6조 업빛투

[Code Definition]

정길종(팀장), 김형림, 윤보람, 인태우, 채길호

[Data Science]

작성일자	2021-10-06	작성자	채길호

1. 데이터 로드 및 전처리

주식 데이터 및 재무지표 로드 함수

- 패키지 활용

```
def money_sur(stock_name, start, end):

    start_date = start
    end_date = end
    sample_code = stock_name

    dataset = fdr.DataReader(sample_code, start = start_date, end = end_date )
    dataset = dataset.reset_index()
    df = stock.get_market_fundamental_by_date(fromdate=start_date, todate=end_date, ticker=sample_code)
    df = df.reset_index()
    df.columns = ['Date','BPS', 'PER', 'PBR', 'EPS', 'DIV', 'DPS']
    data = pd.merge(dataset, df, on='Date')

    return data
```

→ 주식 데이터는 FinanceDataReader 패키지를 통해서 해당 기간의 종목별 시가, 종가 등의 데이터를 가져오고, 재무지표의 경우 PyKrx 패키지를 통해서 BPS, PER, PBR 등의 지표 데이터를 가져온다.

- 실제 로드

```
# 삼성전자 2018년 이후
end_date = datetime.datetime.now().strftime("%Y%m%d")
samsung_df = money_sur('005930', start = '2018', end = end_date)

samsung_df
```

텍스트 데이터 로드 함수

- 긍·부정 텍스트 목록 및 KOSELF 감성사전, 불용어 사전 로드

```
# 블로그에서 가져온 기본적인 한국어 긍부정 텍스트 목록
with open('/content/drive/My Drive/Final PJT - 업빛투/감성분석/data/positive_words_self.txt', encoding='utf-8') as pos_blog:
    positive_blog = pos_blog.readlines()
positive_blog = [pos_blog.replace('₩n', '') for pos_blog in positive_blog]
with open('/content/drive/My Drive/Final PJT - 업빛투/감성분석/data/negative_words_self.txt', encoding='utf-8') as neg_blog:
    negative_blog = neg_blog.readlines()
negative_blog = [neg_blog.replace('₩n', '') for neg_blog in negative_blog]

# KOSELF 감성 어휘 사전
with open('/content/drive/My Drive/Final PJT - 업빛투/감성분석/data/KOSELF_pos.txt', encoding='utf-8') as pos:
    positive = pos.readlines()
positive = [pos.replace('₩n', '') for pos in positive]
with open('/content/drive/My Drive/Final PJT - 업빛투/감성분석/data/KOSELF_neg.txt', encoding='utf-8') as neg:
    negative = neg.readlines()
negative = [neg.replace('₩n', '') for neg in negative]
```

```
url = 'https://raw.githubusercontent.com/chaerui7967/stock_predict_news_and_youtube/master/Sentiment_Analysis/data/stopwords_ver1.txt'
stopwords = list(pd.read_csv(url, header=None)[0])
```

→ TXT 파일을 로드 후 리스트로 전환

- DB에서 데이터 로드

```
def news_db(news, stock):
    db = pymysql.connect(
            user='root',
            passwd='1234',
            host='3.35.70.166',
            db='proj',
            charset = 'utf8'
        )

    cursor = db.cursor(pymysql.cursors.DictCursor)
    news1 = news + '_news_craw'

    sql = "select * from {0}_{1} where (length(date)=10) and (date between {2}00 and {3}23)".format(news1, stock, start, end)
    cursor.execute(sql)
    result = cursor.fetchall()
```

→ 미리 크롤링한 데이터를 로드 해서 사용

- 휴장일 데이터 로드

```
### 4) Holidays
db = pymysql.connect(user='root',
                     passwd='1234',
                     host='3.35.70.166',
                     db='proj',
                     charset='utf8')

cursor = db.cursor(pymysql.cursors.DictCursor)

# 4-1) 주말 및 공휴일 데이터
sql = "select * from holidays"
cursor.execute(sql)
result = cursor.fetchall()
```

