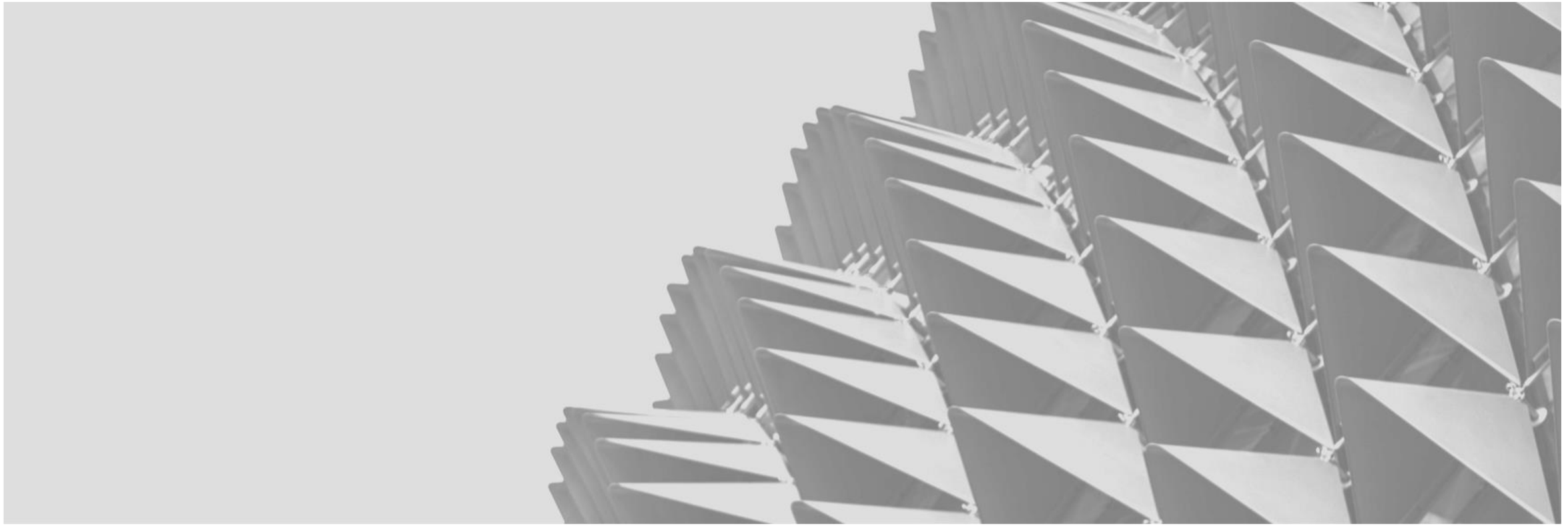
[에이콘아카데미(강남)] 최종 프로젝트



TEAM 3팀 보람삼조

OLLA

▷훈련과정 : 웹 서비스 기반 빅데이터 분석 및 개발자 양성과정
▷훈련기간 : 2022-06-30 ~ 2022-12-27 (960시간/120일)

Index

프로젝트 개요

프로젝트 주제 및 선정 배경

전문 지식을 가지고 투자하는 투자자도 있지만 대부분의 개인 투자자는 뉴스나 미디어를 통해 주가 정보를 얻어 이를 기반으로 투자 방향에 대한 의사 결정을 합니다.
미디어가 투자 방향에 어느정도의 영향을 끼치는지 궁금하여 위 주제를 선정하게 되었습니다.

주가에 영향을 미치는 요인은 다양하지만 모든 요인을 하나의 지표로 산출하기엔 많은 비용과 정보부족의 한계가 있어 사람들이 가장 많이 접하고 쉽게 정보를 얻을 수 있는 뉴스를 대상으로 분석을 진행했습니다.

뉴스에 대해 감정분석을 하고 산출된 결과와 주가 데이터를 이용해 추후 주가 예측을 제공함으로써 투자 전문 기술이 부족한 개인 투자자의 투자 결정에 도움이 되기를 기대합니다.

개발환경

운영체제	window10
프론트 언어	HTML, Javascript(jQuery), CSS
백엔드 언어	Python
빅데이터	numpy, pandas, torch, sklearn, bs4, urllib
	transformers, requests, FinanceDataReader 등
프레임 워크	Django
데이터베이스	Maria DB
협업도구	Git, Slack, Notion
개발도구	Eclipse, Colab

