KQR Final Project 3조

중소형주와 보조지표 활용, 딥러닝 퀀트 전략

박민서, 염제윤, 이화원, 허지원

목차

1. 전략소개

-종목선정(중소형주, 어닝서프라이즈)
-보조지표(MACD, RSI)

전략소개

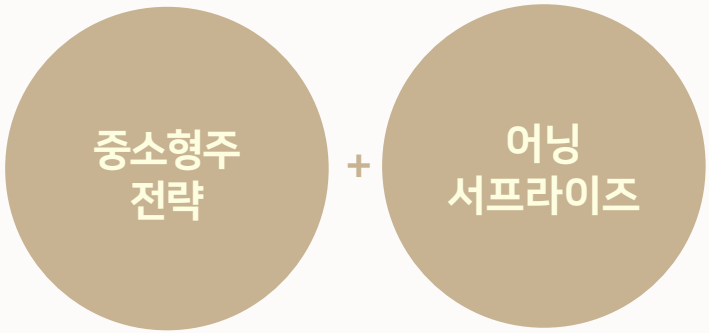
Track1.
종목선정

10개

Track2.
매수/매도

전략소개 Track1. 종목선정

기업 실적발표 기준



약 3개월 단위
리밸런싱

중소형주
(200개 종목)
+
어닝서프라이즈
(상위 10개 종목)

=중소형 서프라이즈 10개 종목

PART 1

전략소개

Track1.

종목선정

중소형
서프라이즈
10개

보조지표 이용한
매수 및 매도
타이밍 설정

Track2.

매수/매도

MACD

RSI

전략소개 Track2. 매수/매도

보조지표

주가의 일봉, 거래량 등 데이터 기반,
매수 매도 타이밍 확보에 도움

1. 추세지표, 가격지표(주가 방향성)

:MACD, ADX, 가격 이평선, 일목균형표 등

2. 모멘텀 지표, 시장강도 지표(주가 방향성 강도)

;이격도, RSI, 투자심리선, OBV 등

전략소개 Track2. 매수/매도

보조지표

주가의 일봉, 거래량 등 데이터 기반,
매수 매도 타이밍 확보에 도움

1. 추세지표, 가격지표(주가 방향성)

:MACD, ADX, 가격 이평선, 일목균형표 등

2. 모멘텀 지표, 시장강도 지표(주가 방향성 강도)

;이격도, RSI, 투자심리선, OBV 등

전략소개 Track2. 매수/매도

보조지표

주가의 일봉, 거래량 등 데이터 기반,

이후 백테스팅 진행하며

최대 CAGR(연평균성장률)

탐색할 예정

2. 모멘텀 지표, 시장강도 지표(주가 방향성 강도)

(지표 추가, 조건 변경 등)

2. 코드구현

-중소형주 종목 선정
-MACD, RSI 전략

1)중소형주 종목 선정

**시가총액 100위-299위

코드 중소형주 종목 선정

입력변수(4)

1. PER데이터
2. EPS증가율
3. 저변동성 데이터
4. 기관매수비율 (거래량/기관순매수)

2017년 1월 1일-2021년 3월 30일

코드 중소형주 종목 선정

1. 데이터 크롤링 (for 중소형주 리스트)

데이터 크롤링

```
## 모듈 가져오기

import pandas as pd
import requests
import time
import html5lib
```

```
import os

os.getcwd()
```

전 종목 코드 구해오기

```
# pykrx를 이용하여 전종목코드와 기업명을 가져옴

from pykrx import stock
tickers = stock.get_market_ticker_list(market='ALL') #pykrx의 stock클래스의 종목코드 불러오기(전체)
Name=[]
for ticker in tickers:
        Name.append(stock.get_market_ticker_name(ticker)) #반복문을 이용해 모든 종목 코드들을 리스트안에 전부 할당

#데이터 프레임 형태로 만들기
code_data=pd.DataFrame()
code_data['종목코드']=tickers
code_data['기업명'] = Name
code_data['종목코드']='A'+code_data['종목코드']
code_data
```

PART 2

코드 중소형주 종목 선정

전체 종목

	종목코드	기업명
0	A060310	3S
1	A095570	AJ네트웍스
2	A006840	AK홀딩스
3	A054620	APS홀딩스
4	A265520	AP시스템
…	…	…
2568	A000547	흥국화재2우B
2569	A000545	흥국화재우
2570	A003280	흥아해운
2571	A037440	희림
2572	A238490	힘스

전체 종목 시가총액 데이터 크롤링

데이터프레임 형태 바꾸기 함수 -- 크롤링한 데이터프레임의 형식을 바꿔줌

```
def change_df(firm_code, dataframe): #종목코드와 해당 데이터 프레임을 매개변수로 가짐

    for num, col in enumerate(dataframe.columns):
        temp_df = pd.DataFrame({firm_code : dataframe[col]})
        temp_df = temp_df.T
        temp_df.columns = [[col]*len(dataframe), temp_df.columns]

        if num == 0:
            total_df = temp_df
        else:
            total_df = pd.merge(total_df, temp_df, how='outer', left_index=True, right_index=True)

    return total_df
```

시가총액 데이터 가져오기

중형주 : 시가총액 101위에서 300위 사이를 뜻한다

```
#시가총액 데이터를 가져오는 함수

def make_allstock_dataframe(firm_code): #종목명을 매개변수로 사용
    allstock_url = 'https://comp.fnguide.com/SVO2/asp/SVD_Main.asp?pGB=1&gicode=' + firm_code #url을 할당
    allstock_page = requests.get(allstock_url) #request한 해당 url의 페이지 데이터를 호출
    allstock_tables = pd.read_html(allstock_page.text) #html파일을 텍스트 파일 형태로 가져와 데이터프레임으로 만듦
    temp_df = allstock_tables[0] #해당 내용이 들어있는 테이블 위치

    temp_df = temp_df.set_index(temp_df.columns[0]) #데이터테이블의 0번째 열을 인덱스로 사용
    temp_df = temp_df.loc[['시가총액(보통주,억원)']] # '시가총액' 인덱스의 데이터를 "데이터프레임 형식으로 가져옴" using [[]] (겹괄호)
    temp_df.index = ['시가총액'] #'시가총액'이라는 이름으로 인덱스 이름 변경
    temp_df = temp_df[[1]] # 다시말하지만 겹괄호를 사용하면 데이터프레임 형식으로 출력
    return temp_df
```

```
#시가총액 가져오기 -- PYKRX로 가져온 종목코드를 이용함

for num, code in enumerate(code_data['종목코드']): #코스피 시가총액구하기 # enumerate로 종목코드의 순서와 이름을 둘다 가져올수 있음
    try:                                                  # ex) (1, A000000)
        print(num, code)   #해당 순번과 종목코드 출력
        time.sleep(1)      #2초간 텀을 둠

        try:
            allstock_df = make_allstock_dataframe(code) #시가총액데이터를 가져오기
        except requests.exceptions.Timeout: #예외상황 -- 시간초과
            time.sleep(60)
            allstock_df = make_allstock_dataframe(code) #다시 시도

        allstock_df_changed = change_df(code, allstock_df) #데이터 프레임 형태 바꾸기(함수)

        if num == 0 :
            total_allstock = allstock_df_changed
        else:
            total_allstock = pd.concat([total_allstock, allstock_df_changed])
    except ValueError:
        continue
    except KeyError:
        continue
```

코드 중소형주 종목 선정

전체 시가총액 데이터

2553	032860
2554	200670
2555	212310
2556	079980
2557	065510
2558	005010
2559	243070
2560	084110
2561	263920
2562	145020
2563	069260
2564	024060
2565	010240
2566	189980
2567	000540
2568	000547
2569	000545
2570	003280
2571	037440
2572	238490