데이터 전처리

- 휴장일 처리

```
## 1-2) 전일 15시 ~ 금일 15시로 날짜 조정
after_market = ['15', '16', '17', '18', '19', '20', '21', '22', '23']

for j in range(len(df['time'])):
    if df['time'][j] in after_market:
        df['date'][j] += datetime.timedelta(1)
    else:
        pass
```

```
### 2) 주말 및 공휴일 제외

## 2-1) 주말 및 공휴일만 추출
market_closed =holidays[holidays['holiday']=="0"].reset_index(drop=True)

## 2-3) 휴장일 List 생성
market_closed_list = list(market_closed['date'])

## 2-4) iteration limit 조정
limit_number = 15000
sys.setrecursionlimit(limit_number)


while len(df[df['date'].isin(market_closed_list)]['date']) != 0:
    for hoil in df[df['date'].isin(market_closed_list)]['date'].index:
        df['date'][hoil] += datetime.timedelta(1)
```

→ 휴장일과 개장시간을 고려하여 (기사·영상의) 업로드 시간에 따라 날짜를 조정

텍스트 토큰화

- 텍스트 토큰화 및 불용어 처리

```
df['Tokenization'] = 0
rows = df.shape[0]
for row in range(rows):
    hangeul = re.compile('[^ ㄱ-ㅣ 가-힣]')
    result = hangeul.sub('', df['text'][row])   # 위0
    okt = Okt()
    nouns = okt.nouns(df['text'][row])
    nouns = [x for x in nouns if len(x) > 1]
    nouns = [x for x in nouns if x not in stopwords]

    corpus = " ".join(nouns)
    df['Tokenization'][row] = corpus
```

→ 한글 텍스트만 남긴 후 KoNLPy 패키지를 통해 명사만 추출

→ 단어 길이가 1 이하인 단어는 삭제하고, 불용어 사전으로 불용어 처리

2. 사용 모델 코드

ARIMA

- 파라미터 조정

→ 로그 변환과 차분 과정

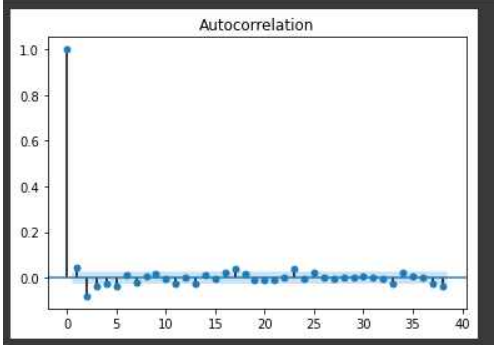
```
x = df1['Price'].values
x = np.log(x)

diff = x[1:] - x[:-1] # 차분
```

→ 차분 후 정상성 검정(ADF 검정)

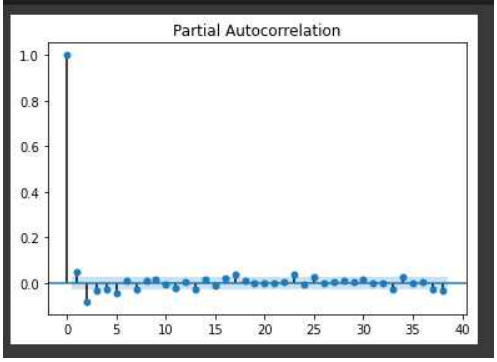
```
from statsmodels.tsa.stattools import adfuller
result = adfuller(df1['Price'])

print(f'ADF statistic: {result[0]}')
print(f'p-value: {result[1]}')
```

→ 적절한 AR, MA값 구하기

```
fig = plot_acf(diff)
```




```
fig = plot_pacf(diff)
```