보람삼조

팀원 소개

이승 (팀장)
데이터분석 & 프론트 & 백엔드

주가&뉴스 데이터 추출
메인 화면 UI 및 기능 구현
예측 모델 구현

이소
프론트 & 백엔드

게시판 UI 및 기능 구현
로그인 UI 및 기능 구현
DB설계

문준
데이터분석

뉴스 본문 요약
데이터 병합

임세
프론트 & 백엔드

게시판 UI 및 기능 구현
로그인 UI 및 기능 구현
DB설계

황이
데이터분석 & 백엔드

뉴스 데이터 추출
워드클라우드 구현
예측 모델 구현

이은
데이터분석

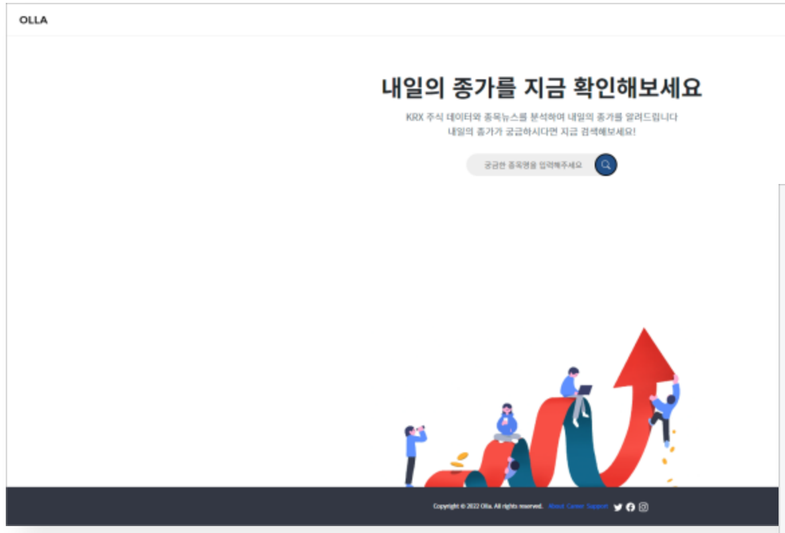
데이터 가공 및 감성 분석&수치화
예측 모델 구현

프로젝트 수행 절차 및 방법

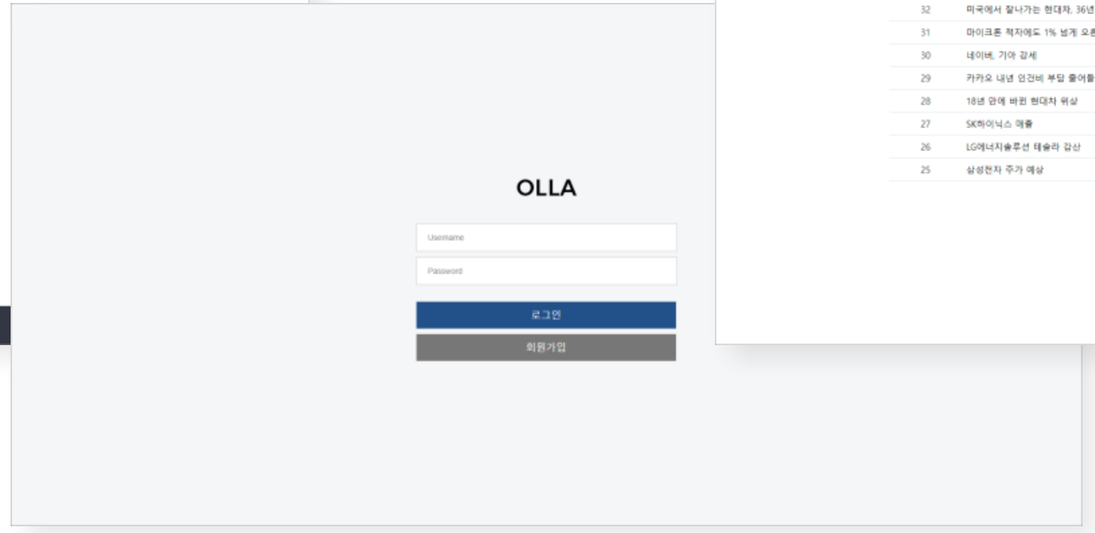
프론트&백엔드

1. UI 구상

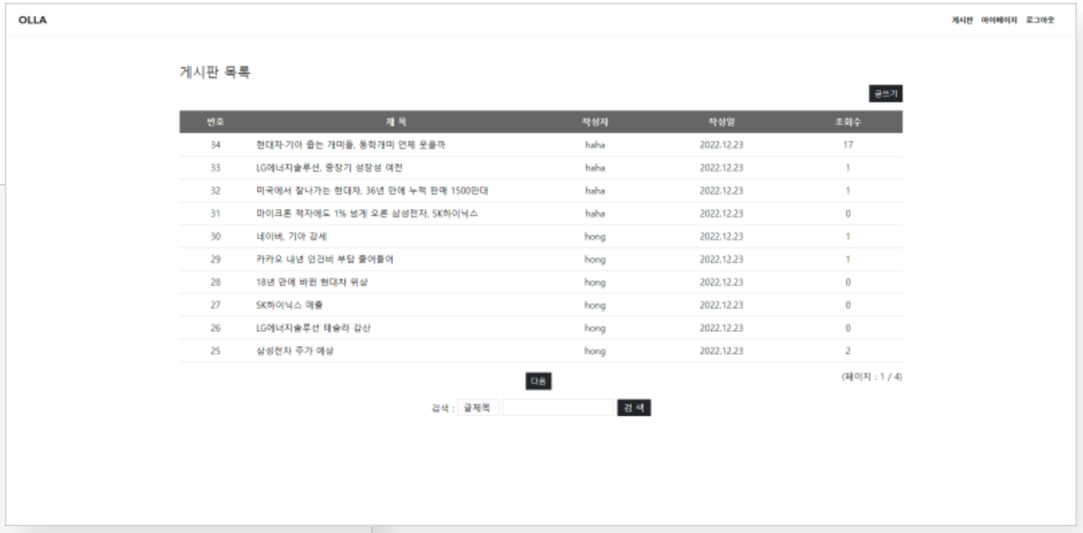
Main Page



Login Page