0 A060310
1 A095570
2 A006840
3 A054620
4 A265520
5 A211270
6 A027410
7 A282330
8 A138930
9 A001460
10 A001465
…

중형주 리스트 만들기(시가총액 100-299위, 200개)

```
#시가총액 데이터저장
total_allstock.to_csv('시가총액.csv')

#중형주 리스트 가져오기

mediumlist = pd.read_csv("시가총액.csv")

mediumlist1 = pd.DataFrame()

mediumlist1['종목코드'] = mediumlist.iloc[1:,0]
mediumlist1['시가총액'] = mediumlist.iloc[1:,1].apply(pd.to_numeric)

mediumlist = mediumlist1

mediumlist = mediumlist.sort_values(by = ['시가총액'],ascending=False) #내림차순 정렬

mediumlist = mediumlist[100:300] #100위 부터 299위까지

mediumlist = mediumlist.reset_index(drop = True)
```

코드 중소형주 종목 선정

중형주리스트.csv

순위	종목코드	시가총액
0	A196170	35093.0
1	A010620	35029.0
2	A036460	34571.0
3	A036490	34280.0
4	A039490	34048.0
…	…	…
195	A010050	9135.0
196	A268280	9104.0
197	A280360	9080.0
198	A082640	9052.0
199	A192400	8962.0

코드 중소형주 종목 선정

입력변수(4)

1. PER데이터
2. EPS증가율
3. 저변동성 데이터
4. 기관매수비율(기관순매수/거래량)

코드 중소형주 종목 선정

1. PER데이터

PER 데이터 가져오기 - kqr 데이터수집 코드 참고

```
# 투자지표 데이터프레임을 만드는 함수 (데이터 수집하기_재무 데이터 구해오기)

def make_invest_dataframe(firm_code):
    invest_url = 'https://comp.fnguide.com/SVO2/asp/SVD_Invest.asp?pGB=1&cID=&MenuYn=Y
    invest_page = requests.get(invest_url)
    invest_tables = pd.read_html(invest_page.text)
    temp_df = invest_tables[1]

    temp_df = temp_df.set_index(temp_df.columns[0])
    temp_df = temp_df.loc[['PER계산에 참여한 계정 펼치기']]
    temp_df.index = ['PER']
    return temp_df
```

```
for num, code in enumerate(mediumlist['종목코드']): #중형주 per 가져오기
    try:
        print(num, code)
        time.sleep(1)
        try:
            invest_df = make_invest_dataframe(code)
        except requests.exceptions.Timeout:
            time.sleep(60)
            invest_df = make_invest_dataframe(code)
        invest_df_changed = change_df(code, invest_df)
        if num == 0 :
            total_invest = invest_df_changed
        else:
            total_invest = pd.concat([total_invest, invest_df_changed])
    except ValueError:
        continue
    except KeyError:
        continue
```

중형주 종목코드 이용해

PER 정보 불러오기

```
for num, code in enumerate(mediumlist['종목코드']): #중형주 per 가져오기
```

```
total_invest.to_csv('PER.csv')  #per 데이터 저장
```

코드 중소형주 종목 선정

2. EPS 증가율 데이터

EPS 증가율 데이터 가져오기

```
#Eps 증가율 가져오기 함수

def make_fr_dataframe(firm_code):
    fr_url = 'https://comp.fnguide.com/SVO2/asp/SVD_FinanceRatio.asp?pGB=1&cID=&MenuYr
    fr_page = requests.get(fr_url)
    fr_tables = pd.read_html(fr_page.text)

    temp_df = fr_tables[0]

    temp_df = temp_df.set_index(temp_df.columns[0])
    temp_df = temp_df.loc[['EPS증가율계산에 참여한 계정 펼치기']]
    temp_df.index = ['EPS증가율']   #행 이름 간편하게
    return temp_df
```

```
for num, code in enumerate(mediumlist['종목코드']): # eps가져오기
    try:
        print(num, code)
        time.sleep(1)
        try:
            fr_df = make_fr_dataframe(code)
        except requests.exceptions.Timeout:
            time.sleep(60)
            fr_df = make_fr_dataframe(code)
        fr_df_changed = change_df(code, fr_df)
        if num == 0 :
            total_fr = fr_df_changed
        else:
            total_fr = pd.concat([total_fr, fr_df_changed])
    except ValueError:
        continue
    except KeyError:
        continue
```

중형주 종목코드 이용해

EPS 증가율 정보 불러오기

```
for num, code in enumerate(mediumlist['종목코드']): #eps 가져오기
```

```
total_invest.to_csv('EPS증가율.csv')
#eps증가율 데이터 저장
```

코드 중소형주 종목 선정

3. 저변동성 데이터

**표준편차 개념, 얼마나 큰 폭으로 변동했나

저변동성 데이터 구하기

```
#pykrx 사용 == 종가데이터 가져오기
#per과 eps데이터, 기관매수대금 데이터의 범위에 맞춰서 데이터를 가져와야함.

price_df = pd.DataFrame()

tickers = stock.get_market_ticker_list(market='ALL')
for i, ticker in enumerate(tickers):
    print(i, ticker)
    try:
        temp = stock.get_market_ohlcv_by_date("20170101", "20210330", ticker) #
        price_df['A'+ticker] = temp['종가']
        time.sleep(0.1)
    except requests.exceptions.Timeout:
        time.sleep(60)
        temp = stock.get_market_ohlcv_by_date("20170101", "20210330", ticker)
        price_df['A'+ticker] = temp['종가']
```

코드 중소형주 종목 선정

3. 저변동성 데이터

```
price_df = pd.read_csv('가격데이터.csv')

keras_df = pd.DataFrame() #데이터 프레임 생성

for i in price_df.columns[1:] : #여기서 데이터의 칼럼명은 종목코드 들이다.

    kenta = price_df.loc[:,i].pct_change() # pandas에 내장된 수익률 구하기 함수 , 현재행과 다음행을 계산하여 수익률을 만듦
    kenta = pd.DataFrame(kenta) #만든 데이터를 데이터 프레임화 시킨 후 keras_df와 병합
    keras_df = pd.concat([keras_df,kenta],axis=1)

#저변동성 == 표준편차
keras_df_1 = keras_df.iloc[:243,:].std()     #거래대금 데이터가 총 13개 (3개월 단위)
keras_df_2 = keras_df.iloc[61:243+61,:].std()  #1년은 243개 행 3개월은 61
keras_df_3 = keras_df.iloc[61*2:243+(61*2),:].std()
keras_df_4 = keras_df.iloc[61*3:243+(61*3),:].std() #아무생각없이 반복문 안쓰고 만들어버렸음
keras_df_5 = keras_df.iloc[61*4:243+(61*4),:].std()
keras_df_6 = keras_df.iloc[61*5:243+(61*5),:].std()
keras_df_7 = keras_df.iloc[61*6:243+(61*6),:].std()
keras_df_8 = keras_df.iloc[61*7:243+(61*7),:].std()
keras_df_9 = keras_df.iloc[61*8:243+(61*8),:].std()
keras_df_10 = keras_df.iloc[61*9:243+(61*9),:].std()
keras_df_11 = keras_df.iloc[61*10:243+(61*10),:].std()
keras_df_12 = keras_df.iloc[61*11:243+(61*11),:].std()
keras_df_13 = keras_df.iloc[61*12:243+(61*12),:].std()



#데이터 프레임화
keras_df_1 = pd.DataFrame(keras_df_1)
keras_df_2 = pd.DataFrame(keras_df_2)
keras_df_3 = pd.DataFrame(keras_df_3)
keras_df_4 = pd.DataFrame(keras_df_4)
keras_df_5 = pd.DataFrame(keras_df_5)
keras_df_6 = pd.DataFrame(keras_df_6)
keras_df_7 = pd.DataFrame(keras_df_7)
keras_df_8 = pd.DataFrame(keras_df_8)
keras_df_9 = pd.DataFrame(keras_df_9)
keras_df_10 = pd.DataFrame(keras_df_10)
keras_df_11 = pd.DataFrame(keras_df_11)
keras_df_12 = pd.DataFrame(keras_df_12)
keras_df_13 = pd.DataFrame(keras_df_13)

keras_df_volacity = pd.concat([keras_df_1,keras_df_2,keras_df_3,keras_df_4,keras_df_5,keras_df_6
                               ,keras_df_7,keras_df_8,keras_df_9,keras_df_10,keras_df_11,keras_df_12,keras_df_13],axis=1)


realvolacity = keras_df_volacity.loc[mediumlist['종목코드'],:] #중형주에 해당하는 행만 뽑아오기
```