→ ACF, PACF 그래프를 확인하여 적절한 AR, MA값 도출

- 조정 후 모델 적합

```
# const 가 유의하지 않으므로 trend를 nc로해서 다시 모델 적합
model = ARIMA(df1.Price.values, order = (2,1,2))
model_fit = model.fit(trend = 'nc', full_output = True, disp = True)
print(model_fit.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                   D.y   No. Observations:                  923
Model:                 ARIMA(2, 1, 2)   Log Likelihood               -7632.184
Method:                       css-mle   S.D. of innovations            943.104
Date:                Sat, 02 Oct 2021   AIC                          15274.368
Time:                        01:38:23   BIC                          15298.506
Sample:                             1   HQIC                         15283.577

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1.D.y     -1.4708      0.024    -60.080      0.000      -1.519      -1.423
ar.L2.D.y     -0.9602      0.029    -32.785      0.000      -1.018      -0.903
ma.L1.D.y      1.4915      0.017     89.611      0.000       1.459       1.524
ma.L2.D.y      0.9931      0.020     50.455      0.000       0.955       1.032
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -0.7659           -0.6744j            1.0205           -0.3851
AR.2           -0.7659           +0.6744j            1.0205            0.3851
MA.1           -0.7510           -0.6656j            1.0035           -0.3846
MA.2           -0.7510           +0.6656j            1.0035            0.3846
```

→ 위에서 구한 파라미터를 조정하여 최종 모델을 만들고 훈련 데이터 셋을 적합

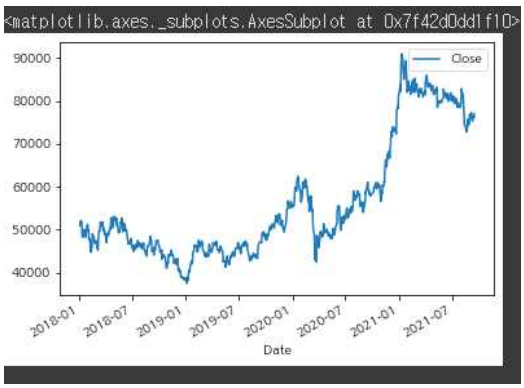
- 예측

```
fore = model_fit.forecast(steps=10) # 10일 예측
print(fore)
```

→ 이후 10일 동안의 종가를 예측

FBProphet

- 파라미터 조정



→ 주식 그래프를 확인한 결과 진폭이 점점 증가하는 그래프이므로 seasonality_mode는 'multiplicative'로 결정, changepoint_prior_scale의 경우 0.5, 0.6, 0.7을 적용한 결과 0.6의 결과가 오차가 제일 적게 나옴을 확인

- 파라미터 조정 후 모델 적합

```
from fbprophet import Prophet

prophet = Prophet(seasonality_mode = 'multiplicative',
                  yearly_seasonality=True,
                  weekly_seasonality=True,
                  daily_seasonality=True,
                  changepoint_prior_scale=0.6)

prophet.fit(train)
```

→ 파라미터를 조정하여 모델 적합

- 예측

```
# 10일단위로 예측값을 가져옴
future_data = prophet.make_future_dataframe(periods = 10, freq = 'd')
forecast_data = prophet.predict(future_data)
forecast_data[['ds','yhat', 'yhat_lower', 'yhat_upper']].tail(10)
```

→ 10일 단위로 예측값을 가져오고, 예측값 목록은 날짜, 예측값, 예측 하한, 상한값으로 이루어짐

LSTM

- 데이터 전처리

```
from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler()

train_sc = sc.fit_transform(train)
test_sc = sc.transform(test)

train_sc
```

```
for i in scale_cols:
  for s in range(1, 11):
    train_sc_df[(i+'shift_{}'.format(s),)] = train_sc_df[(i,)].shift(s)
    test_sc_df[(i+'shift_{}'.format(s),)] = test_sc_df[(i,)].shift(s)

train_sc_df.head(11)
train_sc_df.shape
```