Board Page



로그인&게시판 DB구조 - MariaDB 활용

member	
id	CharField(primary_key=True, max_length=50)
pwd	CharField(max_length=50)
name	CharField(max_length=50)
phone	CharField(max_length=50)
email	CharField(max_length=50)
reg_date	DateField(auto_now_add=True)

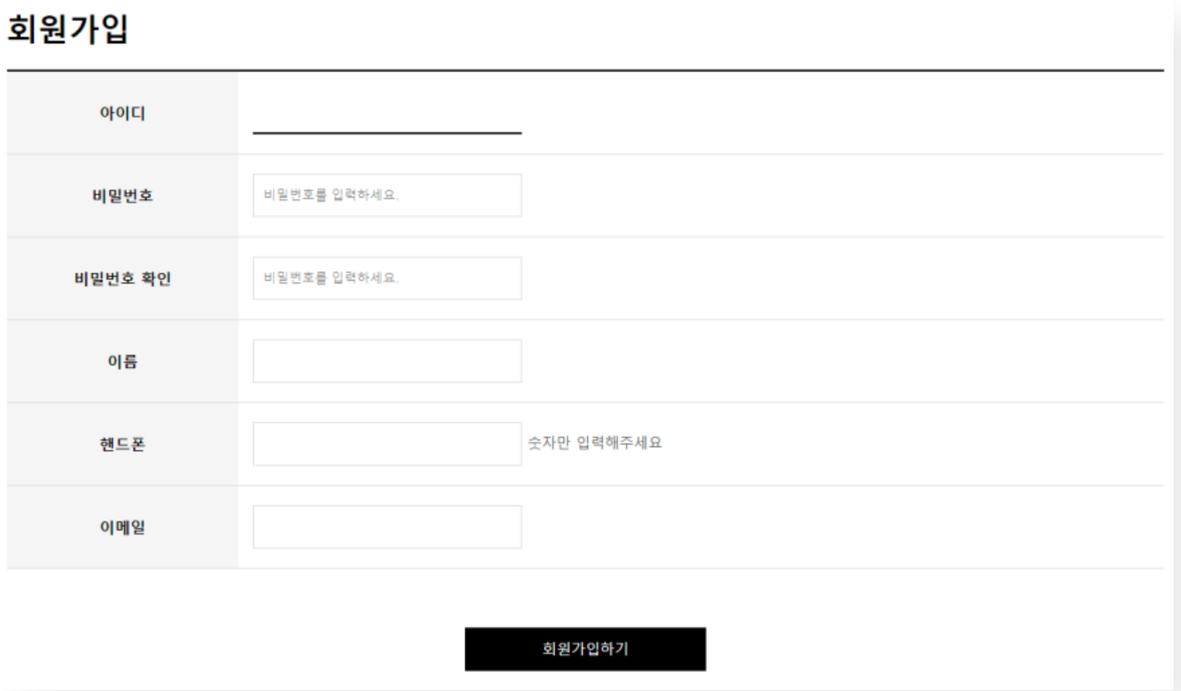
boardtab	
bt_id	CharField(max_length=50)
bt_pwd	CharField(max_length = 50)
title	CharField(max_length = 100)
content	TextField
reg_date	DateField(auto_now_add=True)
readcnt	IntegerField

