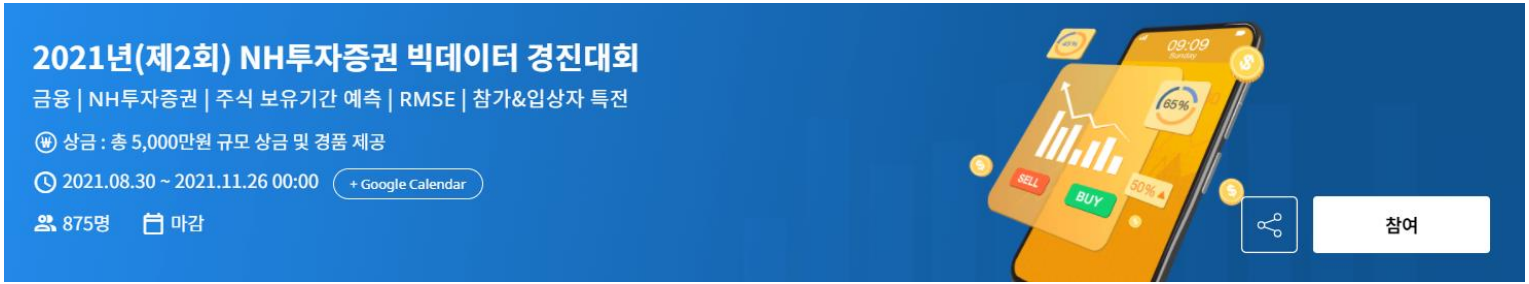
금융 프로젝트

팀 여의도

- NH 증권 주식 보유기간 예측
- 개인 신용 등급 분류 예측

< NH 증권 주식 보유기간 예측 >



cus_info.csv(10,000건): 고객 및 주거래계좌 정보

- act_id: 계좌 ID
- sex_dit_cd: 성별
- cus_age_stn_cd: 연령대
- ivs_icn_cd: 투자성향
- cus_aet_stn_cd: 자산구간
- mrz_pdt_tp_sgm_cd: 주거래상품군
- lsg_sgm_cd: Life Style
- tco_cus_grd_cd: 서비스 등급
- tot_ivs_te_sgm_cd: 총 투자기간
- mrz_btp_dit_cd: 주거래업종구분

stk_bnc_hist.csv(2,573,839건): 국내주식 잔고이력

- act_id: 계좌 ID
- bse_dt: 기준일자
- iem_cd: 종목코드
- bnc_qty: 잔고수량
- tot_aet_amt: 잔고금액
- stk_par_pr: 주당 액면가

iem_info.csv(3,078건): 종목 정보

- iem_cd: 종목코드
- iem_krl_nm: 종목한글명
- btp_cfc_cd: 종목업종
- mkt_pr_tal_scl_tp_cd: 시가총액 규모유형
- stk_dit_cd: 시장구분

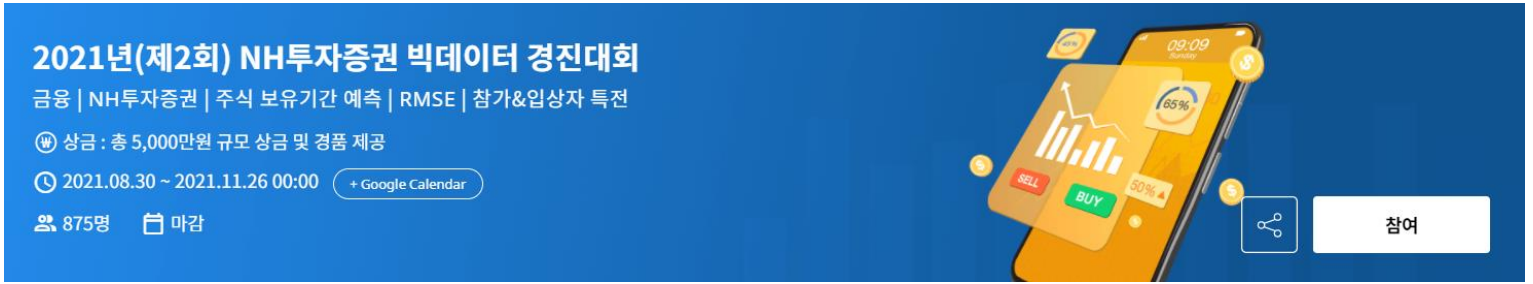
stk_hld_train.csv(681,472건): 16년 1월 ~ 20년 12월 시

- act_id: 계좌 ID
- iem_cd: 종목코드
- byn_dt: 매수일자
- hold_d: 보유기간(일)

stk_hld_test.csv(70,596건): 20년 12월 이전에 매수

- act_id: 계좌 ID
- iem_cd: 종목코드
- byn_dt: 매수 일자
- hist_d: 과거 보유일
- submit_id: 제출ID
- hold_d: 보유기간(일)

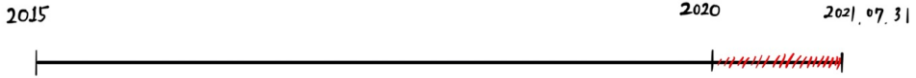
< NH 증권 주식 보유기간 예측 >



2015년~2020년 데이터를 활용해

2020년 12월 31일에 고객이 보유한 주식을

2021년 언제 매도할지 예측하는 대회



" 고객의 이전 데이터를 활용하여

매도 날짜를 예측하라 "

데이터 특성

- 방대한 고객 데이터 중 학습에 필요한 데이터만을 추려내야함
- 고객 데이터 중 같은 고객이 거래한 내용을 추려내야함
- 학습에 필요한 데이터 기간도 스스로 설정

- 외부 데이터 사용 가능!
- but 2021년 데이터는 어떤 방식으로든 사용 불가능

시도한 도전

1. hist_d : 도메인 지식을 활용해 학습에 필요한 기간을 설정
2. 종가 크롤링 후 수익률 feature 생성
3. Past_d : 고객 id를 구분하여 특정 고객의 이전 매도/ 매수 데이터를 학습
4. 추천 시스템의 아이디어 적용