#거래대금 데이터가 총 13개 (3개월 단위)

```
realvolacity = keras_df_volacity.loc[mediumlist['종목코드'],:]
#중형주에 해당하는 행만 뽑아오기
```

코드 중소형주 종목 선정

3. 저변동성 데이터

종목코드	1	2	3	4	5	6	7	8	9	10	11	12	13
A196170	0.023210	0.029842	0.040248	0.047833	0.047590	0.046339	0.040919	0.042686	0.048313	0.055267	0.064995	0.062058	0.058833
A010620	0.028438	0.030077	0.030836	0.030151	0.027574	0.025320	0.022648	0.022803	0.020515	0.029984	0.033940	0.033125	0.035604
A036460	0.015430	0.018204	0.022190	0.023176	0.023765	0.022461	0.019308	0.018254	0.015487	0.022505	0.025068	0.025291	0.029185
A036490	0.021455	0.021863	0.023509	0.023198	0.022414	0.022757	0.020281	0.021662	0.020808	0.024791	0.027171	0.028960	0.031748
A039490	0.021396	0.023811	0.025702	0.026160	0.027716	0.026191	0.024253	0.023651	0.019594	0.022498	0.026358	0.028777	0.032015
...	...	...	...	...	...	...	...	...	...	...	...	...	...
A010050	0.014966	0.018808	0.035039	0.036250	0.036963	0.035108	0.019268	0.016654	0.013011	0.022433	0.023973	0.023695	0.023780
A268280	0.034042	0.029143	0.012883	0.012505	0.012039	0.011625	0.011930	0.010816	0.009978	0.011523	0.010345	0.011035	0.011470
A280360	0.022222	0.019395	0.017718	0.018894	0.019171	0.020371	0.021170	0.019350	0.017100	0.018800	0.021247	0.021417	0.020932
A082640	0.015101	0.014614	0.016040	0.016458	0.017922	0.017891	0.016751	0.017865	0.016400	0.031059	0.033684	0.033218	0.033029
A192400	0.018258	0.021592	0.024807	0.026864	0.030089	0.029793	0.030764	0.030701	0.027197	0.028768	0.026537	0.024641	0.023662

200 rows × 13 columns

코드 중소형주 종목 선정

4. 기관매수비율(거래량/기관순매수)

대신증권 API 사용하여 거래량과 기관순매수 데이터 가져오기

API 사용 코드 참고

```python
import win32com.client
import sqlite3
import time
import os
import pandas as pd
import numpy as np
import sys
```

```python
# 연결 여부 체크
def Connection():
    objCpCybos = win32com.client.Dispatch("CpUtil.CpCybos")
    bConnect = objCpCybos.IsConnect
    if (bConnect == 0):
        print("PLUS가 정상적으로 연결되지 않음. ")
        exit()
    else:
        print("PLUS가 정상적으로 연결되었음")
    return
Connection()
```

```python
def get_stock_list():

    Connection()
    print("종목을 가져옵니다")
    objCpCodeMgr = win32com.client.Dispatch("CpUtil.CpCodeMgr")

    code_list = objCpCodeMgr.GetStockListByMarket(1) #코스피
    code_list2 = objCpCodeMgr.GetStockListByMarket(2) #코스닥

    name_list = []
    name_list2 = []

    for i, code in enumerate(code_list):
        name1 = objCpCodeMgr.CodeToName(code)
        name_list.append(name1)

    for i, code in enumerate(code_list2):
        name2 = objCpCodeMgr.CodeToName(code)
        name_list2.append(name2)

    df_kospi = pd.DataFrame([code_list,name_list])
    df_kosdaq = pd.DataFrame([code_list2,name_list2])
    df_kospi = df_kospi.transpose()
    df_kosdaq = df_kosdaq.transpose()
    df_kospi.columns = ['code','name']
    df_kosdaq.columns = ['code','name']


    return (df_kospi, df_kosdaq)

kospi_list, kosdaq_list = get_stock_list()
```

코드 중소형주 종목 선정

4. 기관매수비율(거래량/기관순매수)

```
def RequestDT(code):

    def RequestData(objStockChart):

        objStockChart.BlockRequest()

        rqStatus = objStockChart.GetDibStatus()
        rqRet = objStockChart.GetDibMsg1()
        #print('통신상태', rqStatus, rqRet)
        if rqStatus != 0:
            exit()

        length = objStockChart.GetHeaderValue(3)

        for i in range(length):
            dates.append(objStockChart.GetDataValue(0, i))

            A1.append(objStockChart.GetDataValue(2, i))
            A2.append(objStockChart.GetDataValue(3, i))

        return True

    # 폴더 없으면 생성
    path = "./day_data_kospi"
    if not os.path.isdir(path):
        os.mkdir(path)

    objStockChart = win32com.client.Dispatch("CpSysDib.StockChart")

    g_objCodeMgr = win32com.client.Dispatch('CpUtil.CpCodeMgr')
    g_objCpStatus = win32com.client.Dispatch('CpUtil.CpCybos')

    name = g_objCodeMgr.CodeToName(code)

    #  이름에 / 있으면 빼주기
    #------------------------------------------------
    if name.find('/') != -1:
        name = name.split('/')[0] + name.split('/')[1]
    else:
        name = name
    #------------------------------------------------

    dates = []
    A1 = []
    A2 = []



    objStockChart.SetInputValue(0, code)  # 종목코드


    objStockChart.SetInputValue(1, ord('2'))  # 개수로 받기
```

```
    # 기관으로 받을 경우
    #------------------------------------------------
    #objStockChart.SetInputValue(1, ord('1'))  # 개수로 받기
    #startdate = '20100101'
    #enddate = '20200101'
    #objStockChart.SetInputValue(2, enddate)
    #objStockChart.SetInputValue(3, stardate)
    #------------------------------------------------


    objStockChart.SetInputValue(4, 916)  # 조회 개수
    objStockChart.SetInputValue(5, [0, 1,8,20])  # 요청항목 - 날짜,
    objStockChart.SetInputValue(6, ord('D'))  # '차트 주기 - 분/틱
    objStockChart.SetInputValue(9, ord('1'))  # 수정주가 사용


    ret = RequestData(objStockChart)

    # 통신확인
    #------------------------------------------------
    if ret == False:
        print('False')
        exit()
    #------------------------------------------------


    # 데이터 연속 조회
    #------------------------------------------------
    NextCount = 0
    time.sleep(0.3)
    while objStockChart.Continue:

        NextCount+=1;
        if (NextCount > 500):
            break

        ret = RequestData(objStockChart)
        if ret == False:
            print('False')
            exit()

        time.sleep(0.3)
    #------------------------------------------------

    # 받은 데이터를 데이터프레임으로 만들기 및 저장
    #------------------------------------------------
    df = pd.DataFrame({"date":dates,"거래량":A1,"기관순매수":A2})
    df = df.set_index('date')
    charfile = path + "/" + name + '.csv'
    df.to_csv(charfile,encoding='utf-8-sig')
    #------------------------------------------------

    return df

df = RequestDT('A035720') # A005930
df
```

```
objStockChart.SetInputValue(4, 916)
# 조회 개수
objStockChart.SetInputValue(5, [0, 1,8,20])
# 요청항목 - 날짜, 시간,시가,고가,저가,종가,거래량
# 거래량 ,기관순매수량
objStockChart.SetInputValue(6, ord('D'))
# '차트 주기 - 분/틱
objStockChart.SetInputValue(9, ord('1'))
# 수정주가 사용
```