→ 스케일링 및 window_size를 10일 기준으로 하여 데이터 전처리 진행

- layer 확인

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_18 (LSTM)               (None, 10, 64)            19712
_________________________________________________________________
dropout_17 (Dropout)         (None, 10, 64)            0
_________________________________________________________________
dense_23 (Dense)             (None, 10, 32)            2080
_________________________________________________________________
lstm_19 (LSTM)               (None, 10, 64)            24832
_________________________________________________________________
dropout_18 (Dropout)         (None, 10, 64)            0
_________________________________________________________________
dense_24 (Dense)             (None, 10, 32)            2080
_________________________________________________________________
lstm_20 (LSTM)               (None, 64)                24832
_________________________________________________________________
dropout_19 (Dropout)         (None, 64)                0
_________________________________________________________________
flatten_4 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_25 (Dense)             (None, 100)               6500
_________________________________________________________________
dense_26 (Dense)             (None, 32)                3232
_________________________________________________________________
dense_27 (Dense)             (None, 1)                 33
=================================================================
Total params: 83,301
Trainable params: 83,301
Non-trainable params: 0
_________________________________________________________________
```

→ Dense층과 Dense층을 번갈아가면서 진행하고, 과적합을 방지하기 위하여 중간중간 Dropout을 시행
→ 층의 activation function은 ReLU와 tanh를 사용

- compiler 조정

```
loss = Huber()
optimizer = Adam(0.0005)
model.compile(loss=Huber(), optimizer=optimizer, metrics=['mse'])
```

→ Loss 함수의 경우 Adam과 비교하여 조금 더 좋은 성능을 보이는 Huber를 사용하였으며, 평가는 MSE로 진행

- 모델 적합

```
# earlystopping은 30번 epoch동안 val_loss 개선이 없다면 학습 중지
earlystopping = EarlyStopping(monitor='val_mse', patience=100)
# val_loss 기준 체크포인터도 생성
filename = 'lstm_samsung.h5'
checkpoint = ModelCheckpoint(filename,
                             save_weights_only=True,
                             save_best_only=True,
                             monitor='val_mse',
                             verbose=1)
```

```
history = model.fit(X_train_t, y_train, validation_data=(X_test_t,y_test),
                    epochs=1000,
                    callbacks=[checkpoint, earlystopping])
```

```
Epoch 00002: val_mse improved from 0.42782 to 0.33366, saving model to lstm_samsung.h5
Epoch 3/1000
23/23 [==============================] - 0s 19ms/step - loss: 0.0081 - mse: 0.0162 - val_loss: 0.1156 - val_mse: 0.2312

Epoch 00003: val_mse improved from 0.33366 to 0.23125, saving model to lstm_samsung.h5
Epoch 4/1000
23/23 [==============================] - 0s 19ms/step - loss: 0.0057 - mse: 0.0113 - val_loss: 0.0442 - val_mse: 0.0884

Epoch 00004: val_mse improved from 0.23125 to 0.08838, saving model to lstm_samsung.h5
Epoch 5/1000
23/23 [==============================] - 0s 19ms/step - loss: 0.0047 - mse: 0.0095 - val_loss: 0.0272 - val_mse: 0.0543

Epoch 00005: val_mse improved from 0.08838 to 0.05434, saving model to lstm_samsung.h5
```

→ 1,000 Epochs를 돌렸으나, checkpoint와 earlystopping을 지정하여 최적의 값이 나오면 모델을 저장 후 정지

RL(강화 학습)

- 환경설정

```
# ENV 설정
class Environment1:
    def __init__(self, data, history_t=90):
        self.data = data
        self.history_t = history_t
        self.reset()

    def reset(self):
        self.t = 0
        self.done = False
        self.profits = 0
        self.positions = []
        self.position_value = 0
        self.history = [0 for _ in range(self.history_t)]
        return [self.position_value] + self.history # obs
```

→ 모델에 적합할 수 있도록 환경을 설정

```
reward = 0

# action
# 0: Idle
# 1: 매수
# 2: 매도
if act == 1: #매수
    self.positions.append(self.data.iloc[self.t, :]['Close'])
elif act == 2: # 매도
    if len(self.positions) == 0:
        reward = -1
    else:
        profits = 0
        for p in self.positions:
            profits += (self.data.iloc[self.t, :]['Close'] - p)
        reward += profits
        self.profits += profits
        self.positions = []

# set next time
self.t += 1

self.position_value = 0
for p in self.positions:
    self.position_value += (self.data.iloc[self.t, :]['Close'] - p)
self.history.pop(0)
self.history.append(self.data.iloc[self.t, :]['Close'] - self.data.iloc[(self.t-1), :]['
if (self.t==len(self.data)-1):
    self.done=True
# clipping reward
if reward > 0:
    reward = 1
elif reward < 0:
    reward = -1
#print ("t={%d}, done={%str}"%(self.t,self.done))
return [self.position_value] + self.history, reward, self.done # obs, reward, done
```

→ 모델의 step당 reward 설정

- Modeling