거래일 기준으로 날짜 데이터 작업

주식은 거래소 영업일을 기준으로 날짜를 계산해야 함

Ex) 월요일 매수 & 다음주 화요일 매도
: 주식 보유 기간은 주말을 제외한 7일

공휴일, 12월 31일은 주식 거래소 휴장

거래일 기준으로 날짜 데이터 작업

KRX 휴장일 정보를 반영하여 주식 보유 기간 함수 구현

```
class Stock_Calendar(AbstractHolidayCalendar):
    rules = [
      Holiday('2016_신정', year=2016, month=1, day=1),
      Holiday('2016_설날1', year=2016, month=2, day=8),
      Holiday('2016_설날2', year=2016, month=2, day=9),
      Holiday('2016_설날3', year=2016, month=2, day=10),
      Holiday('2016_삼일절', year=2016, month=3, day=1),
      Holiday('2016_국회의원_선거', year=2016, month=4, day=13),
      Holiday('2016_어린이날', year=2016, month=5, day=5),
      Holiday('2016_어린이날_대체공휴일', year=2016, month=5, day=6),
      Holiday('2016_현충일', year=2016, month=6, day=6),
      Holiday('2016_광복절', year=2016, month=8, day=15),
      Holiday('2016_추석1', year=2016, month=9, day=14),
      Holiday('2016_추석2', year=2016, month=9, day=15),
      Holiday('2016_추석3', year=2016, month=9, day=16),
      Holiday('2016_개천절', year=2016, month=10, day=3),
      Holiday('2016_연말', year=2016, month=12, day=30),

      Holiday('2017_설날1', year=2017, month=1, day=27),
      Holiday('2017_설날_대체공휴일', year=2017, month=1, day=30),
      Holiday('2017_삼일절', year=2017, month=3, day=1),
```

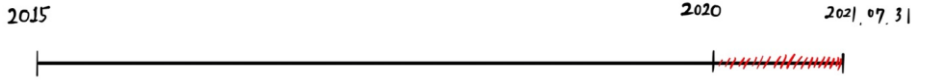
```
from datetime import datetime, date
from pandas.tseries.offsets import CustomBusinessDay
TDay = TradingDay = CustomBusinessDay(calendar=Stock_Calendar(AbstractHolidayCalendar)) # 공휴일을 포함한 영업일 함수 정의
```

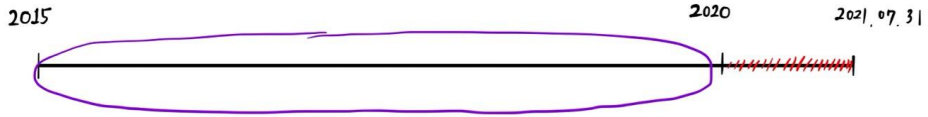
Function (2021.12.30 , 2022.01.05)

: 4

h i s t __ d

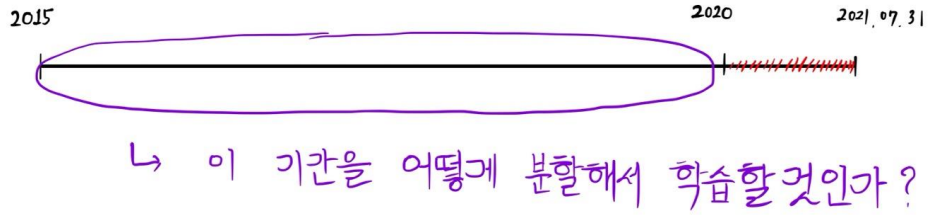
목표 :





↳ 이 기간을 어떻게 분할해서 학습할것인가?

h i s t _ d

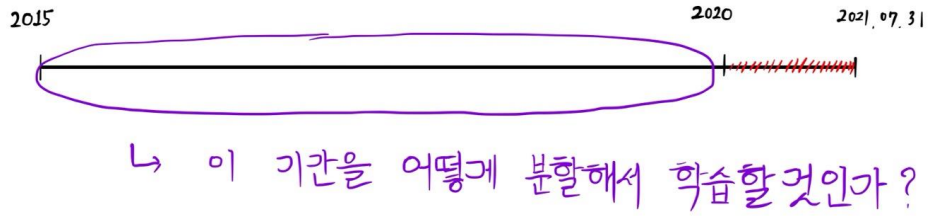


최종 target 기간에 맞춰서 7 개월 단위로 끊기?

Train 갯수를 최대한 늘리기위해 한달 단위로 끊기

장기 보유 고객의 매도는 특이값으로 따로 빼고 일년 단위로 끊기

h i s t _ d



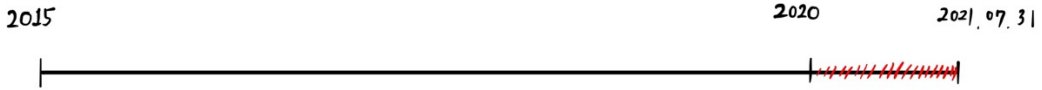
핵심 Idea : `배당`에 대한 도메인 지식을 이용하자!

" 대부분의 회사는 6개월 단위 반기배당 혹은 연말 배당 기준일로 배당금 지급 "

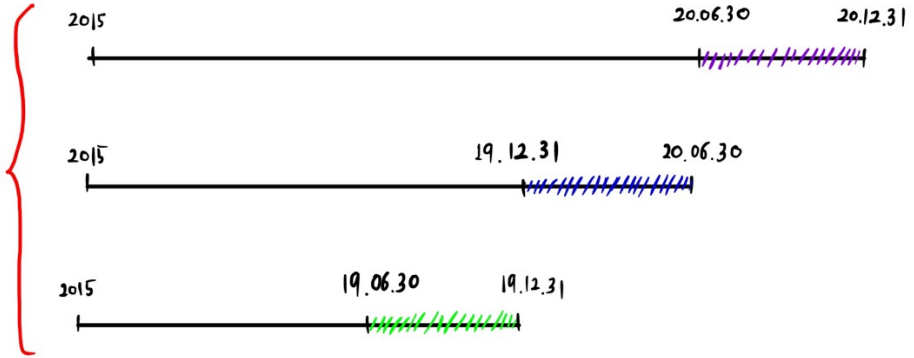
h i s t __ d

핵심 Idea : `배당`에 대한 도메인 지식을 이용하자!

목표



기간 분할 후 학습



Rmse 값

주식 개장일 정보 포함, 배당 idea 적용하여 기간 분할, 변수 타입에 맞는 전처리

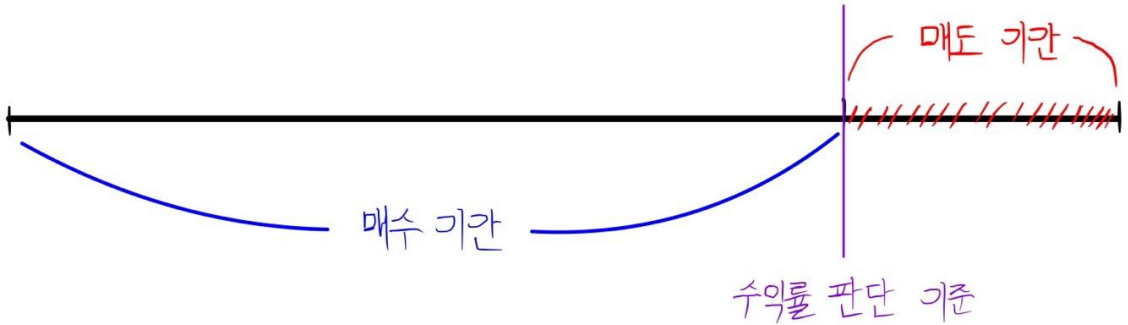
dacon_초안.csv edit	2021-09-07 20:02:12	82.0114459908 -
dacon_주식 보유기간 예측_09.14.csv edit	2021-09-14 20:32:55	71.2191455649 -