코드 중소형주 종목 선정

4. 기관매수비율(거래량/기관순매수)

기관매수대금 데이터 가져오기

```
# using RequestDT(code)

mediumlist = pd.read_csv("중형주리스트.csv")


def makingbuy(x) :

    ratio = []

    for j in range(13):
        a = df[61*j :61*(j+1)][["거래량","기관순매수"]].sum()[1]/df[61*j:61*(j+1)][["거래량","기관순매수"]].sum()[0]
        ratio.append(a)

    b = pd.DataFrame(ratio)

    return b



for num , i in enumerate(mediumlist['종목코드']) :
    df = RequestDT(i)
    df = df[121:] # 2017년 10월부터 (10,11,12) 2020년 12월 30일까지

    b = makingbuy(i)

    b.columns = [i]

    if num == 0 :
        total_df = b

    else :
        total_df = pd.merge(total_df,b,how='outer',left_index=True, right_index=True)

total_df #오류메시지는 결측치생성에 따라 생성되는 것 (거래량이 0인 경우들) # 오류가 발생해도 결과값은 제대로 나온다 걱정 ㄴㄴ
```

기관매수비율 데이터 저장

```
total_df.to_csv("기관매수.csv")
#기관매수량 저장 주의! 역순으로 되어있음(최근 부터 시작)
```

코드 중소형주 종목 선정

5. 딥러닝 데이터 전처리

딥러닝을 위한 데이터 전처리

```
per_rawdata = pd.read_csv('PER.csv')
eps_rawdata = pd.read_csv('EPS증가율.csv')
volacity_rawdata = pd.read_csv('volacity.csv')
quantity_rawdata = pd.read_csv('기관매수.csv')


per_rawdata = per_rawdata.set_index('Unnamed: 0')
eps_rawdata = eps_rawdata.set_index('Unnamed: 0')
volacity_rawdata = volacity_rawdata.set_index('Unnamed: 0')

#quantity는 이미 인덱스가 바뀌어있음

qunatity_rawdata = quantity_rawdata.fillna(-1)


X_data = pd.DataFrame()

quantity_rawdata = quantity_rawdata.T
```

```
quantity_rawdata
```

X학습 데이터

1. PER.csv
2. EPS증가율.csv
3. volacity.csv
4. 기관매수.csv

코드 중소형주 종목 선정

5. 딥러닝 데이터 전처리

```
#에 해당하는 학습데이터 만들기

for i in range(4) : #2017 , 2018, 2019, 2020
    perdata = per_rawdata.iloc[1:,i+2] #per 데이터 할당

    epsdata = eps_rawdata.iloc[1:,i+2] #eps 데이터 할당
    epsdata1=epsdata.replace('흑전','0.5')    # 흑자전환과 적자전환 적자지속 에 해당하는 '문자열'을
    epsdata1=epsdata1.replace('적지','-80.0') # -80과 0.5 의 임의의 숫자로 할당 == 시실상 고려하지 않기를 위해 극단적인 값으로 할당
    epsdata1=epsdata1.replace('적전','-80.0')

    if i <=2 :
        for j in range(4):
            volacitydata = volacity_rawdata.iloc[:,(4*i)+j] #저변동성 데이터 할당

            quantitydata = quantity_rawdata.iloc[1:,(4*i)+j]

            final_data = pd.concat([perdata,epsdata1,volacitydata,quantitydata],axis=1) #데이터 프레임 합치기
            final_data.columns = ['PER','EPS증가율','저변동성','QUANTITY'] #칼럼 이름 지정
            final_data = final_data.apply(pd.to_numeric,errors = 'coerce') #계산이 가능하도록 numeric으로 변환

    # Z-SCORE 형태로 표준화를 시켜줌 -- 서로간의 단위가 다르기 때문

            final_data['PER스코어'] = (final_data['PER'] - final_data['PER'].mean())/final_data['PER'].std()
            final_data['EPS스코어'] = -(final_data['EPS증가율'] - final_data['EPS증가율'].mean())/final_data['EPS증가율'].std()
            final_data['VOL스코어'] = (final_data['저변동성'] - final_data['저변동성'].mean())/final_data['저변동성'].std()
            final_data['QUT스코어'] = -(final_data['QUANTITY'] - final_data['QUANTITY'].mean())/final_data['QUANTITY'].std()

    #EPS,QUT 스코어에  ' - ' 를 붙인이유 : PER과 저변동성 모두 낮을 수록 좋은 수치이기 때문에 이와 같이 통일하기 위해서 -를 붙임

            X_data = pd.concat([X_data,final_data])


    else :
        volacitydata = volacity_rawdata.iloc[:,12] #저변동성 데이터 할당

        quantitydata = quantity_rawdata.iloc[1:,12]

        final_data = pd.concat([perdata,epsdata1,volacitydata,quantitydata],axis=1) #데이터 프레임 합치기
        final_data.columns = ['PER','EPS증가율','저변동성','QUANTITY'] #칼럼 이름 지정
        final_data = final_data.apply(pd.to_numeric) #계산이 가능하도록 numeric으로 변환


        final_data['PER스코어'] = (final_data['PER'] - final_data['PER'].mean())/final_data['PER'].std()
        final_data['EPS스코어'] = -(final_data['EPS증가율'] - final_data['EPS증가율'].mean())/final_data['EPS증가율'].std()
        final_data['VOL스코어'] = (final_data['저변동성'] - final_data['저변동성'].mean())/final_data['저변동성'].std()
        final_data['QUT스코어'] = -(final_data['QUANTITY'] - final_data['QUANTITY'].mean())/final_data['QUANTITY'].std()


        X_data = pd.concat([X_data,final_data])
```

#Z-SCORE 형태로 값 표준화 진행

코드 중소형주 종목 선정

5. 딥러닝 데이터 전처리

종목코드	PER	EPS증가율	저변동성	QUANTITY	PER스코어	EPS스코어	VOL스코어	QUT스코어
A196170	NaN	-80.0	0.023210	0.017803	NaN	0.616026	-0.019487	-0.585316
A010620	3.61	1163.8	0.028438	-0.008420	-0.401098	-5.961855	0.576659	-0.144113
A036460	NaN	-80.0	0.015430	0.016213	NaN	0.616026	-0.906773	-0.558573
A036490	18.63	-6.2	0.021455	0.009042	-0.251741	0.225732	-0.219643	-0.437918
A039490	8.06	33.5	0.021396	-0.040183	-0.356848	0.015777	-0.226464	0.390309
...	...	...	...	...	...	...	...	...
A010050	6.06	13.0	0.023780	0.002644	-0.133014	0.107188	-0.913004	0.107086
A268280	15.54	-10.8	0.011470	0.003613	-0.119446	0.155562	-1.731309	0.093386
A280360	15.97	7.7	0.020932	-0.029429	-0.118831	0.117960	-1.102329	0.560997
A082640	4.42	11.6	0.033029	0.003856	-0.135361	0.110034	-0.298128	0.089945
A192400	6.66	34.3	0.023662	0.004299	-0.132155	0.063895	-0.920836	0.083678