```
# MODEL
class Q_Network(nn.Module):
    def __init__(self,obs_len,hidden_size,actions_n):
        super(Q_Network,self).__init__()
        self.fc_val = nn.Sequential(
            nn.Linear(obs_len, hidden_size),
            nn.ReLU(),
            nn.Linear(hidden_size, hidden_size),
            nn.ReLU(),
            nn.Linear(hidden_size, actions_n)
        )
    def forward(self,x):
        h = self.fc_val(x)
        return (h)
```

→ 모델 설정을 Linear와 ReLU를 반복하는 layer로 구성

- 학습

```
# 학습 시키기
env = Environment1(train)
env.reset()

hidden_size = 100
input_size = env.history_t+1
output_size = 3
USE_CUDA = False
LR = 0.001

Q = Q_Network(input_size, hidden_size, output_size)
Q_ast = copy.deepcopy(Q)

if USE_CUDA:
    Q = Q.cuda()
loss_function = nn.MSELoss()
optimizer = optim.Adam(list(Q.parameters()), lr=LR)

epoch_num = 50
step_max = len(env.data)-1
memory_size = 200
batch_size = 50
gamma = 0.97
```

→ 데이터 환경을 조정한 후에 모델에 적합

뉴스 기사 긍·부정 라벨링

- 긍·부정 점수 산출

```
# 점수 산출

df['Positive_Score'] = 0
df['Negative_Score'] = 0
df['Ratio'] = 0.1
df['Pred'] = 0
df['NSI'] = 0.1

for score in range(len(df)):
    pos_score = 0 ; neg_score = 0

    for token in range(len(df['Tokenization'][score].split())):
        if df['Tokenization'][score].split()[token] in positive:
            pos_score += 1
        elif df['Tokenization'][score].split()[token] in negative:
            neg_score += 1
        else:
            pass

    df['Positive_Score'][score] = pos_score
    df['Negative_Score'][score] = neg_score

    # 긍정과 부정의 비율
    if (pos_score==0) and (neg_score==0):
        df['Ratio'][score] = 0.5   # 둘 다 0일 경우에는 긍정으로 가정
    else:
        df['Ratio'][score] = pos_score / (pos_score + neg_score)

    # 예측 결과
    if df['Ratio'][score]>=0.5:
        df['Pred'][score] = 1
    else:
        df['Pred'][score] = -1
```

→ 구축된 사전을 통해서 긍정점수와 부정점수 및 긍정 비율을 산출

- 뉴스심리지수(NSI)

```
# 뉴스심리지수(NSI) 계산
if (pos_score==0) and (neg_score==0):
    df['NSI'][score] = 101
else:
    df['NSI'][score] = (pos_score - neg_score) / (pos_score + neg_score) * 100 + 100
```

→ 문장 수를 단어 수로 변환하여 기존 NSI 산출 공식에 동일하게 적용

YouTube 스크립트 긍·부정 라벨링

- 훈련 데이터 로드

```
urllib.request.urlretrieve("https://raw.githubusercontent.com/e9t/nsmc/master/ratings_train.txt", filename="ratings_train.txt")
urllib.request.urlretrieve("https://raw.githubusercontent.com/e9t/nsmc/master/ratings_test.txt", filename="ratings_test.txt")
```

→ YouTube 특성상 맞춤법이 맞지 않고 대화식으로 진행될 것으로 판단하여, 라벨링된 데이터 중 네이버 영화 리뷰 데이터를 활용

- 전처리 및 불용어 처리

```
test_data.drop_duplicates(subset = ['document'], inplace=True) # document 열에서 중복인 내용이 있다면 중복 제거
test_data['document'] = test_data['document'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","") # 정규 표현식 수행
test_data['document'] = test_data['document'].str.replace('^ +', "") # 공백은 empty 값으로 변경
test_data['document'].replace('', np.nan, inplace=True) # 공백은 Null 값으로 변경
test_data = test_data.dropna(how='any') # Null 값 제거
```