RMSE값 11 하락, Dacon Private 기준 12등

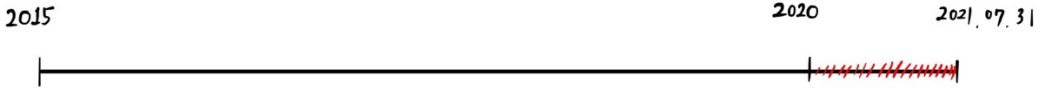
수익률 feature 제작

대회 규칙상 외부 데이터 사용 가능

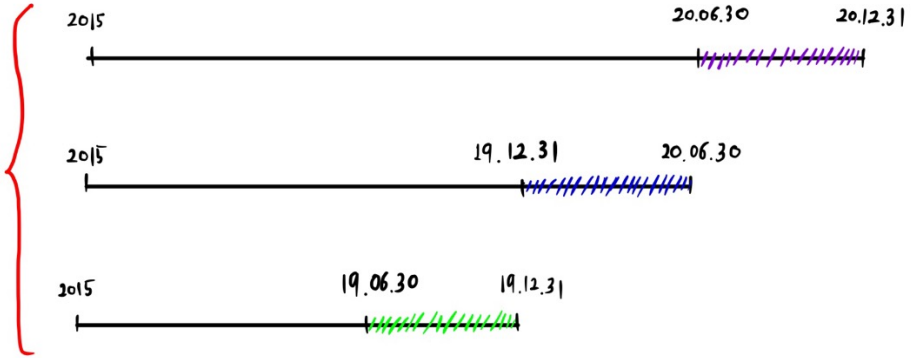
매도 판단에 있어서 수익률은 굉장히 중요한 정보



목표



기간 분할 후 학습

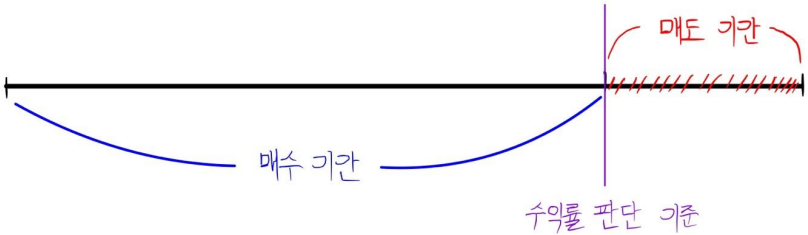


Ex) 2020년 07월 21일에 구매한 주식을 2021년 언제 매도할지 예측

- 사용 가능한 정보는 2020년 12월 31일까지
- 이때, 2020년 12월 31일까지의 수익률은 굉장히 중요한 정보!

Why...? 수익률이 굉장히 높거나 낮으면 일반적으로 쉽게 매도하지 않으니까

주식 종가 정보 크롤링



Ex) 2020년 07월 21일에 구매한 주식을 2021년 언제 매도할지 예측
- 사용 가능한 정보는 2020년 12월 31일까지
- 이때, 2020년 12월 31일까지의 수익률은 굉장히 중요한 정보!

Why...? 수익률이 굉장히 높거나 낮으면 일반적으로 쉽게 매도하지 않으니까

주식 매수 당시 구매 금액과 수익률 판단 기준일자 종가를 비교하여 '수익률' feature 추가

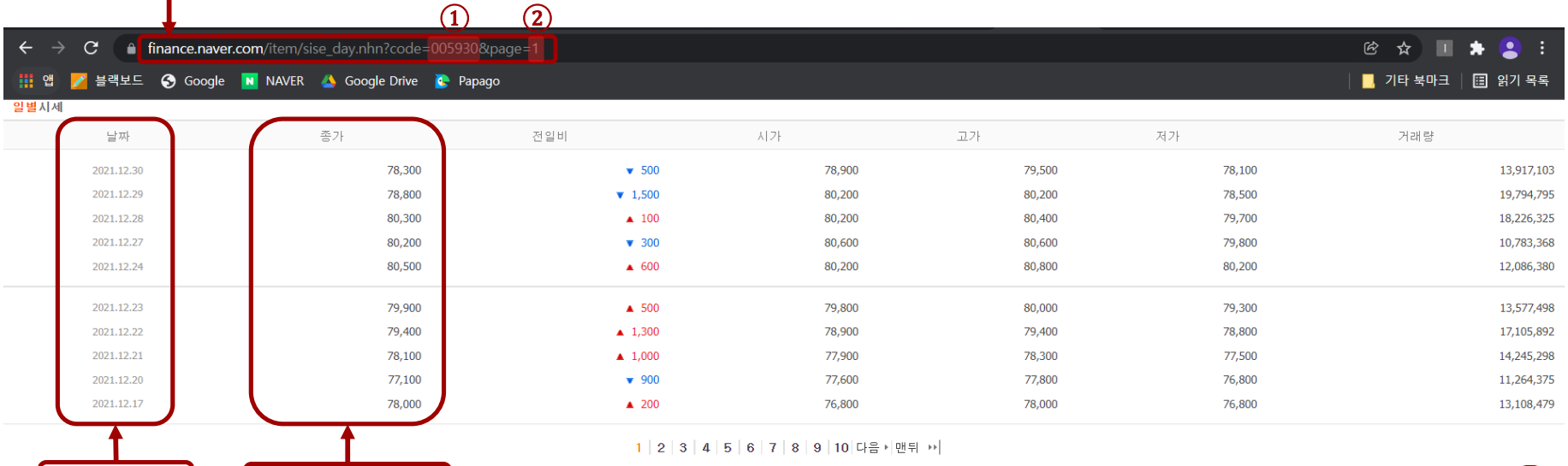
주식 종가 정보 크롤링 (네이버 금융)

Ex) 삼성전자 일별시세 (종목코드 : 005930)

네이버 금융 페이지 내 다양한 정보들 중
특정 주식 종목의 일별시세 테이블만 볼 수 있는 URL
① 종목 코드 지정
② 페이지 번호 지정

목표

**특정 일자(기준일)를 지정하여
해당 일자의 종가 정보를 크롤링**



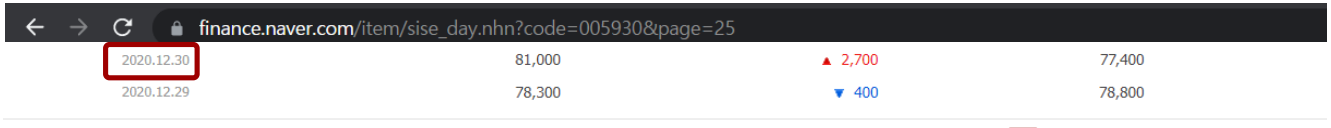
ppt 제목

종가 크롤링

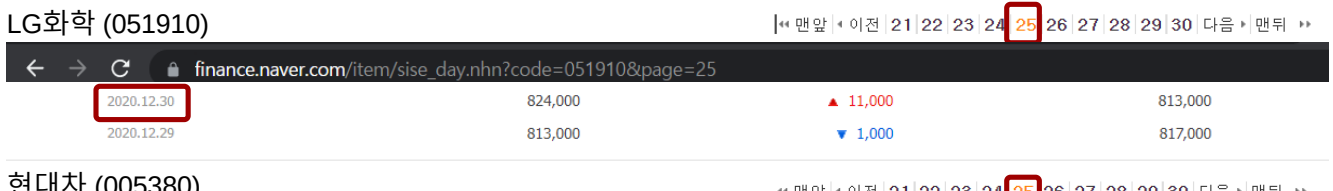
방법 (1)