2600 rows × 8 columns

코드 중소형주 종목 선정

5. 딥러닝 데이터 전처리

딥러닝을 효과적으로 이용하고 해석하기 위해서 특정값을 예측하기 보다는 일정 범위안에 들었는지 분류하는 것이 더 용이하다고 생각하여

분류 형태 (클래스)로 Y (종속변수) 값을 설정하였다

```
#y데이터 만들기
price_df = pd.read_csv('가격데이터.csv')
price_dfx = price_df.iloc[:,1:]   #종목주어 해당하는 수익률 데이터만 가져오기
price_dfx = price_dfx.apply(pd.to_numeric) #계산이 가능하도록 바꿔줌

#특정 날짜 이후의 값과 비교하여 수익률을 계산
#23 1개월  #61 3개월  <- 원하는 개월수에 맞추기
#현재 3개월 이후의 수익률을 y값으로 활용하고 있다.

# 10위 안에 들었으면 1 , 아니면 0
def finite(x):
    if x <= 10 :
        return 1
    else :
        return 0


Y_data = pd.DataFrame()

for i in range(13):
    kunka = (price_dfx.iloc[61*(i+1)+186,1:]-price_dfx.iloc[61*i + 186,1:])/price_dfx.iloc[61*i + 186,1:]
    kunka =  pd.DataFrame(kunka)
    kunkax = kunka.loc[mediumlist['종목코드'],:]
    kunkax['rank'] = kunkax.rank()
    kunkax['zscore']= kunkax[0]-kunkax[0].mean()/kunkax[0].std()
    kunkax['class'] = kunkax['rank'].apply(finite) # finite 함수 적용

    Y_data = pd.concat([Y_data,kunkax])

Y_data
```

종속변수 설정

코드 중소형주 종목 선정

5. 딥러닝 데이터 전처리

종목코드	0	rank	zscore	class
A196170	0.710322	161.0	0.257107	0
A010620	0.018256	64.0	-0.434959	0
A036460	0.042118	78.0	-0.411097	0
A036490	-0.109101	12.0	-0.562316	0
A039490	0.248968	132.0	-0.204246	0
...	...	...	...	...
A010050	0.055118	52.0	-0.311944	0
A268280	0.101064	80.0	-0.265998	0
A280360	0.034483	44.0	-0.332580	0
A082640	0.155993	111.0	-0.211069	0
A192400	0.045263	49.0	-0.321799	0

코드 중소형주 종목 선정

5. 딥러닝 데이터 전처리

```
# class데이터만 X데이터와 합치고 결측치를 제거한 후 저장

X_data['y'] = Y_data['class']
X_data = X_data.dropna(how='any')
X_data.isnull().sum().sum()
```

```
X_data
```

종목코드	PER	EPS증가율	저변동성	QUANTITY	PER스코어	EPS스코어	VOL스코어	QUT스코어	y
A010620	3.61	1163.8	0.028438	-0.008420	-0.401098	-5.961855	0.576659	-0.144113	0
A036490	18.63	-6.2	0.021455	0.009042	-0.251741	0.225732	-0.219643	-0.437918	0
A039490	8.06	33.5	0.021396	-0.040183	-0.356848	0.015777	-0.226464	0.390309	0
A088350	10.26	-26.5	0.016679	0.016250	-0.334971	0.333089	-0.764392	-0.559187	0
A020150	38.40	3.0	0.035474	0.005524	-0.055151	0.177077	1.379063	-0.378725	0
...	...	...	...	...	...	...	...	...	...
A010050	6.06	13.0	0.023780	0.002644	-0.133014	0.107188	-0.913004	0.107086	0
A268280	15.54	-10.8	0.011470	0.003613	-0.119446	0.155562	-1.731309	0.093386	0
A280360	15.97	7.7	0.020932	-0.029429	-0.118831	0.117960	-1.102329	0.560997	0
A082640	4.42	11.6	0.033029	0.003856	-0.135361	0.110034	-0.298128	0.089945	0
A192400	6.66	34.3	0.023662	0.004299	-0.132155	0.063895	-0.920836	0.083678	0

1864 rows × 9 columns

코드 중소형주 종목 선정

5. 딥러닝 적용

딥러닝 적용하기 ¶

```
###############딥러닝 기동####################  X_data Y_rank_data Y_zscore_data
X_data = pd.read_csv("Dataset.csv")
X_data = X_data.set_index("Unnamed: 0")
X_data
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold

import numpy
import pandas as pd
import tensorflow as tf


X = X_data.values[:,4:8]
Y = X_data.values[:,8]

X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.1, random_state=0)

model = Sequential()
model.add(Dense(30,activation='relu',input_dim=4))
model.add(Dense(20,activation='relu'))
model.add(Dense(1,activation='sigmoid'))

model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])


model.fit(X_train,Y_train,epochs=100,batch_size=100)
```

```
Epoch 6/100 17/17 [==============================]
- 0s 561us/step - loss: 0.2162 - accuracy: 0.9474
Epoch 7/100 17/17 [==============================]
- 0s 561us/step - loss: 0.1630 - accuracy: 0.9628

                                ...
```

2) MACD, RSI 전략

매수/매도

코드 MACD, RSI 전략

1. MACD

중소형 서프라이즈
종목 10개 전달

```
stock_codes = [] #종목 선별해서 종목코드 리스트 넘겨줄것

stock_list = ["096530","036170","007390","043100","084370","049080","025550","035610",

stock_codes = stock_list
```

```
#종목코드, 종목이름 df 생성 -> 중간에 한번 체크하면 편할것같아서
stock_names = []
for stock_code in stock_codes :
    name = stock.get_market_ticker_name(stock_code) # 종목코드로 종목이름 가져와서
    stock_names.append(name)

stock_dic = {
    '종목코드':stock_codes,
    '종목이름':stock_names
}
stock_code_name = pd.DataFrame(stock_dic)
stock_code_name
```

코드 MACD, RSI 전략

1. MACD

각 종목별 날짜, 시가, 종가 등 데이터프레임 생성

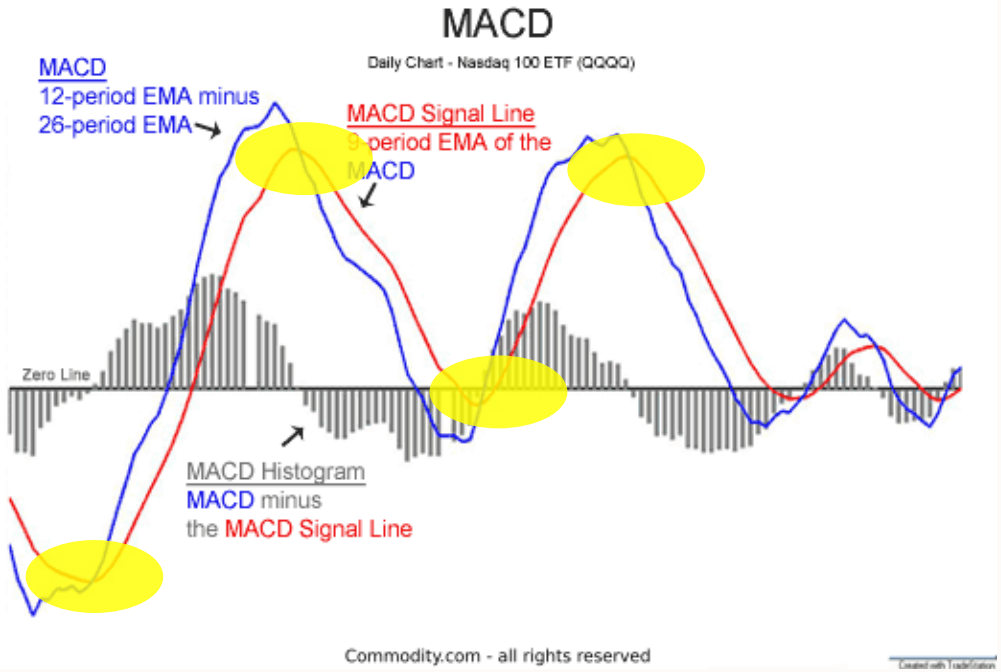
```
#각 종목의 날짜별 시가종가 df 생성
stock_dfs = [] #df 여러개 보관
start_date = "20190101"
end_date = "20210627"

for code in stock_codes :
    stock_df = stock.get_market_ohlcv_by_date(start_date, end_date, code)
    stock_dfs.append(stock_df)
```

```
stock_dfs[0].head()
```