```
X_train = []
for sentence in train_data['document']:
    temp_X = okt.morphs(sentence, stem=True) # 토큰화
    temp_X = [word for word in temp_X if not word in stopwords] # 불용어 제거
    X_train.append(temp_X)
```

→ 전처리 방식은 뉴스 기사의 전처리 방식과 동일

- Tokenizer 정의

```
vocab_size = total_cnt - rare_cnt + 1
print('단어 집합의 크기 :',vocab_size)
```

```
단어 집합의 크기 : 19196
```

```
tokenizer = Tokenizer(vocab_size)
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
```

→ 단어 집합 크기를 구한 후에 집합 크기에 맞게 Tokenizer를 정의하고, 훈련 데이터에 적합

- 모델 적합

```
max_len = 200
below_threshold_len(max_len, X_train)
```

```
전체 샘플 중 길이가 200 이하인 샘플의 비율: 100.0
```

```
X_train = pad_sequences(X_train, maxlen = max_len)
X_test = pad_sequences(X_test, maxlen = max_len)
```

```
from tensorflow.keras.layers import Embedding, Dense, LSTM
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import load_model
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

```
model = Sequential()
model.add(Embedding(vocab_size, 100))
model.add(LSTM(128))
model.add(Dense(1, activation='sigmoid'))
```

→ max_len=200으로 하여 pad_sequences를 진행한 후에 모델에 적합

→ layer는 Embedding을 진행한 후에 LSTM층을 쌓고, sigmiod 함수를 통해 긍·부정을 도출

[Data Engineering]

작성일자	2021-10-06	작성자	정길종

1. 뉴스 기사 크롤링

언론사 선택

```
######### 언론사 선택 및 confirm ######################
print('설정한 언론사를 선택합니다.₩n')

bx_press = browser.find_element_by_xpath('//div[@role="listbox" and @class="api_group_option_sort _search_option_detail

# 기준 두번 째(언론사 분류순) 클릭하고 오픈하기
press_tablist = bx_press.find_elements_by_xpath('.//div[@role="tablist" and @class="option"]/a')
press_tablist[1].click()
time.sleep(sleep_sec)

# 첫 번째 것(언론사 분류선택)
bx_group = bx_press.find_elements_by_xpath('.//div[@class="api_select_option type_group _category_select_layer"]/div[@c

press_kind_bx = bx_group.find_elements_by_xpath('.//div[@class="group_select _list_root"]')[0]
press_kind_btn_list = press_kind_bx.find_elements_by_xpath('.//ul[@role="tablist" and @class="lst_item _ul"]/li/a')
```

페이지 넘기면서 뉴스 크롤링

```
################## 한 페이지 전체 읽기 ######################
urls=browser.current_url
res=one_page_craw(urls,query,cd,press_nm)
news_df=news_df.append(res,ignore_index=True)
  print(news_df.head(1))
news_df.drop_duplicates(['title'],inplace=True)
news_df.dropna(axis=0,inplace=True)
insertTOdb(news_df,press_nm,cd)
news_df = DataFrame(columns=['st_n','st_cd','news','date','title','url','text'])

############### 다음 페이지 넘기기 ############################
try:
    btn_next=browser.find_element_by_xpath('//a[@role="button" and @class="btn_next"]').get_attribute('href')
except:
    continue
print(btn_next)
while btn_next != None:
    btn_next=browser.find_element_by_xpath('//a[@role="button" and @class="btn_next"]').get_attribute('href')
  while page_ct >cureent_nm:
    print('cureent_nm-----',cureent_nm)
    browser.find_element_by_xpath('//a[@role="button" and @class="btn_next"]').click()
    nxt_pg = browser.current_url
      nxt_pg= browser.current_url+'&start='+str(cureent_nm*10+1)
    #### 다음 페이지 이동 ####
    browser.get(nxt_pg)
    time.sleep(sleep_sec)
    urls=browser.current_url
    res=one_page_craw(urls,query,cd,press_nm)
    news_df=news_df.append(res,ignore_index=True)
    news_df.drop_duplicates(['title'],inplace=True)
    news_df.dropna(axis=0,inplace=True)
    cureent_nm += 1
    insertTOdb(news_df,press_nm,cd)
    news_df = DataFrame(columns=['st_n','st_cd','news','date','title','url','text'])

#################################################################
```

