**개발 계획(개발 일정을 작성합니다)**

5/10~5/16 전에 만들었던 크롤링 과제와 수업, 구글링을 통하여 selenium, beautifulsoup 라이브러리를 다시 공부하며 기초적인 토대 만들기

5/16~5/23 for문, try/except 구문을 이용하여 각 기업에 해당하는 재무정보들을 네이버 금융에서 크롤링하여 가져오기(OpenDartReader 혹은 financedatareader를 이용할 경우 어느 정도 구문이 생략될 수 있습니다)

5/23~과제 제출(6월초) 최종정리 및 오류 수정 단계

**개발 내용(지금은 채우지 않습니다. 기말 과제 제출시 실제 개발 내용을 자세히 설명합니다.)**

개발 과정을 코드와 같이 보시면 이해하기 편합니다.

**1. 데이터 전처리 과정**

데이터를 전체적으로 긁어모아 KOSPI, KOSDAQ과는 상관없이 오로지 숫자만을 가지고 비교하게 만들고 싶었습니다. 따라서 파이썬 라이브러리에 있는 FinanceDataReader 패키지를 이용하여 각 종목의 간단한 정보 리스트를 받았습니다. 그리고 FDR 데이터 + 네이버 금융 크롤링을 통해 비교분석할 수 있는 정보들을 만들어내는 것을 목표로 하였습니다.

df\_krx = fdr.StockListing('KRX')

다음과 같이 변수를 df\_krx로 지정하여 한국거래소의 상장 종목 전체에 대한 정보를 가져옵니다. 각 정보리스트는 다음과 같습니다. – 리스트에 대한 정보는 **krx\_total.xlsx** 에서 확인할 수 있습니다.

| **Symbol** | **Market** | **Name** | **Sector** | **Industry** | **ListingDate** | **SettleMonth** | **Representative** | **HomePage** | **Region** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

이 정보 리스트 중에서 필요 없는 정보들인 ListingData, SettleMonth, HomePage, Region 열은 삭제하였습니다.

df\_krx\_new = df\_krx.drop(['ListingDate','SettleMonth','HomePage','Region'],axis=1)

데이터를 제거하는 drop() 함수 안에는 axis= 라는 파라미터가 있습니다. axis=0으로 설정하면 행의 축을 제거하고, axis=1로 설정하면 열의 축을 삭제합니다. 삭제된 정보를 df\_krx\_new 변수로 지정하였습니다.

또한, 열의 축을 삭제했어도 정보가 없는 콜, 풋옵션, ETF, 우선주들이 같이 정리되어 있기에 이에 대해서 확인을 하고 이 또한 제거하였습니다.

df\_krx\_new.isnull() – 공백 행 확인하기

df\_krx\_new = df\_krx\_new.dropna() – 결측치 값 지우기

이렇게 정리된 데이터들에 대한 정보는 **krx\_total\_new.xlsx** 에서 확인할 수 있습니다.

**2. 웹크롤링**

웹크롤링을 하기 위해서 웹페이지를 확인할 필요가 있습니다. 네이버 금융의 삼성전자 주식의 정보를 확인해 보면 페이지 주소가 ‘https://finance.naver.com/item/main.nhn?code=005930’로 되어있고, SK하이닉스의 페이지 주소는‘https://finance.naver.com/item/main.nhn?code=000660’로 구성되어 있습니다.

따라서 다른 종목 페이지 주소는 ‘https://finance.naver.com/item/main.nhn?code=해당종목코드’ 라는 것을 알 수 있습니다.

code\_symbol = df\_krx\_new['Symbol']

for i in code\_symbol:

    print(i)

반복적으로 수행하기 위해서 for문을 이용해야 합니다. for문의 활용을 위해 종목코드를 추출해 보았습니다.

# 최대 크기 윈도우로 시작

options = webdriver.ChromeOptions()

options.add\_argument("--start-maximized")

driver = webdriver.Chrome(options=options,executable\_path=r'C:\Users\yangh\chromedriver.exe')

웹 크롤링을 하기 위해서 필요한 기본적인 세팅입니다. Selenium 모듈을 사용하며, 웹드라이버가 최신화 되어 있어야 오류없이 진행할 수 있습니다. 옵션은 다음과 같으나, 웹드라이버를 실행하기 위한 경로는 변경해야 다른 컴퓨터 환경에서 실행이 가능합니다.

# 1. 데이터 생성, 데이터프레임에 들어갈 열 선택

columnlist=['Symbol','Market','Name','Sector','Industry','Representative','Price','Market Cap','Revenue(2020/12)','Operating Income(2020/12)','Net Income(2020/12)','Volume','ROE','PER','PBR','D/E Ratio','Dividend Yield(%)','52W High|Low']

df = pd.DataFrame(columns=columnlist)

판다스를 사용하고, 각 열에 들어갈 제목들을 데이터 프레임에 넣습니다.