Comment	
user	ForeignKey('Member', on_delete=models.CASCADE)
post	ForeignKey('BoardTab', on_delete=models.CASCADE)
content	TextField
reg_date	DateField(auto_now_add=True)

dbdiagram.io

회원가입

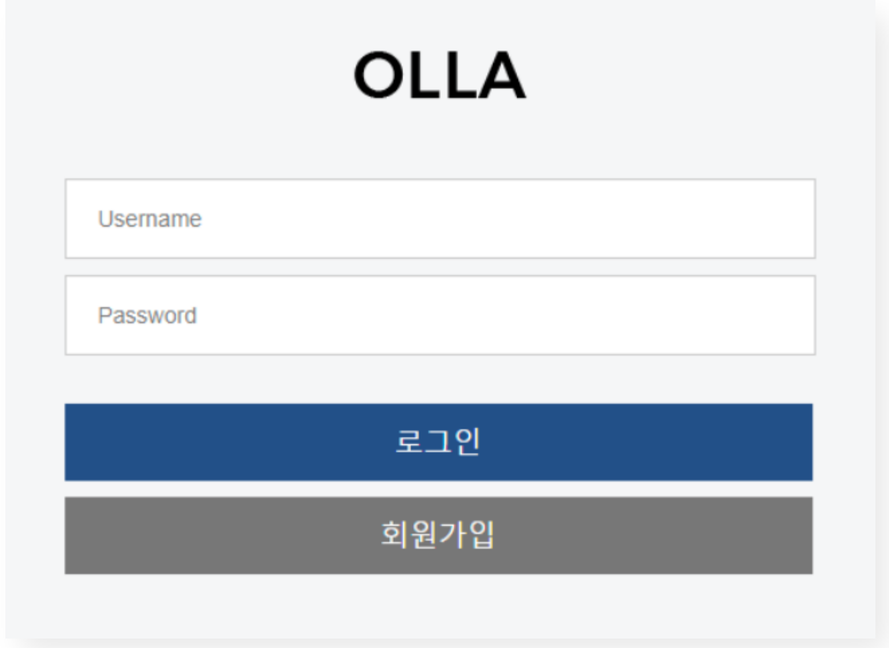
```python
def joinFunc(request):
    if request.method == "GET":
        return render(request, 'user/join.html')
    elif request.method == "POST":
        context = {}
        id = request.POST["id"]
        pwd = request.POST["pwd"]
        name = request.POST["name"]
        phone = request.POST["phone"]
        email = request.POST["email"]

        # 회원가입 중복체크
        rs = Member.objects.filter(id=id)
        if rs.exists():
            context['message'] = id + "가 중복됩니다."
            return render(request, 'user/join.html', context)

        else:
            Member.objects.create(
                id=id, pwd=pwd,  name=name, phone=phone, email=email,
                reg_date=datetime.now())
            context['message'] = name + "님 회원가입 되었습니다."
            return render(request, 'main.html', context)
```



→ 아이디 중복 체크

로그인 및 로그아웃

```python
def loginFunc(request):
    if request.method == "GET":
        return render(request, 'user/login.html')
    elif request.method == "POST":
        context = {}

        id = request.POST.get('id')
        pwd = request.POST.get('pwd')

        # 로그인 체크하기
        rs = Member.objects.filter(id=id, pwd=pwd).first()
        print(id + '/' + pwd)
        print(rs)

        #if rs.exists():
        if rs is not None:

            # OK - 로그인
            request.session['m_id'] = id
            request.session['m_name'] = rs.name


            context['m_id'] = id
            context['m_name'] = rs.name
            context['message'] = rs.name + "님이 로그인하셨습니다."
            return render(request, 'main.html', context)

        else:

            context['message'] = "로그인 정보가 맞지않습니다.\\n\\n확인하신 후 다시 시도해 주십시오."
            return render(request, 'user/login.html', context)
```



로그인 완료 시 아이디, 이름을 세션에 저장
로그아웃 될 때 저장된 세션 삭제

```python
def logoutFunc(request):
    request.session.flush()
    return redirect('/')
```

마이페이지

마이페이지

```python
def profileFunc(request):
    profile_data = Member.objects.get(id=request.session.get('m_id'))
    return render(request, 'user/profile.html', {'profile_data':profile_data})
```

→ 세션 아이디와 동일한 회원 정보 전송

회원정보 수정

```python
def upProfileFunc(request):
    profile_data = Member.objects.get(id=request.session.get('m_id'))
    return render(request, 'user/upProfile.html', {'profile_data':profile_data})

def upProfileOkFunc(request):
    try:
        profile_data = Member.objects.get(id=request.session.get('m_id'))
        upProfile = Member.objects.get(id=request.session.get('m_id'))
        # 비밀번호 비교 후 수정 여부 결정
        print(request.POST.get('pwd'))
        if upProfile.pwd == request.POST.get('pwd'):
            upProfile.name = request.POST.get('name')
            upProfile.phone = request.POST.get('phone')
            upProfile.email = request.POST.get('email')
            upProfile.save()
        else:
            return render(request, 'user/upProfile.html', {'message':'비밀번호가 일치하지 않습니다.', 'profile_data':profile_data})
    except Exception as e:
        return render(request, 'board/error.html')

    return redirect('/user/profile')   # 수정 후 목록 보기
```