일반적으로 기준일자가 존재하는 페이지만의 html만을 가져와서 해당 페이지 내에 기준일자가 존재한다면, 해당 기준일의 종가 정보 추출

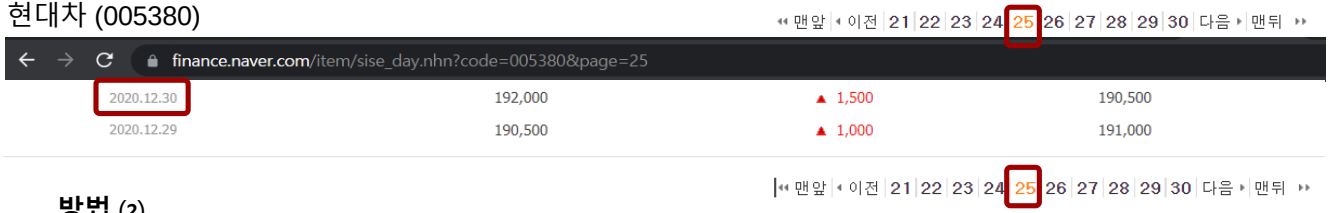
삼성전자 (005930)



LG화학 (051910)



현대차 (005380)



① 기준일자 이전/이후에 상장 폐지/상장되거나,
② 특정 주식개장일에 결측값이 존재하는 경우가 아닌 이상,

일반적인 경우에는 특정 페이지에 기준일자가 존재

↓

크롤링 소요 시간을 단축하기 위해 해당 페이지의 html만을 가져와서 기준일자의 종가 정보 추출

방법 (2)

(1)번 방법을 통해 종가 정보를 얻지 못한 케이스 중, (1)에서 지정해준 페이지 번호가 아닌, 다른 페이지 번호 내에 기준일자가 존재하는 경우 첫번째 페이지부터 마지막 페이지까지 크롤링하면서 기준일자가 존재하는 페이지를 찾고, 해당 기준일자의 종가 추출

필요한 함수 생성 - ① PARSE_PAGE(CODE, PAGE)

```python
# 종목코드와 페이지 번호를 입력으로 받아서 일별 시세를 판다스 데이터 프레임 객체로 반환하는 함수
def PARSE_PAGE(CODE, PAGE):
    try:
        URL = 'http://finance.naver.com/item/sise_day.nhn?code={code}&page={page}'.format(code = CODE, page = PAGE)
        # 네이버 금융 사이트는 반드시 헤더 정보를 요구
①       HEADERS = {'user-agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36'}
②       RES = requests.get(URL, headers = HEADERS)
③       SOUP = BeautifulSoup(RES.text, 'lxml')
④       DF = pd.read_html(str(SOUP.find("table")), header = 0)[0]
        DF = DF.dropna()
        return DF
⑤   except Exception as e:
        traceback.print_exc()
    return None
```

① 2021년 1월 7일부터 네이버 금융 웹 스크래핑 차단. 따라서 User-Agent 헤더 정보를 입력해야 함.
→ User-Agent Header란? 현재 사용자가 어떤 클라이언트(운영체제, 브라우저 등)를 이용해 요청을 보냈는지 알려줌

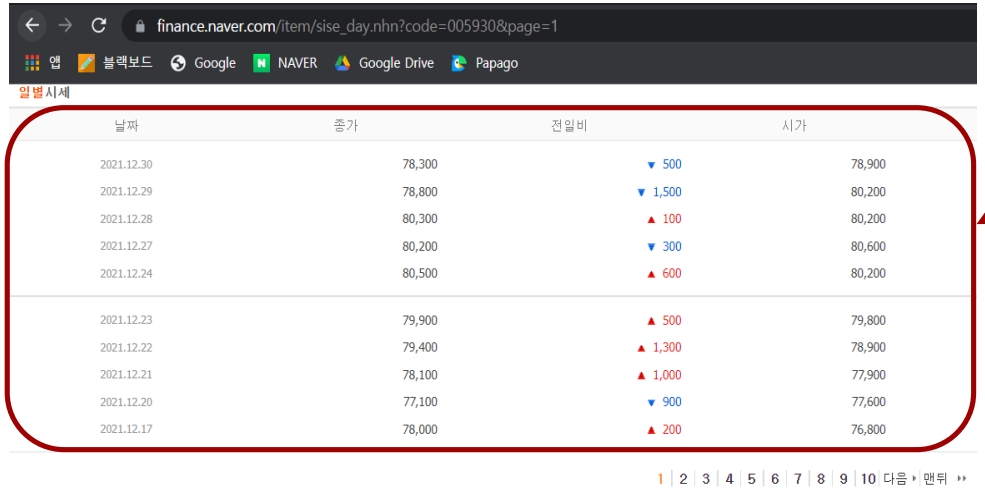
② 해당 URL에 데이터(html) 요청. 이 경우, 요청 시 header에 User-Agent 정보를 크롬으로 지정하여 요청

③ 해당 URL의 html 문자열 파싱. 사용한 html parser는 lxml (lxml은 설치하여 사용 가능. 좀 더 유연하고 빠른 처리 가능)

⑤ 에러에 대한 Stack Trace 출력. 프로그램을 종료시키지 않고 에러에 대한 자세한 로그 출력 가능

필요한 함수 생성 - ① PARSE_PAGE(CODE, PAGE)

네이버 금융 일별 시세 페이지



④ 일별 시세 테이블 가져오기

```
<html lang="ko">
▶ <head>…</head>
▼ <body>
  ▶ <script language="JavaScript">…</script>
  ▶ <h4 class="tlline2">…</h4>
  ▶ <table cellspacing="0" class="type2">…</table> == $0
    <!--- 페이지 네비게이션 시작--->
  ▶ <table summary="페이지 네비게이션 리스트" class="Nnavi" align="center">…</table>
    <!--- 페이지 네비게이션 끝--->
    <script type="text/javascript" src="https://ssl.pstatic.net/imgstock/static.pc/20211216210327/js/jindo.min.ns.1.5.3.euckr.js"></script>
    <script type="text/javascript" src="https://ssl.pstatic.net/imgstock/static.pc/20211216210327/js/lcslog.js"></script>
  ▶ <script type="text/javascript">…</script>
  </body>
</html>
```