날짜	시가	고가	저가	종가	거래량
2019-01-02	8141	8191	7789	7840	107409
2019-01-03	7840	7890	7387	7413	188496
2019-01-04	7287	7638	7262	7538	177514
2019-01-07	7638	7739	7513	7639	132940
2019-01-08	7739	8116	7613	7840	200127

코드 MACD, RSI 전략

1. MACD



MACD값〉MACD Signal

매수

MACD값〈MACD Signal

매도

코드 MACD, RSI 전략

1. MACD

```python
# 지수 이동 평균(Exponential Moving Average, EMA)
def EMA(data, period=20, column='종가'):
    return data[column].ewm(span=period, adjust=False).mean()

# 단순 이동 평균(Simple Moving Average, SMA)
def SMA(data, period=30, column='종가'):
    return data[column].rolling(window=period).mean()
```

```python
# MACD 함수 정의
## df, 단기기간, 장기기간, signal기간 넣기면
## MACD, Signal column이 추가된 df 반환

def MACD(data, period_long, period_short, period_signal, column='종가'):

    # 단기 지수 이평선 계산 (AKA Fast moving average)
    ShortEMA = EMA(data, period_short, column=column)

    # 장기 지수 이평선 계산 (AKA Slow moving average)
    LongEMA = EMA(data, period_long, column=column)

    # 이동 평균 수렴/발산 계산
    data['MACD'] = ShortEMA - LongEMA

    # 신호선 계산
    data['Signal_Line'] = EMA(data, period_signal, column='MACD')

    return data
```

MACD(Moving Average Convergence & Divergence)의 세가지 지표

1) MACD= 단기지수이동평균값(12일)-장기지수이동평균값(26일)

2) MACD Signal=MACD의 9일 이동평균선

2. RSI

$$RSI = \frac{\text{14일 상승폭 합계}}{\text{14일 상승폭 합계(양)+14일 하락폭 합계(음)}}$$

0.7이면 과열, 0.3이면 침체 시그널

30이면 침체, 70이면 과열 시그널

코드 MACD, RSI 전략

2. RSI

```
: #상대적 강도 지수 계산

def RSI(data, period=14, column='종가'):
    delta = data[column].diff(1)
    delta = delta.dropna()

    up = delta.copy()
    down = delta.copy()
    up[up < 0] = 0
    down[down > 0] = 0
    data['up'] = up
    data['down'] = down

    AVG_Gain = SMA(data, period, column = 'up')
    AVG_Loss = abs(SMA(data, period, column='down'))
    RS = AVG_Gain / AVG_Loss

    RSI = 100.0 - (100.0 / (1.0 + RS))
    data['RSI'] = RSI

    return data
```

코드 MACD, RSI 전략

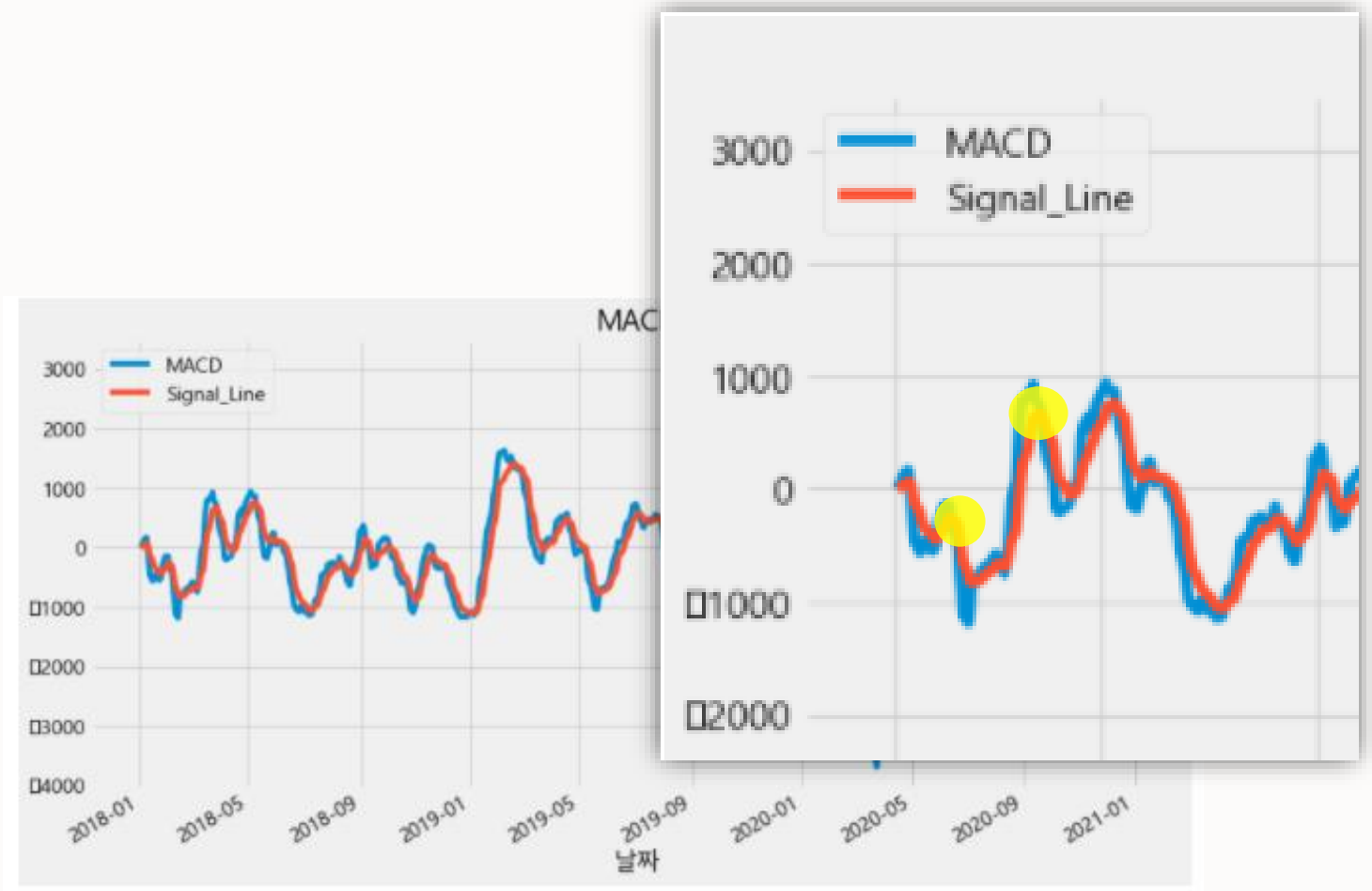
2. RSI

```
#기간들 설정
macd_short, macd_long, macd_signal=12,26,9
rsi_period = 14

stock_dfs[0] = MACD(stock_dfs[0],macd_long, macd_short, macd_signal)
stock_dfs[0] = RSI(stock_dfs[0], rsi_period)
stock_dfs[0] = stock_dfs[0].round(decimals=2) #소수점 2자리로 제한

stock_dfs[0].tail()
```

날짜	시가	고가	저가	종가	거래량	MACD	Signal_Line	up	down	RSI
2021-06-21	66000	76300	65600	74700	7980862	-2890.41	-4358.77	10500.0	0.0	61.94
2021-06-22	74800	75800	70600	71700	2827294	-2226.41	-3932.30	0.0	-3000.0	63.85
2021-06-23	73000	91000	72900	84800	16182453	-635.80	-3273.00	13100.0	0.0	75.96
2021-06-24	88800	92000	84500	85200	8169461	649.56	-2488.48	400.0	0.0	79.63
2021-06-25	85800	87500	82200	83900	3542140	1545.51	-1681.69	0.0	-1300.0	78.80