회원 탈퇴

```python
def delProfileFunc(request):
    try:
        del_profile = Member.objects.get(id=request.session.get('m_id'))
    except Exception as e:
        return render(request, 'board/error.html')

    return render(request, 'user/delProfile.html', {'del_profile':del_profile})

def delProfileOkFunc(request):
    del_profile = Member.objects.get(id=request.session.get('m_id'))

    if del_profile.pwd == request.POST.get('del_pwd'):
        del_profile.delete();
        return redirect('/user/logout')   # 삭제 후 목록 보기
    else:
        return render(request, 'user/delProfile.html', {'message':'비밀번호가 일치하지 않습니다.', 'del_profile':del_profile})
```

→ 회원 가입 시 입력한 비밀번호 비교 후 수정 여부 및 탈퇴 여부 결정

아이디	haha
이름	하길동
핸드폰	01060000000
이메일	abc@aa.aaa
가입일	2022년 12월 20일

회원정보 수정 | 회원 탈퇴

게시판 페이지

```python
def listFunc(request):
    data_all = BoardTab.objects.all().order_by('-id')

    paginator = Paginator(data_all, 10)
    page = request.GET.get('page')

    try:
        datas = paginator.page(page)
    except PageNotAnInteger:
        datas = paginator.page(1)
    except EmptyPage:
        datas = paginator.page(paginator.num_pages)

    return render(request, 'board/board.html', {'datas':datas})
```

→ 페이징 처리와 게시글 정렬

```python
def searchFunc(request):
    if request.method == 'POST':
        s_type = request.POST.get('s_type')
        s_value = request.POST.get('s_value')
        # print(s_type, s_value)
        # SQL의 like 연산 --> ORM에서는 __contentains=값
        if s_type == 'title':
            datas_search = BoardTab.objects.filter(title__contains=s_value).order_by('-id')
        elif s_type == 'bt_id':
            datas_search = BoardTab.objects.filter(bt_id__contains=s_value).order_by('-id')

        paginator = Paginator(datas_search, 5)
        page = request.GET.get('page')

        try:
            datas = paginator.page(page)
        except PageNotAnInteger:
            datas = paginator.page(1)
        except EmptyPage:
            datas = paginator.page(paginator.num_pages)

        return render(request, 'board/board.html', {'datas':datas})
```

→ 페이징 처리와 검색 기능

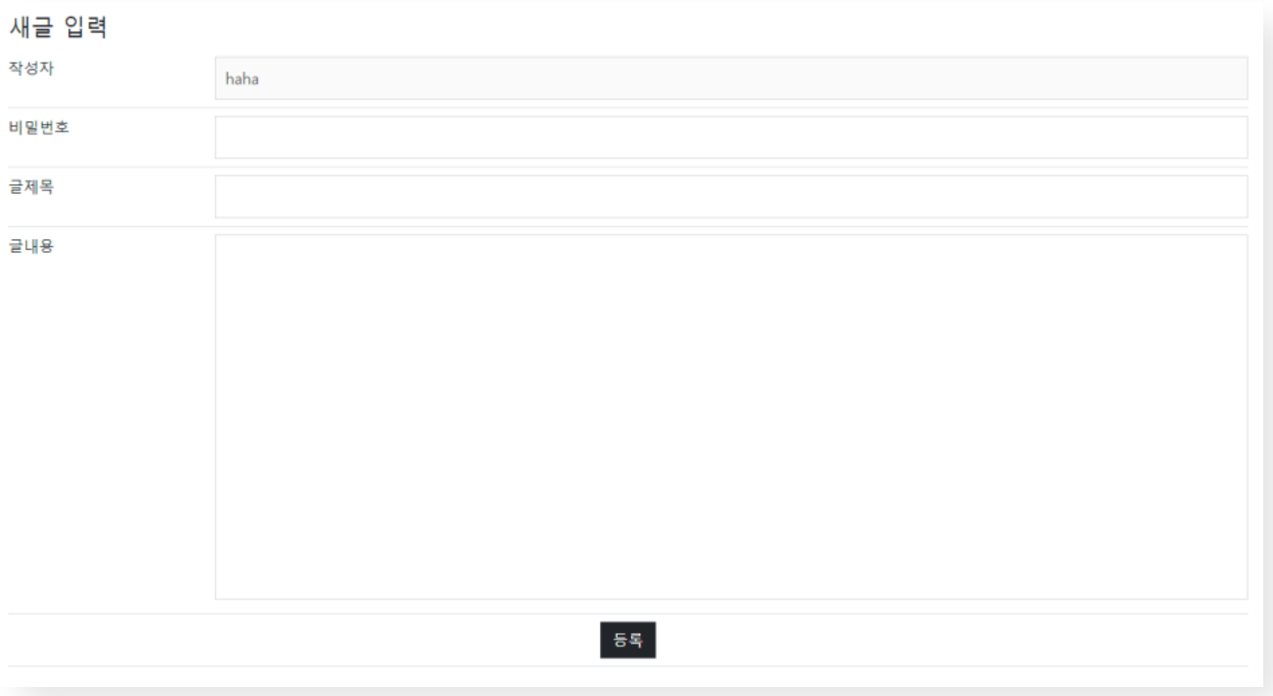
게시글 작성

```python
def insertFunc(request):
    if not 'm_id' in request.session:
        return render(request, 'user/login.html')
    else:
        return render(request, 'board/insert.html')
```