필요한 함수 생성 - ② CRAWL_LAST_PAGE(CODE)

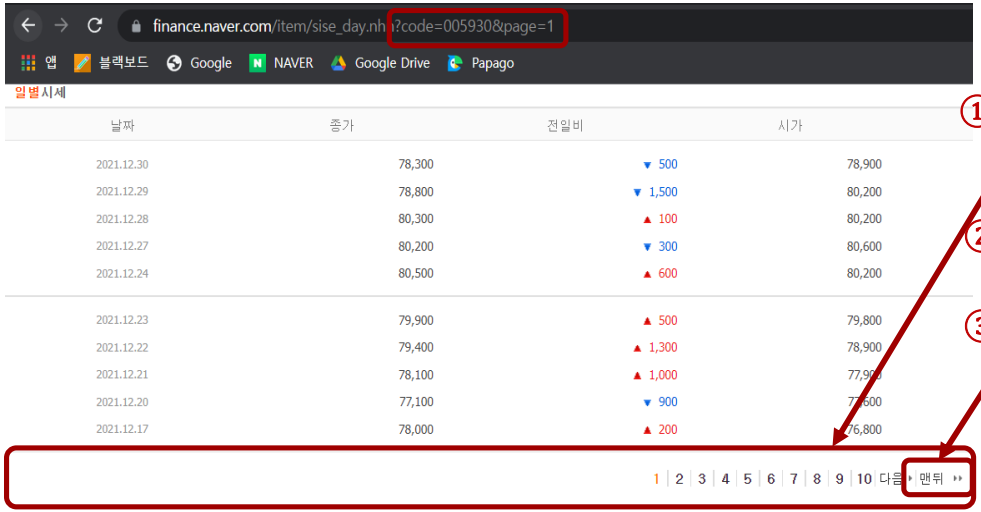
하나의 주식 종목에 대해 일별 시세 마지막 페이지 번호를 확인하는 함수

```
# 하나의 주식에 대해 일별 시세 마지막 페이지를 확인하는 함수

def CRAWL_LAST_PAGE(CODE):
  URL = 'http://finance.naver.com/item/sise_day.nhn?code={code}'.format(code = CODE)
  HEADERS = {'user-agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36'}
  RES = requests.get(URL, headers = HEADERS)
  RES.encoding = 'utf-8'

  SOUP = BeautifulSoup(RES.text, 'lxml')

① PAGE_NAVI = SOUP.find("table", class_="Nnavi")
② TO_LAST_PAGE = PAGE_NAVI.find("td", class_="pgRR")

  # 네이버 금융에 아예 주식 정보가 없는 경우
  if TO_LAST_PAGE == None:
    LAST_PAGE = None
  else:
③   LAST_PAGE = TO_LAST_PAGE.a.get('href').rsplit('&')[1]
    LAST_PAGE = LAST_PAGE.split('=')[1]
    LAST_PAGE = int(LAST_PAGE)
    LAST_PAGE # 마지막 페이지 번호

  return LAST_PAGE
```

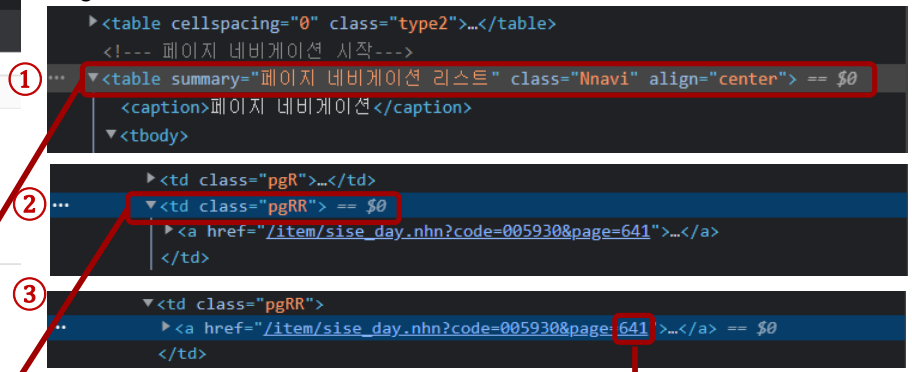
네이버 금융에 아예 주식 정보가 없는 경우,
마지막 페이지 번호는 None으로 설정

필요한 함수 생성 - ② CRAWL_LAST_PAGE(CODE)

네이버 금융 일별 시세 페이지



Pagination 영역 가져오기



필요한 함수 생성 - ③ CRAWL_CLOSINGS_SIMPLE()

하나의 주식 종목에 대해 지정해준 기준일(종가 정보를 얻고자 하는 일자)의 종가를 추출하는 함수 (반복 과정 없음)

```
# 하나의 주식에 대해 지정해준 기준일의 종가를 추출하는 함수 (반복 과정 X)

def CRAWL_CLOSINGS_SIMPLE(YEAR, MONTH, DAY, CODE, PAGE):
  # 기준일 지정
  BASE_DATE = datetime.datetime.strftime(datetime.datetime(year = YEAR, month = MONTH, day = DAY), '%Y.%m.%d')

①DF = PARSE_PAGE(CODE, PAGE)                    # PARSE_PAGE(CODE, PAGE) 함수 사용
②BASE_DF = DF[DF['날짜'] == BASE_DATE]           # 기준일에 해당하는 데이터만 필터링

③# 기준일에 해당하는 데이터가 없는 경우 (→ 후처리 해줄 것)
  if len(BASE_DF) == 0:
    CLOSINGS = None

  # 추출된 종가
  else:
    CLOSINGS = int(BASE_DF['종가'])

  return CLOSINGS
```

기준일 이전이나 이후부터 일별 시세가 기록되었거나 중간에 결측치가 있는 종목을 제외한다면, 일반적으로 주식 개장일에 매일 일별 시세 정보가 기록된 종목들은 기준일이 존재하는 페이지가 다 같음 → 이 경우에 해당되는 종목들에 대해선, 굳이 이전 페이지들까지 크롤링할 필요 없음. 해당 페이지만 크롤링하여 시간 단축하기.

① 종가 정보를 얻고자 하는 일자, 즉 기준일이 일반적으로 존재하는 페이지 번호 지정하여 PARSE_PAGE(CODE, PAGE) 함수 사용

② 기준일에 해당하는 데이터만 추출

③ 기준일에 해당하는 데이터가 없는 경우, 일단 종가는 None으로 설정. 이후 CRAWL_CLOSING() 함수를 통해 첫번째 페이지부터 마지막 페이지까지 반복해서 찾아본 후, 기준일에 해당하는 데이터 추출하여 종가 지정. 그래도 기준일에 해당하는 데이터가 없는 경우에는 종가 정보 None으로 지정.

필요한 함수 생성 - ④ CRAWL_CLOSINGS ()

일반적으로 기준일이 존재하는 페이지에 기준일 데이터가 존재하지 않는 경우.