# 2. 기존 한국거래소 상장종목 + 네이버 금융 웹크롤링

for i in code\_symbol: # 전체가 2428개가 있음

    for row\_n in range(0,2428): # df\_krx\_new에 있는 값에서 가져오는 것이기 때문에 코드의 일치시켜야 함

        if df\_krx\_new.iat[row\_n,0] == str(i):

            symbol = df\_krx\_new.iat[row\_n,0] # n행,1열의 값 가져오기

            market = df\_krx\_new.iat[row\_n,1] # n행,2열의 값 가져오기

            name = df\_krx\_new.iat[row\_n,2] # n행,3열의 값 가져오기

            sector = df\_krx\_new.iat[row\_n,3] # n행,4열의 값 가져오기

            industry = df\_krx\_new.iat[row\_n,4] # n행,5열의 값 가져오기

            representive = df\_krx\_new.iat[row\_n,5] # n행,6열의 값 가져오기

    url = 'https://finance.naver.com/item/main.nhn?code=' + str(i) # url은 string 자료구조로 되어 있기에 str()로 변환

    driver.get(url)

    html = driver.page\_source

    soup = BeautifulSoup(html,'lxml')

기존에 넣었던 종목코드를 반복합니다. 반복하면서, 변수로 지정해 두었던 df\_krx\_new 데이터의 행을 같이 반복합니다. 만약 종목코드(1행1열)의 값이 현재 반복하고 있는 종목코드와 같다면 symbol 변수에 값을 넣습니다. 종목코드의 값이 같다면 1행2열의 값은 market 변수에 넣습니다… 이렇게 df\_krx\_new 데이터 종목 수만큼 반복시키면서 웹크롤링 작업을 같이 실시합니다.

웹크롤링 과정은 현재 페이지를 반복하면서 데이터를 추출해옵니다. 이에 필요한 라이브러리는 beautifulsoup와 selenium webdriver입니다. url을 지속적으로 변화하려면 종목코드의 변화만 필요하기에 string 구조로 변환시켜 url변수에 str(i)을 넣어 반복적으로 현 웹페이지의 자료를 추출할 수 있도록 합니다. 현 웹페이지 자료를 soup 변수에 넣습니다.

# 3. 데이터 수집 - 줄바꿈,탭처리 제거/ 자료가 없어 수집을 멈추는 것을 막기 위해 try/except 구문 사용

    try:

        price = soup.select\_one('#chart\_area > div.rate\_info > div > p.no\_today').text

        price = price.replace('\n','')

    except:

        price = ''

    try:

        market\_cap = soup.select\_one('#tab\_con1 > div.first > table > tbody > tr.strong > td').text

        market\_cap = market\_cap.replace('\n','').replace('\t','')

    except:

        market\_cap = ''

데이터를 수집해오기 위해 select.one() 함수를 이용하여 주가(price)에 해당하는 정보를 가져옵니다. Css selector를 카피해서 텍스트 형태로 가져왔으며, 가져오고 나서는 불필요한 텍스트를 제거하기 위해 줄바꿈처리와 탭처리를 제거하였습니다. 해당 데이터 프레임에 총 12개의 열이 들어가 있었으니, 12개의 열 모두 이와 같은 방식으로 처리를 해줍니다. 하지만, 예외적으로 크롤링해오는 위치에 정보가 없다면 오류를 일으키는데 이와 같은 오류는 try/except 구문을 통해서 오류가 일어난 부분에는 공백처리를 하였습니다.

# 4. 데이터 저장 및 출력

    data = [symbol,market,name,sector,industry,representive,price,market\_cap,revenue,operating\_income,net\_income,volume,roe,per,pbr,de\_ratio,dividend\_yield,week52\_high\_low]

    df = df.append(pd.Series(data, index=df.columns), ignore\_index=True)

driver.close() # 데이터 수집이 완료되면 웹드라이버를 끝내기

df

수집해온 18개(6개-FDR데이터, 12개-네이버 금융 웹크롤링) 데이터를 변수에 넣고, 이를 새로운 데이터 프레임으로 만들기 위해 append() 함수를 이용합니다. Data 변수를 그대로 붙이면 오류가 나기 때문에 이를 인덱스를 열로 구성시킨 시리즈 객체로 변환시켜 df 변수에 넣었습니다. 이후 결번 오류를 제거하기 위해 파라미터를 ignore\_index=True로 인덱스 무시를 하면 18개의 열을 가진 데이터 프레임을 얻을 수 있습니다. -정리된 결과 데이터는 **krx\_total\_refined.xlsx**에서 확인해 볼 수 있습니다.