로그인 확인

```python
def insertOkFunc(request):
    if request.method == 'POST':
        try:
            datas = BoardTab.objects.all()
            BoardTab(
                bt_id = request.POST.get('bt_id'),
                bt_pwd = request.POST.get('bt_pwd'),
                title = request.POST.get('title'),
                content = request.POST.get('content'),
                reg_date = datetime.now(),
                readcnt = 0,

            ).save()
        except Exception as e:
            print('insert err : ', e)
            return render(request, 'board/error.html')

    return redirect('/board/list')    # 추가 후 목록 보기
```

게시글 등록



게시글 수정

```python
def updateFunc(request):    # 수정 화면
    try:
        data = BoardTab.objects.get(id=request.GET.get('id'))
    except Exception as e:
        return render(request, 'board/error.html')

    return render(request, 'board/update.html', {'data_one':data})
```

→ 게시글 수정 화면

```python
def updateOkFunc(request):  # 수정 처리
    try:
        upRec = BoardTab.objects.get(id=request.POST.get('id'))

        # 비밀번호 비교 후 수정 여부 결정
        if upRec.bt_pwd == request.POST.get('up_pwd'):
            upRec.bt_id = request.POST.get('bt_id')
            upRec.title = request.POST.get('title')
            upRec.content = request.POST.get('content')
            upRec.save()
        else:
            return render(request, 'board/update.html', {'data_one':upRec, 'msg':'비밀번호가 일치하지 않습니다.'})

    except Exception as e:
        return render(request, 'board/error.html')

    return redirect('/board/list')    # 수정 후 목록 보기
```

→ 게시글 등록 시 작성한 비밀번호 확인 후 수정 여부 결정

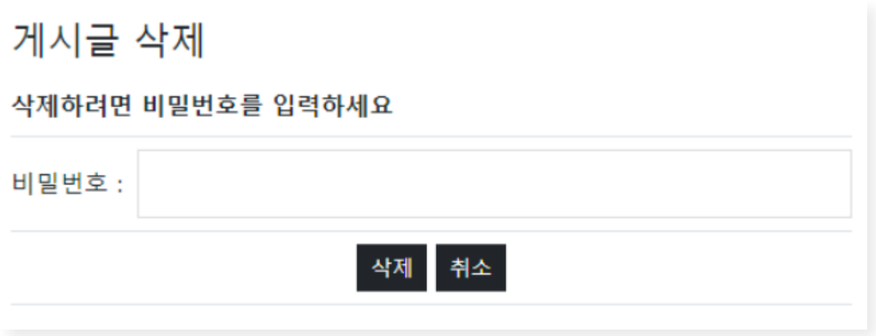
게시글 삭제

```python
def deleteFunc(request):
    try:
        del_data = BoardTab.objects.get(id=request.GET.get('id'))
    except Exception as e:
        return render(request, 'board/error.html')

    return render(request, 'board/delete.html', {'data_one':del_data})
```

→ 삭제할 게시글 정보를 담아서 delete.html로 이동

```python
def deleteOkFunc(request):
    del_data = BoardTab.objects.get(id=request.POST.get('id'))

    if del_data.bt_pwd == request.POST.get('del_pwd'):
        del_data.delete();
        return redirect('/board/list')    # 삭제 후 목록 보기
    else:
        return render(request, 'board/error.html')
```

→ 게시글 비밀번호 확인 및 게시글 삭제



게시글 상세페이지

```
def contentFunc(request):
    page = request.GET.get('page')
    data = BoardTab.objects.get(id=request.GET.get('id'))

    comment = Comment.objects.filter(post_id=request.GET.get('id')).order_by('-id')
    data.readcnt = data.readcnt + 1    # 조회수 증가
    data.save()    # 조회수 update
    return render(request, 'board/content.html', {'data_one':data, 'page':page, 'comment_one':comment})
```