하나의 주식 종목에 대해 지정해준 기준일(종가 정보를 얻고자 하는 일자)의 종가를 추출하는 함수 (반복 과정 있음)

```
# 하나의 주식에 대해 지정해준 기준일의 종가를 추출하는 함수 (반복 과정 O)

def CRAWL_CLOSINGS(YEAR, MONTH, DAY, LAST_PAGE, CODE):

  # 기준일 지정
  BASE_DATE = datetime.datetime.strftime(datetime.datetime(year = YEAR, month = MONTH, day = DAY), '%Y.%m.%d')

①# 네이버 금융에 아예 주식 정보가 없는 경우
  if LAST_PAGE == None:
    CLOSINGS = None

②else:
    # 기준일 이전 데이터밖에 없는 경우
    DF_TRY = PARSE_PAGE(CODE, 1)
    if len(DF_TRY[DF_TRY['날짜'] >= BASE_DATE]) == 0:
      CLOSINGS = None

③  else:
      # 기준일 이후 데이터밖에 없는 경우
      DF_TRY_LAST = PARSE_PAGE(CODE, LAST_PAGE)
      if len(DF_TRY_LAST[DF_TRY_LAST['날짜'] <= BASE_DATE]) == 0:
        CLOSINGS = None

④    # 마지막 페이지에 기준일 데이터가 있는 경우
    elif len(DF_TRY_LAST[DF_TRY_LAST['날짜'] == BASE_DATE]) == 1:
      CLOSINGS = int(DF_TRY_LAST['종가'])

⑤  else:
    BASE_DF = None
    for page in range(1, LAST_PAGE + 1):                # 1 페이지부터 마지막 페이지까지 반복
      DF = PARSE_PAGE(CODE, page)                       # PARSE_PAGE(CODE, PAGE) 함수 사용
      DF_filtered = DF[DF['날짜'] == BASE_DATE]         # 기준일에 해당하는 데이터만 필터링

      if BASE_DF is None:
        BASE_DF = DF_filtered
      else:
        BASE_DF = pd.concat([BASE_DF, DF_filtered])
⑥   if len(BASE_DF) == 1:
      break

  if len(BASE_DF) == 0 :
    CLOSINGS = None

  else:
    CLOSINGS = int(BASE_DF['종가'])

return CLOSINGS
```

① 네이버 금융에 아예 주식 정보가 없는 경우, 종가는 None으로 설정

② 첫번째 페이지의 날짜 목록 중에서 기준일보다 미래인 날짜가 0개이면, 즉, 기준일 이전 데이터밖에 없으면, 종가는 None으로 설정

③ 마지막 페이지의 날짜 목록 중에서 기준일보다 과거인 날짜가 0개이면, 즉, 기준일 이후 데이터밖에 없으면, 종가는 None으로 설정

④ 마지막 페이지에 기준일이 있다면, 굳이 앞 페이지들을 크롤링할 필요 없이, 시간 단축을 위해 바로 종가 지정

⑤ 앞 4개의 케이스에 해당하지 않는 경우, 첫번째 페이지부터 ~ 마지막 페이지까지 반복하면서 날짜가 기준일과 일치하는 데이터만 추출

⑥ 일치하는 데이터는 1개일 것이므로, 데이터프레임의 길이가 1일 때 반복을 멈추고 해당 데이터의 종가를 추출 데이터프레임의 길이가 0이라면 종가는 None으로 설정

종가 크롤링

종가 정보를 추출하고자 하는 종목들의
종목 코드 리스트 생성

```
# 종목코드 리스트 생성
CODE_LIST = []
for i in range(len(train['종목코드'])):
  CODE = train['종목코드'][i][1:]
  CODE_LIST.append(CODE)
```

```
# 중복되지 않은 종목코드 리스트만 추출
UNIQUE_CODE_LIST = []

for code in CODE_LIST:
  if code not in UNIQUE_CODE_LIST:
    UNIQUE_CODE_LIST.append(code)
```

해당 종목 코드 리스트에 대해
Ex) 2019년 12월 30일의 종가 추출

(1) CRAWL_CLOSINGS_SIMPLE() 함수 사용 (반복 X)

```
# 2019년 12월 30일 종가 추출

UNIQUE_CLOSINGS_LIST = []
for code in UNIQUE_CODE_LIST:
  CODE = code
①PAGE = 43
②CLOSINGS = CRAWL_CLOSINGS_SIMPLE(2019, 12, 30, CODE, PAGE)
  UNIQUE_CLOSINGS_LIST.append(CLOSINGS)
```

① 2019-12-30은 대개 페이지 43에 존재 (크롤링 당시 기준)

② CRAWL_CLOSINGS_SIMPLE() 함수 이용하여 해당 종목코드들의 종가 추출

종가 크롤링

처리 가능한 결측치에 대해 종가 추출 (2) CRAWL_CLOSINGS 함수 사용 (반복 O)

```
UNIQUE_MISSING_INDEX = list(UNIQUE_CLOSINGS_LIST_DF[UNIQUE_CLOSINGS_LIST_DF['종가_191230'].isnull()].index)

UNIQUE_MISSING_CODE_LIST = []
for i in UNIQUE_MISSING_INDEX:
  UNIQUE_MISSING_CODE_LIST.append(UNIQUE_CODE_LIST[i])
```

결측치가 존재하는 종목 코드 리스트 생성
(즉, 종가 정보가 없는 종목 코드 리스트)
→ UNIQUE_MISSING_CODE_LIST

```
# 2019년 12월 30일 종가 추출

UNIQUE_MISSING_CLOSINGS_LIST = []
for code in tqdm(UNIQUE_MISSING_CODE_LIST):
  sleep(0.1)
  CODE = code
  LAST_PAGE = CRAWL_LAST_PAGE(CODE)
  UNIQUE_MISSING_CLOSINGS = CRAWL_CLOSINGS(2019, 12, 30, LAST_PAGE, CODE)
  UNIQUE_MISSING_CLOSINGS_LIST.append(UNIQUE_MISSING_CLOSINGS)
```

UNIQUE_MISSING_CODE_LIST 내의
종목코드들에 대하여
CRAWL_CLOSINGS() 함수를 이용하여
기준일 데이터가 존재하는 경우,
첫번째부터 마지막 페이지까지 반복하여
기준일 데이터 필터링한 뒤 종가 추출

```
j=0
for i in UNIQUE_MISSING_INDEX:
  UNIQUE_CLOSINGS_LIST_DF['종가_191230'][i] = UNIQUE_MISSING_CLOSINGS_LIST[j]
  # UNIQUE_CLOSINGS_LIST[i] = UNIQUE_MISSING_CLOSINGS_LIST[j]
  j += 1
```

UNIQUE_MISSING_CODE_LIST 내
종목들의 기준일 종가 정보를
기존 전체 종목들의 기준일 종가 정보
데이터 프레임에 합치기

	종가_191230
0	31050.0
1	55800.0
2	8900.0
3	6630.0
4	30000.0
...	...
2779	13205.0
2780	NaN
2781	NaN
2782	NaN
2783	NaN