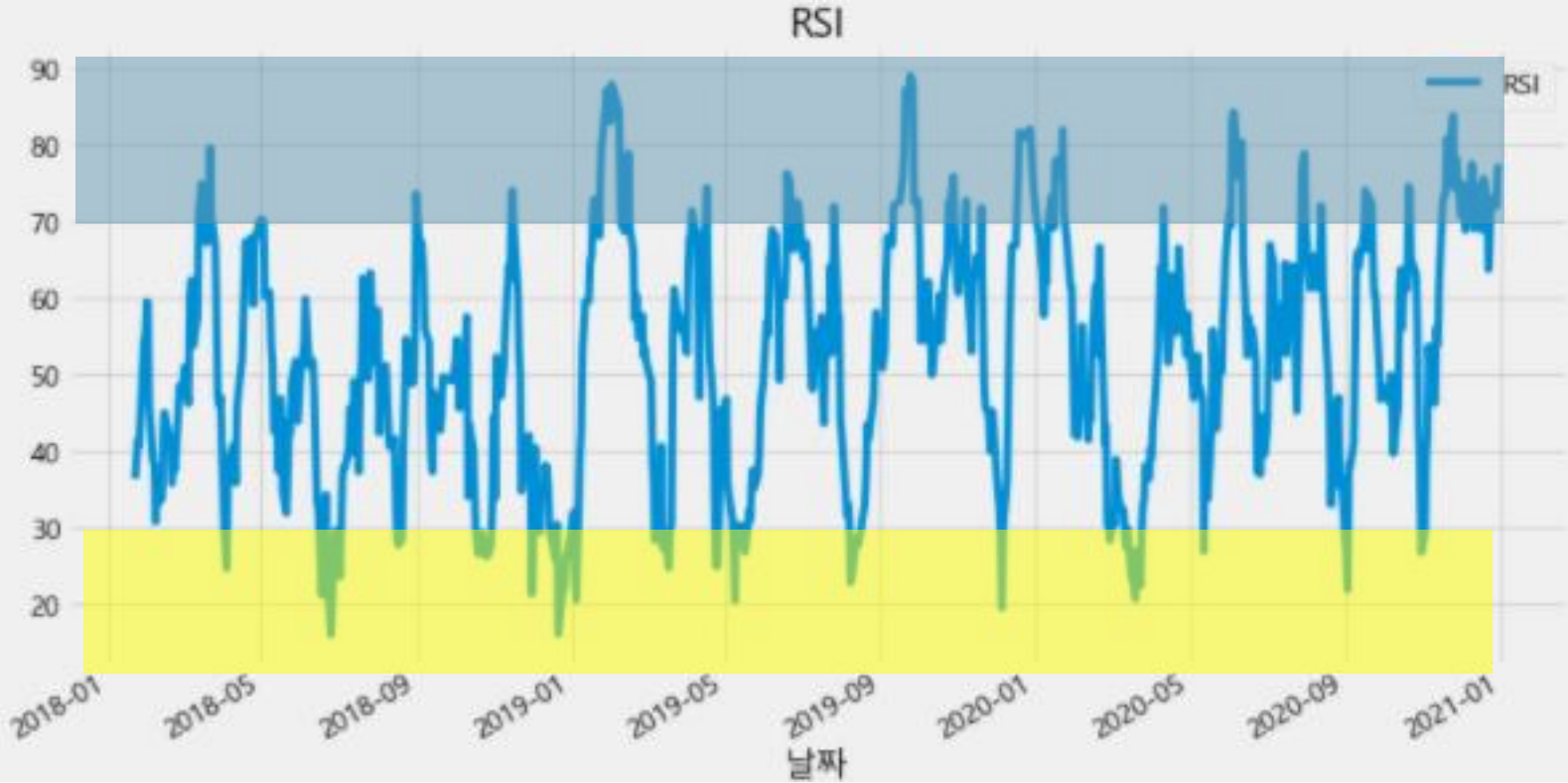
코드 MACD, RSI 전략

2. RSI



코드 MACD, RSI 전략

2. RSI

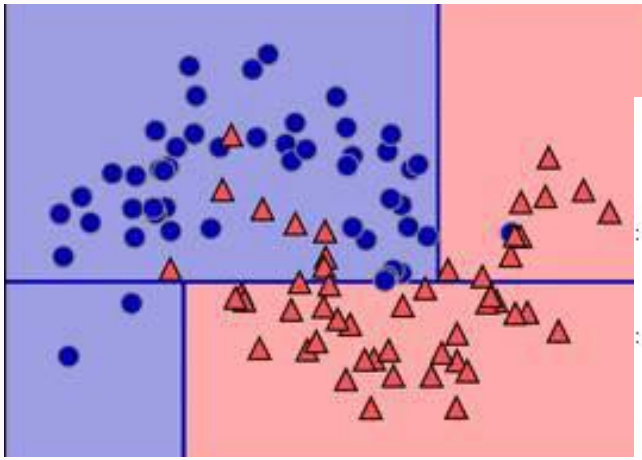


3. 머신러닝

결정트리모델
랜덤포레스트 모델
그래디언트 부스팅
SVM 소프트벡터 머신

머신러닝

1. 결정트리모델



결정트리 모델

```
from sklearn.tree import DecisionTreeClassifier
```

```
tree = DecisionTreeClassifier(max_depth=4,random_state=0)
tree.fit(X_train, Y_train)

print("훈련 세트 정확도: {:.3f}".format(tree.score(X_train, Y_train)))
print("테스트 세트 정확도: {:.3f}".format(tree.score(X_test, Y_test)))
```