→ 게시글 상세페이지, 조회수 기능

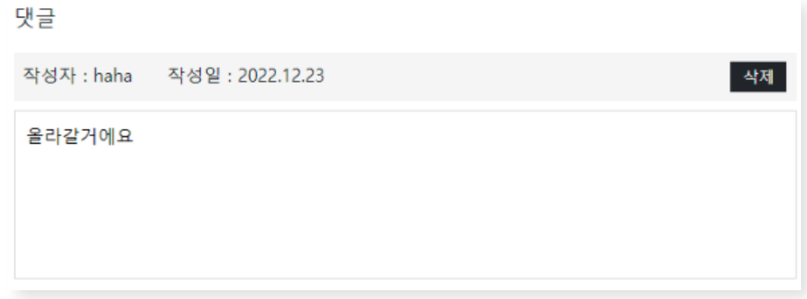
게시글 댓글

```python
def commentInsert(request):
    if not 'm_id' in request.session:
        return render(request, 'user/login.html')
    else:
        if request.method == 'POST':
            try:
                page = request.GET.get('page')
                data = BoardTab.objects.get(id=request.GET.get('id'))
                comment = Comment.objects.filter(post_id=request.GET.get('id')).order_by('-id')
                user = request.POST.get('user')
                id = request.GET.get('id')
                Comment(
                    user = Member.objects.get(id=user),
                    post = BoardTab.objects.get(id=id),
                    content = request.POST.get('content'),
                    reg_date = datetime.now(),
                ).save()
            except Exception as e:
                print('insert err : ', e)
                return render(request, 'board/error.html')

        return render(request, 'board/content.html', {'data_one':data, 'page':page,'comment_one':comment})
```

로그인 확인, 댓글 작성(게시글 및 회원가입정보와 연동)


```python
def commentDelete(request):
    try:
        page = request.GET.get('page')
        data = BoardTab.objects.get(id=request.GET.get('id'))
        comment = Comment.objects.filter(post_id=request.GET.get('id')).order_by('-id')
        del_comment = Comment.objects.get(id=request.GET.get('comment_id'))
        del_comment.delete();

        return render(request, 'board/content.html', {'data_one':data, 'page':page,'comment_one':comment})
    except Exception as e:
        return render(request, 'board/error.html')
```

특정 댓글 삭제



KOSPI 상위 50개를 대상으로 1년치 뉴스 개수를 크롤링하여 Top6 선정

종목 : 삼성전자, 카카오, 현대차, SK하이닉스, 기아, LG에너지솔루션

```
'''
kospi 50 종목별 기사갯수 일년치 크롤링 해오기
'''
from bs4 import BeautifulSoup
import requests
import pandas as pd
import csv
import re
from urllib import parse
from selenium import webdriver
from selenium.webdriver.common.by import By
import time
import selenium
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

#읽을 웹 주소
baseUrl="https://finance.naver.com/sise/sise_market_sum.naver"
sourceData=requests.get(baseUrl)

#페이지 소스가 넘어온다.(텍스트로 넘어옴)
plainText = sourceData.text

#BeautifulSoup 객체로 만들기
soup= BeautifulSoup(plainText,'lxml')

#종목명을 담을 list
stockItems = []
#a태그만 가져오기
for atag in soup.find_all('a',{'class':'tltle'}) :
    stockItems.append(atag.string) #a태그 텍스트를 리스트에 담기
```

```
# selenium 사용했을 때 코드
url = "https://finance.naver.com/news/news_search.naver?rcdate=&q=&x=0&y=0&sm=all.basic&pd=4&stDateStart=2021-12-09&stDateEnd=2022-12-09"
driver = webdriver.Chrome('C:/work/chromedriver.exe')
driver.implicitly_wait(1)
driver.get(url)

NewsCount=[]
for idx, i in enumerate(stockItems):
    search = driver.find_element(By.XPATH, '//*[@id="contentarea_left"]/form/div/div/div/input[1]') #검색창path
    button = driver.find_element("xpath", '//*//*[@id="contentarea_left"]/form/div/div/div/input[2]') #검색버튼 path

    search.clear() #검색창 빈칸으로
    search.send_keys(i) #종목명을 검색창에 입력
    driver.implicitly_wait(1)
    button.click() #검색 버튼 클릭
    driver.implicitly_wait(1)
    res= requests.get(driver.current_url) #해당 종목이 검색된 웹 주소 받기
    soup = BeautifulSoup(res.text, 'html.parser')
    body = soup.select('p.resultCount > strong') #종목명과 총 기사갯수가 적힌 문장

    stock = body[0].text #종목명
    count = body[1].text.replace(",","") # 총 기사갯수
    print(idx+1, stock, count)
    NewsCount.append([stock, count])
    driver.back()
driver.close()

#데이터 프레임 생성
df = pd.DataFrame(NewsCount)
df.columns=['종목명','count'] #칼럼명 추가

#count의 Dtype을 object에서 int32로 변경
df = df.astype({'count':'int'})
#print(df.info())

#count를 기준으로 내림차순 및 출력
print(df.sort_values('count',ascending=False))
```