**3. 데이터 자료 구조 변환**

시가총액의 비교구분, 수익의 비교, PER 비교 etc… 을 하기 위해서는 객체 데이터가 아닌 숫자 데이터(int,float) 형식으로 바꾸어야 이에 대한 대소 비교를 할 수 있습니다. 따라서 비교 분석하기 위한 데이터를 숫자형 자료구조로 바꿀 필요성이 있습니다.

df.dtypes

데이터의 자료 구조 확인을 위해 데이터 타입을 확인하는 dtypes함수를 씁니다. 18개의 데이터는 object(객체) 형식으로 구분되어 있는 것을 확인할 수 있습니다.

# 콤마 지우는 함수 만들기

def remove\_comma(x):

    return x.replace(',','')

rm\_cma\_columns = ['Price','Market Cap','Revenue(2020/12)','Operating Income(2020/12)','Net Income(2020/12)','Volume','ROE','PER','D/E Ratio','Dividend Yield(%)']

for a in rm\_cma\_columns:

    df[str(a)] = df[str(a)].apply(remove\_comma)

콤마는 사람들이 보기 쉽게 구분할 수 있도록 한 구분자입니다. 이는 숫자 자료에 적용되지 않기 때문에 쉼표를 숫자형으로 바꾸려는 모든 열에 대해서 지워야 합니다. 쉼표를 숫자열로 바꾸기 위한 작업의 코드는 위와 같습니다.

# 조,억 단위 삭제

def remove\_m(x):

    return x.replace('조','').replace('억원','')

df['Market Cap'] = df['Market Cap'].apply(remove\_m)

df

또한, 숫자 자료형에는 ‘조’, ‘억원’ 과 같은 문자형 구조는 들어가지 않습니다. 이를 제거하기 위해서 조, 억단위를 없애는 함수를 만들고 이를 적용시켜 df 변수에 넣었습니다. – 조단위까지 정리한 데이터는 **krx\_total\_refined\_조단위삭제.xlsx**에서 확인할 수 있습니다.

# 조단위 삭제

# df.to\_excel(r'C:\Users\yangh\OneDrive\바탕 화면\대학자료\2021-1\컴퓨터언어\krx\_total\_refined\_조단위삭제.xlsx',index=False)

df = pd.read\_excel(r'C:\Users\yangh\OneDrive\바탕 화면\대학자료\2021-1\컴퓨터언어\krx\_total\_refined\_조단위삭제.xlsx')

df

df.dtypes

엑셀로 정리한 결과를 불러오고, 이에 대한 데이터 구조를 확인하면 엑셀이 숫자로만 이루어진 열은 자동으로 숫자자료형으로 인식하고 바꿔줍니다. 하지만 오류가 나는 열에 대해서는 그대로 놔두게 됩니다.

# 공백 오류 및 비숫자형 오류 숫자형 열로 변경하기

to\_numeric\_columns = ['Revenue(2020/12)','Operating Income(2020/12)','Net Income(2020/12)','ROE','PER','PBR']

for a in to\_numeric\_columns:

    df[str(a)] = df[str(a)].apply(pd.to\_numeric, errors = 'coerce')

df

df.dtypes

오류가 난 열들은 다음 to\_numeric\_columns와 같습니다. 오류가 난 원인은 공백이 나 있던지, 웹크롤링 과정중 숫자가 아닌 ‘-‘와 같은 문자 자료를 가져왔기 때문입니다. 따라서 이에 대해서 apply 함수를 이용하며 파라미터를 errors = ‘coerce’를 이용합니다. 이럴 경우 value error가 나며 문자열을 숫자로 파싱할 수 없는 오류를 해결하며, 문자가 포함되어 있는 경우에는 강제적으로 ‘NaN’으로 처리합니다. 그 후 최종 자료를 확인해 보면 숫자형 자료로 바뀐 것을 확인할 수 있습니다. – 최종 결과 자료는 **krx\_total\_final**에서 확인할 수 있습니다.

**결과(지금은 채우지 않습니다. 기말 과제 제출시 채웁니다)**

정리된 자료를 통해서, 많은 여러가지 투자 지표들을 확인해 볼 수 있습니다.

df.sort\_values(by='Market Cap',ascending=True)

df.sort\_values(by='Market Cap',ascending=False)

시가총액을 기준으로 오름차순, 내림차순 정렬하며 확인해 볼 수 있습니다.

df.sort\_values(by='PER',ascending=True)

df.sort\_values(by='PER',ascending=False)

PER도 마찬가지로 오름차순 내림차순으로 정렬하며 확인할 수 있습니다.

**결론(지금은 채우지 않습니다. 기말 과제 제출시 채웁니다)**

엑셀을 통하여 퀀트투자 포트폴리오를 만들고 싶다면 여러 복합적인 요소를 고려하여 투자 포트폴리오를 선정할 수 있습니다.

Ex1) 시가총액 상위 20% 저PER 상위 20개 종목

Ex2) 시가총액 하위 20% 저PER 상위 20개 종목(적자 종목제외)

Ex3) 시가총액 상위 20% 저PBR 상위 20개 종목