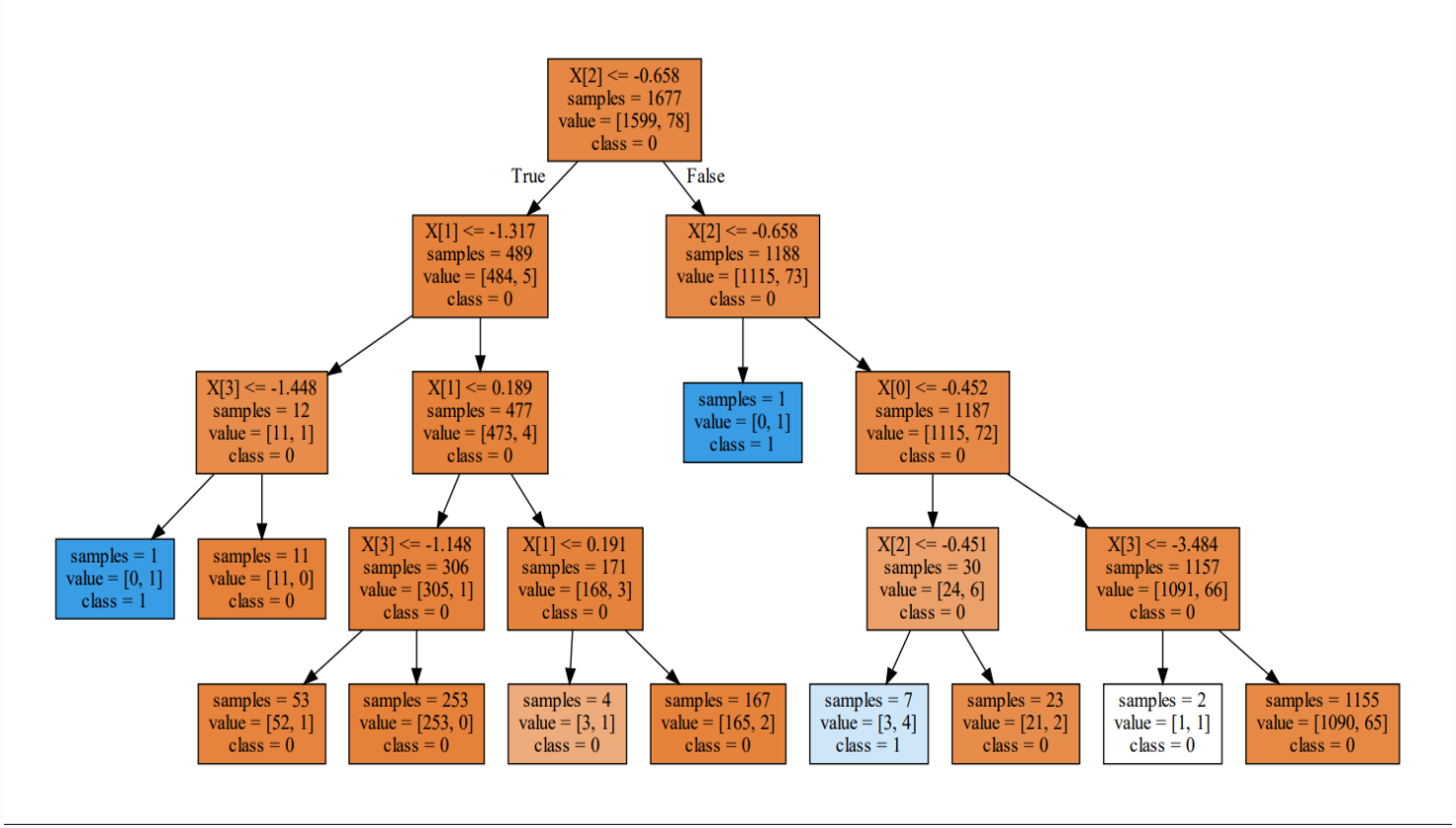
```
훈련 세트 정확도: 0.955
테스트 세트 정확도: 0.957
```

```
#특성중요도

print("특성 중요도:₩n{}".format(tree.feature_importances_))
```

```
특성 중요도:
[0.11312658 0.05857563 0.57740262 0.25089517]
```

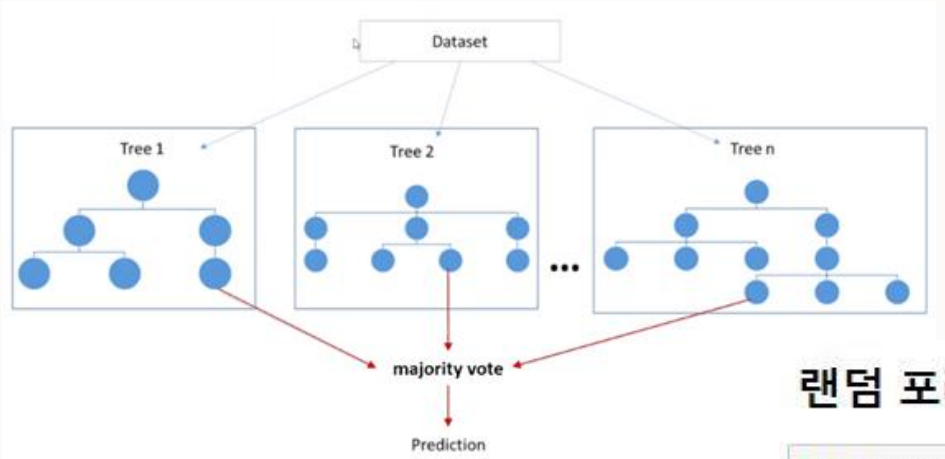
머신러닝

1. 결정트리모델



머신러닝

2. 랜덤포레스트 모델



랜덤 포레스트

```
from sklearn.ensemble import RandomForestClassifier
```

```
forest = RandomForestClassifier(n_estimators=100, random_state=0)
forest.fit(X_train, Y_train)

print("훈련 세트 정확도: {:.3f}".format(forest.score(X_train, Y_train)))
print("테스트 세트 정확도: {:.3f}".format(forest.score(X_test, Y_test)))
```

```
훈련 세트 정확도: 1.000
테스트 세트 정확도: 0.957
```

```
print("특성 중요도:₩n{}".format(forest.feature_importances_))
```

```
특성 중요도:
[0.20094126 0.17937712 0.3205069  0.29917472]
```

머신러닝

3. 그래디언트 부스팅

그래디언트 부스팅

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
gbrt = GradientBoostingClassifier(random_state=0) #기본값 : 트리 100개, 학습률 0.1
gbrt.fit(X_train, Y_train)

print("훈련 세트 정확도: {:.3f}".format(gbrt.score(X_train, Y_train)))
print("테스트 세트 정확도: {:.3f}".format(gbrt.score(X_test, Y_test)))
```

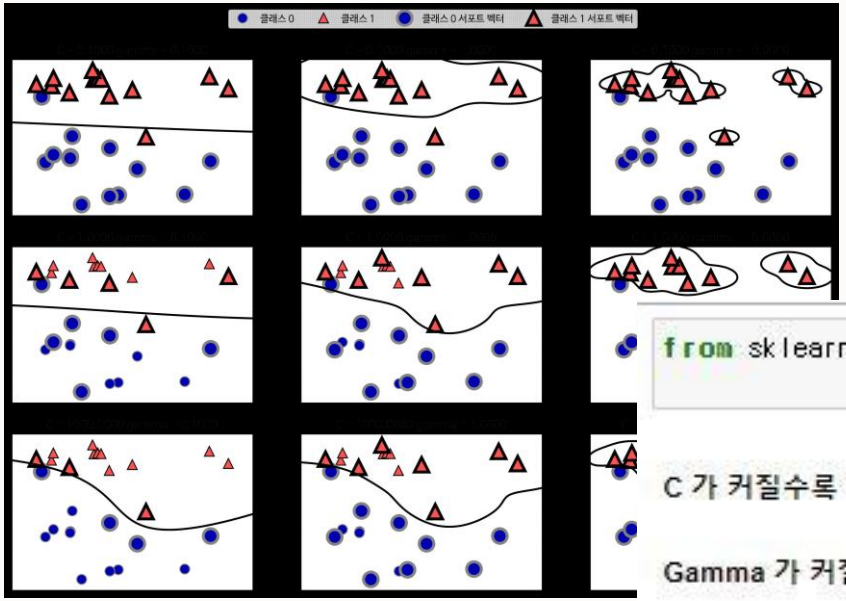
```
훈련 세트 정확도: 0.967
테스트 세트 정확도: 0.952
```

```
print("특성 중요도:₩n{}".format(gbrt.feature_importances_))
```

```
특성 중요도:
[0.16793128 0.09576762 0.40015163 0.33614946]
```

머신러닝

4. SVM 소프트벡터 머신




```
from sklearn.svm import SVC
```

C 가 커질수록 제약이 작아지고 데이터 포이터들의 영향력이 커진다.

Gamma 가 커질수록 경계와 포인트의 거리가 줄어들고 모델의 복잡도가 커진다.

```
svc = SVC()
svc.fit(X_train, Y_train)  #기본값 C=1, gamma=1/n_features

print("훈련 세트 정확도: {:.2f}".format(svc.score(X_train, Y_train)))
print("테스트 세트 정확도: {:.2f}".format(svc.score(X_test, Y_test)))
```

```
훈련 세트 정확도: 0.95
테스트 세트 정확도: 0.97
```

4. 계획

KQR Korea Quant Research

계획

1. 어닝서프라이즈 코드 병합 이슈

2. 백테스트 구현

   -매수매도 전략 적용
   -새로운 보조지표 활용

감사합니다