2784 rows × 1 columns

최종 2019년 12월 30일 종가 데이터

종가 크롤링

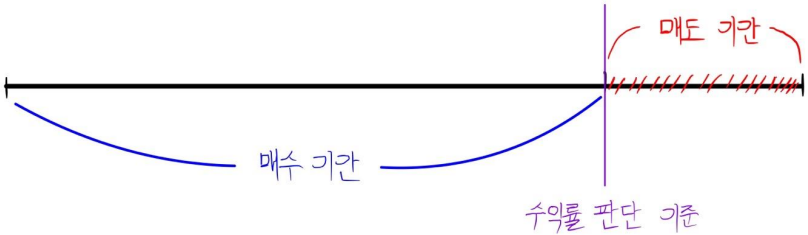
위 과정을 종가 정보를 얻고자 하는 모든 기준일에 대하여 반복하여,
Train/Test 데이터셋 내의 모든 중복되지 않은 종목 코드들에 대하여 종가 크롤링

```
종가_ALL_3M = pd.concat([UNIQUE_CODE_LIST_DF, 종가_160331, 종가_160630, 종가_160930, 종가_161229, 종가_170331, 종가_170630, 종가_170929, 종가_171228,
                        종가_180330, 종가_180629, 종가_180928, 종가_181228, 종가_190329, 종가_190628, 종가_190930, 종가_191230,
                        종가_200331, 종가_200630, 종가_200929, 종가_201230], axis = 1)
종가_ALL_3M
```

	종목코드	종가_20160331	종가_160630	종가_20160930	종가_161229	종가_20170331	종가_170630	종가_20170929	종가_171228	종가_20180330	종가_180629	종가_20180928	종가_181228	종가_20190329	종가_190628	종가_20190930	종가_191230
0	A006360	27300.0	27750.0	29350.0	26500.0	30700.0	30450.0	26650.0	28300.0	29600.0	46000.0	52300.0	43750.0	42600.0	40350.0	33000.0	31050.0
1	A005930	1312000.0	1425000.0	1598000.0	1802000.0	2060000.0	2377000.0	2564000.0	2548000.0	2461000.0	46650.0	46450.0	38700.0	44650.0	47000.0	49050.0	55800.0
2	A005070	2570.0	2980.0	6010.0	5160.0	4505.0	7700.0	15100.0	15450.0	17100.0	21300.0	25000.0	17350.0	14600.0	13450.0	8210.0	8900.0
3	A003520	3515.0	11700.0	12400.0	8820.0	8170.0	11350.0	9690.0	8780.0	8260.0	7430.0	8430.0	5940.0	6660.0	5070.0	4680.0	6630.0
4	A002310	17600.0	20500.0	20850.0	18300.0	18850.0	20450.0	19300.0	18200.0	32500.0	34550.0	42900.0	31400.0	43600.0	38500.0	29700.0	30000.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2921	A302440	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2922	A272850	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2923	A241390	NaN	9905.0	9940.0	9825.0	9970.0	10430.0	10330.0	10575.0	10395.0	10275.0	10380.0	10130.0	10535.0	10560.0	10415.0	10615.0
2924	A153760	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2925	A35732K	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

2926 rows × 21 columns

주식 종가 정보 크롤링



Ex) 2020년 07월 21일에 구매한 주식을 2021년 언제 매도할지 예측
- 사용 가능한 정보는 2020년 12월 31일까지
- 이때, 2020년 12월 31일까지의 수익률은 굉장히 중요한 정보!

Why...? 수익률이 굉장히 높거나 낮으면 일반적으로 쉽게 매도하지 않으니까

주식 매수 당시 구매 금액과 수익률 판단 기준일자 종가를 비교하여 '수익률' feature 추가

Rmse 값

주식 개장일 정보 포함, 배당 idea 적용하여 기간 분할, 변수 타입에 맞는 전처리, 수익률 feature 추가

dacon_초안.csv edit	2021-09-07 20:02:12	82.0114459908 -
dacon_주식 보유기간 예측_09.14.csv edit	2021-09-14 20:32:55	71.2191455649 -
dacon_주식 보유기간 예측_09.25(feature 추가& 명목변수 전처리).csv edit	2021-09-25 17:28:01	67.5932279678 -

RMSE값 4 추가 하락, Dacon Private 기준 10위권 중반, 안정적인 본선 진출권

파생 변수 제작

변수들 간에 맞물리는 정보들과 관계성으로 새로운 feature를 만들 수 있었음

1. 잔고 금액, 총 투자 기간 -> 잔고금액 / 총 투자기간 feature로 변경

2. 고객 특성을 고려해 새로운 변수 조합 만들어보기

: Life Stage & 자산 구간 조합으로 새로운 feature 도전

Past_d

주식 매수 / 매도 데이터에서 고객 정보를 뽑아내어

특정 고객이 과거에 특정 종목을 보유한 기간 추출

특정 고객, 특정 종목에 있어서 매수 및 매도가 한번만 이뤄진 것이 아님!
: 많은 경우 복수의 매수 및 매도 거래가 이뤄졌었음.

따라서 (매수, 매도) 쌍을 기준으로 past_d를 제작

복수 거래의 평균 보유 일자로 past_d를 계산
(앞에서 제작한 주식 시장 개장일 함수 사용)

(단 이때, 해당 종목을 일부 보유한 상태에서 추가 매수를 하거나 전량 매도를 하지 않은 경우는 제외)

Past_d

주식 매수 / 매도 데이터에서 고객 정보를 뽑아내어

특정 고객이 과거에 특정 종목을 보유한 기간 추출

<복수의 거래 기록을 어떻게 처리했는가>

Past_d= hist_d
: pass

Past_d != hist_d & 전량 매도 기록은 없는 경우
: pass

Past_d != hist_d & 전량 매도 기록이 있는 경우
: 보유기간 평균

```
## past_d 제작 코드 ##

for i in range(len(test_1)):
  a=test_1['act_id'][i]
  b=test_1['iem_cd'][i]
  selected_test=test_1[(test_1['act_id']==a) & (test_1['iem_cd']==b)]
  selected_hist=hist_1[(hist_1['act_id']==a) & (hist_1['iem_cd']==b)].sort_values(by=['bse_dt'], axis=0)

  # hist기록이 곧 train 기록인 경우, past_d=hold_d
  if (selected_hist.shape[0] < 4):
    pass
  # train 매수 시점 이전에 전량매도가 없는 경우
  else:
    zero_hist=selected_hist[selected_hist['bnc_qty']==0]
    if zero_hist.shape[0] < 2:
      pass
    # train 매수 시점이전에 전량매도가 있는 경우
    elif zero_hist.shape[0] > 1:
      zero=[]
      first=[]
      first.append(selected_hist['bse_dt'].iloc[0])

      for a in range(len(selected_hist)-1):
        if selected_hist['bnc_qty'].iloc[a]==0:
          zero.append(selected_hist['bse_dt'].iloc[a])
          first.append(selected_hist['bse_dt'].iloc[a+1])

      del first[-1]
```