언론사별 태그 지정

```
###### 전체 html로 수정 ######
# 매일경제, req.encoding = None 설정 필요
 elif 'mk.co' in url:
    try:
        text = soup.find('div', {'class' : 'art_txt'}).text
        text2 = soup.find('li', {'class' : 'lasttime'}).text
        dates = re.sub(r'[^0-9]', '', text2)[:10]
    except:
        try:
            text = soup.find('div', {'class' : 'view_txt'}).text
        except:
            text=cleanhtml(str(soup))
        try:
            text2 = soup.find('li', {'class' : 'lasttime'}).text
            dates = re.sub(r'[^0-9]', '', text2)[:10]
        except:
            text2 = '0'
```

```
    ## 아시아
    elif 'asiae.co' in url:
        try:
            text = soup.find('div',{'class':'article fb-quotable'}).text.replace('₩n','')
#             dates = soup.find('p',{'class':'user_data'}).text.split('₩t')[0].split(' ')[1:3]
#             dates =" ".join(dates).split(':')[0].replace(' ','').replace(',','')
            dates = soup.find('p',{'class':'user_data'}).text
            dates = re.sub(r'[^0-9]', '', dates)
            dates=dates[:10]
        except:
            text = cleanhtml(str(soup))
            dates= 0
```

DB에 결과 저장

```
########################################## DB에 데이터 넣기 ##########################################
def insertTOdb(news_df,press_nm,cd):
    if press_nm == '매일경제':
        press_nm = 'maeil'
    else:
        press_nm = 'asia'
    table_name=press_nm+'_news_craw_'+str(cd)
    news_df.to_sql(name=table_name, con=engine,if_exists='append',index = False,dtype = {
    'st_n':sqlalchemy.types.VARCHAR(10),
    'st_cd':sqlalchemy.types.VARCHAR(10),
    'news': sqlalchemy.types.TEXT(),
    'date':sqlalchemy.types.VARCHAR(20),
    'title' : sqlalchemy.types.TEXT(),
    'url' :sqlalchemy.types.TEXT(),
    'text' : sqlalchemy.types.TEXT()

    })
```

2. YouTube 스크립트 크롤링

스크롤 내리기

```
######## 스크롤 내리기 #################
while True:
    driver.execute_script('window.scrollTo(0,document.documentElement.scrollHeight)')
    time.sleep(2)
    new_scroll_page_height = driver.execute_script('return document.documentElement.scrollHeight')
    print(scroll_pane_height,new_scroll_page_height)
    if scroll_pane_height == new_scroll_page_height:
        break
    scroll_pane_height = new_scroll_page_height
```

태그값 읽기

```
name = soup.select('a#video-title')
time.sleep(1)
video_url = soup.select('a#video-title')
time.sleep(1)
view = soup.select('a#video-title')
time.sleep(1)
dates = soup.find_all('div',{'id':'metadata-line'})
time.sleep(1)
times=soup.find_all('span',{'class':'style-scope ytd-thumbnail-overlay-time-status-renderer'})
```

데이터 전처리(전체 데이터 중 당일 업로드된 영상만 데이터프레임에 넣기)

```
for i in range(len(name)):
    name_list.append(name[i].text.strip())
    view_list.append(view[i].get('aria-label').split()[-1])
    date_split=view[i].get('aria-label').split()
    try:
        date_index=date_split.index('전')
        #월~일 매일밤 11시에 당일 데이터 크롤링
        ## ~~시간, 며칠 전
        ## 당일 자료 아님 넘김
        if '시간' in date_split[date_index-1:date_index][0]:
            date_list.append(date_split[date_index-1:date_index][0])
        else:
            date_list.append(np.nan)
            continue


    except:
        date_list.append(np.nan)
        continue
```

```
######## 스크롤 내린 결과
result= pd.DataFrame(youtubeDic)
result.dropna(axis=0,inplace=True)
result.drop_duplicates(inplace=True)
result.reset_index(inplace=True)
print(ch_nm,"-의 ",key,'--스크롤 끝--')
```

검색 결과를 URL 검색 후 스크립트 크롤링 결과 DB에 저장

```
for i in range(len(result)):
  for i in range(len(youtubeDf)):
    print('유튜브 스크립트 출력')
    print('{}번째'.format(a))
    dic={}
    try:
        video_url = result.loc[i,'url']
        yt = YouTube(video_url)
        yt.publish_date=yt.publish_date + timedelta(days=1)
        caption = yt.captions.get_by_language_code('a.ko')

        aa =str(caption.xml_captions)
        text=str(cleanhtml(aa)).replace('₩n','')
        date=str(yt.publish_date).split(' ')[0].replace('-','')+'00'
        dic ={
            'st_n':keyword,
            'st_cd':st_cd,
             'ch_nm':yt.author,
            'date':date,
            'title':yt.title,
             'text':text,
            'views':yt.views,
            'length':yt.length,
            'description':yt.description,
           'url': result['url'][i]
        }
        vals=val#ch_nm
          print(dic)
          globals()['result_{}'.format(vals)] = youtubeDf_ss.append(pd.DataFrame.from_dict([dic]))
        youtubeDf_ss = youtubeDf_ss.append(pd.DataFrame.from_dict([dic]))
        youtubeDf_ss.reset_index(inplace=True)
        youtubeDf_ss.drop(columns=['index'],inplace=True)
        youtubeDf_ss.drop_duplicates(['text','title'],inplace=True)
        youtubeDf_ss.dropna(axis=0,inplace=True)
    except Exception as e:
        print('---except 발생---')
        print(e)
        continue
    a +=1

    ### DB에 insert
try:
    insertTOdb(youtubeDf_ss,ch_nm,val)
```

3. 웹 페이지 구현

URL.py

```python
from django.urls import path
from . import views

appname = 'practice'

urlpatterns = [

    #페이지
    path('', views.index),
    path('index.html',views.index, name='index'),

    path('charts.html',views.charts),
    path('charts_fin.html',views.charts_fin),
    path('samsung.html', views.samsung),
    path('sk_hynicx.html', views.sk_hynicx),
    path('hyundai.html', views.hyundai),
    path('lg_chem.html', views.lg_chem),
    path('celltrion.html', views.celltrion),

    # 주식, 코스피 그래프
    path('index.html/<int:yearid>/', views.index, name='result'),
    path('samsung.html/<int:yearid>/', views.samsung, name='samsung'),
    path('hyundai.html/<int:yearid>/', views.hyundai, name='hyundai'),
    path('sk_hynicx.html/<int:yearid>/', views.sk_hynicx,name='sk_hynicx'),
    path('lg_chem.html/<int:yearid>/', views.lg_chem,name='lg_chem'),
    path('celltrion.html/<int:yearid>/', views.celltrion,name='celltrion'),

    # 투표
    path(r'^like/$', views.samsung_chk, name='samsung_chk'),

]
```

chart-area-demo.js

차트 표출 데이터 및 라벨 정의

```javascript
function datasearch(data) {
    var values = document.getElementById("datasearch").value;
    var fin =[]
    var dates = []

    myLineChart.data.datasets[0].pointBorderColor = "rgba(255, 0, 66, 1)"
    myLineChart.data.datasets[0].data = fin;
    myLineChart.data.labels = dates;
    myLineChart.update();
}.
```

celltrion.html

DB → view.py → html

```javascript
var news = JSON.parse("{{ datefinacedataJSON|escapejs }}");
var analysisJSON = JSON.parse("{{ analysisJSON|escapejs }}");
var news_script= JSON.parse("{{ news_pop_negJSON|escapejs }}")
var wordcloud = JSON.parse("{{ wordcloudJSON|escapejs }}")
</script>
```