추천 시스템의 아이디어

```python
# knn collab filtering 실행 코드

def cf_knn(user_id, inv_code, neighbor_size=0):
    if inv_code in hold_matrix:
        sim_scores = user_similarity[user_id].copy()
        hold = hold_matrix[inv_code].copy()
        none_rating_idx = hold[hold.isnull()].index
        hold = hold.drop(none_rating_idx)
        sim_scores = sim_scores.drop(none_rating_idx)

        if neighbor_size == 0:
            mean_hold = np.dot(sim_scores, hold) / sim_scores.sum()

        else:
            if len(sim_scores) > 1:
                neighbor_size = min(neighbor_size, len(sim_scores))
                sim_scores = np.array(sim_scores)
                hold = np.array(hold)
                user_idx = np.argsort(sim_scores)
                sim_scores = sim_scores[user_idx][-neighbor_size:]
                hold = hold[user_idx][-neighbor_size:]
                mean_hold = np.dot(sim_scores, hold) / sim_scores.sum()
            else:
                mean_hold = 10
    else:
        mean_hold = 10
    return mean_hold
```

사용자 특성의 데이터와 종목의 데이터로 구분 가능

이는 일반적인 추천시스템의 데이터셋과 유사

추천시스템 알고리즘 중 KNN collaborative filtering을 적용

Rmse 값

1. 파생변수 제작 : rmse값 변동 존재

But, feature selection으로 최소한의 feature를 적용시킨 것이 제출 점수는 더 좋았음

2. 추천시스템 방식
: 유저가 특정 아이템을 사용했던 기록이 없으면 내부에서 다시 예측의 과정을 거쳐야해 오히려 성능이 떨어지는 현상

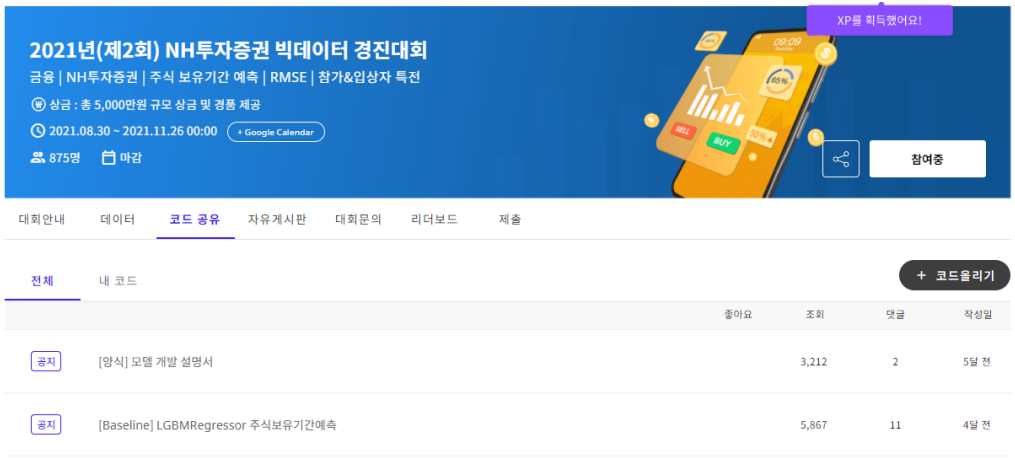
후반부 시도

후반부에 도전했던 시도들이 rmse값을 더 이상 낮추지 못함

본선 진출이 어렵다고 판단, 시험에 집중하기로 합의

2021-10-07 10:42:35 67.4813980198

코드 리뷰



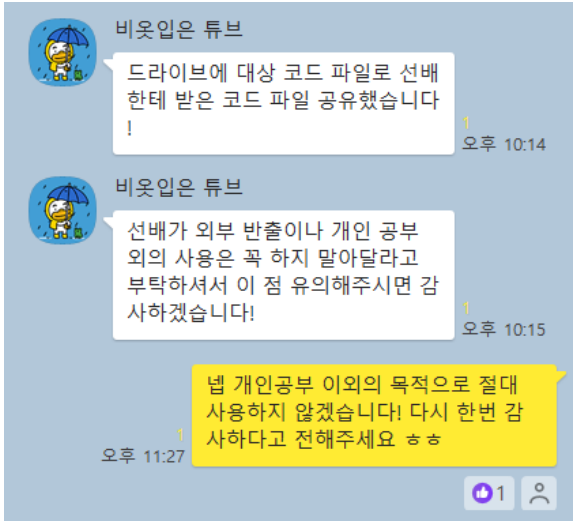
증권사의 실제 데이터를 사용하는 대회

철저한 데이터 관리

수상 이후에도 코드 리뷰가 올라오지 않았음

코드 리뷰

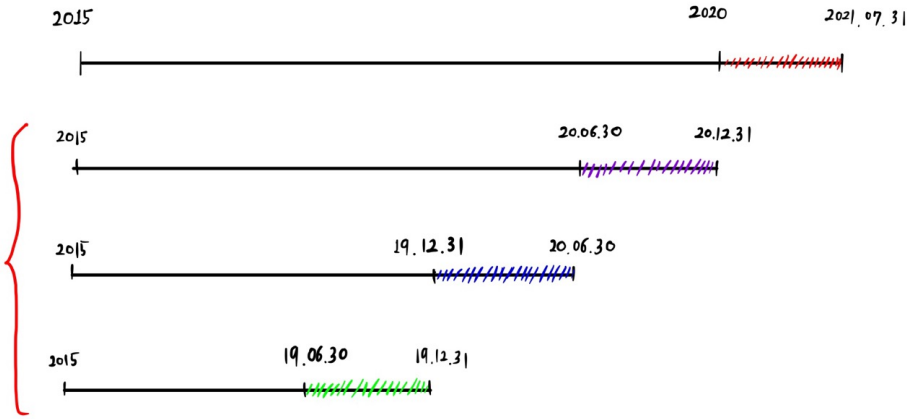
AVCD Mo Dd Ju 56.66855



대상을 수상한 팀과 연락이 닿아 감사하게도
공부 목적으로 코드를 공유받을 수 있었음

코드 리뷰를 통해 보완점 및 실수한 곳 파악

보완점



- 전처리 이후 다시한번 EDA

- Smote나 Adasyn을 통한 불균형 데이터 처리

- 단기 트레이딩 거래가 훨씬 많을 수 밖에 없음

얻어간 점

- “당연한걸 놓치지 말자...” 는 교훈

- 금융 도메인 지식을 최대한 적용해봤다는 점

: 실제로 성능을 향상시켰던 전처리는 전부 도메인 지식을 적용한 경우

- 각자 2개씩 포트폴리오
  - NH 증권 주식 보유기간 예측
  - 개인 신용 등급 분류 예측

금융 프로젝트

팀 여의도

- NH 증권 주식 보유기간 예측
- 개인 신용 등급 분류 예측