	종목명	count
18	SK	85715
26	LG	79116
0	삼성전자	76224
10	카카오	63049
7	현대차	37441
2	SK하이닉스	25678
9	기아	24660
1	LG에너지솔루션	24609
20	LG전자	21939

종목별 네이버 종목뉴스에서 1년치(2021.12.09 ~ 2022.12.09) 뉴스 크롤링

```
if __name__ == "__main__":
    #각 URL별로 financialNews함수 실행
    #검색 기간 : 2021-12-09 ~ 2022-12-09 을 한달씩 검색
    start_date = date(2021, 12, 9)
    end_date = date(2022, 1, 9)

    while end_date <= date(2022, 12, 9):
        # 크롤링 함수 호출
        for url in urls:
            print(start_date.strftime("%Y-%m-%d"), end_date.strftime("%Y-%m-%d"))
            financialNews(url +"&stDateStart="+start_date.strftime("%Y-%m-%d")+"&stDateEnd="+end_date.strftime("%Y-%m-%d")+"&page=")

        # dateutil의 relativedelta함수를 사용, 시작날짜와 끝 날짜에 한달씩 추가해서 다음달 검색
        end_date = end_date + relativedelta(months=1)
        if end_date == date(2022, 2, 9): #두번째 달부터는 검색 시작 날짜가 10일부터 시작되도록
            start_date = start_date + relativedelta(months=1, days=1)
        else:
            start_date = start_date + relativedelta(months=1)
```

date 함수를 사용하여
시작 날짜와 종료 날짜를 한달간격으로 지정

dateutil의 relativedelta 함수를 사용해서
두번째 달 부터는 중복 방지를 위해 검색 시작 날짜가 9일이 아닌 10일부터 시작되도록
month와 day에 1을 더하고 종료 날짜에도 month를 추가하여 한달씩 차례대로 검색되도록 함

종목별 네이버 종목뉴스에서 1년치(2021.12.09 ~ 2022.12.09) 뉴스 크롤링

```
def financialNews(url):
    #컬럼스 : 종목명, 뉴스 제목 , 언론사, 작성날짜, 시간, 본문 내용
    stocklist=[]
    titlelist=[]
    companylist=[]
    datelist=[]
    # timelist=[]
    contentlist=[]

    #header에 User Agent 정보(네이버의 경우 네이버 검색로봇 Yeti)를 기입하여 웹 서버로 페이지를 요청할 시에 같이 보내서 크롤링 제한을 피한다
    headers = {'User-Agent': 'Mozilla/5.0 (compatible; Yeti/1.1; +http://naver.me/spd)'}

    #i는 페이지 번호 | 1~200 반복
    for i in range(1,201) :
        #본문으로 들어가는 링크를 담을 리스트
        links=[]
        res = requests.get(url+str(i), headers = headers)
        soup = BeautifulSoup(res.text, 'html.parser')
        body = soup.find(class_='newsSchResult')
        title = body.find_all(['dt','dd'], {'class': 'articleSubject'})  #뉴스제목

        #다음 페이지로 넘어갔을 때, 읽어올 title이 없을 경우 for문 종료
        if title == [] : break

        summary = body.find_all('dd', {'class': 'articleSummary'}) #요약문,언론사,날짜,시간
        stock = soup.select('p.resultCount > strong')[0].text  #종목명
```

BeautifulSoup 라이브러리를 사용
페이지별로 find or find_all 함수를 활용하여
뉴스제목,요약문,언론사,날짜, 종목명에 걸려있는
하이퍼링크 추출

데이터분석
1. 데이터 추출 및 가공


```
    #한 페이지의 20개의 언론사 , 작성날짜, 시간 가져오기
    for item in summary:
      # information = item.string # 요약본, 언론사, 날짜, 시간
      press =  re.sub(r"\s", "", item.find('span', {'class': 'press'}).text) #언론사 -> 공백 제거완료
      wdate_all =  re.sub(r"\s", "", item.find('span', {'class': 'wdate'}).text) #날짜,시간 -> 공백 제거완료
      wdate=wdate_all[:10] #날짜만
      # time=wdate_all[10:] #시간만

      #언론사 리스트에 저장
      companylist.append(press)
      #작성날짜 리스트에 저장
      datelist.append(wdate)
```

```
    #본문 추출
    for link in links:
      print('link : ',link)
      res = requests.get(link, headers = headers)
      soup = BeautifulSoup(res.text, 'html.parser')
      try:
          # link_news 내용이 있을 경우
          if soup.find('div',{'class': 'link_news'}) != None:
              # link_news 제거
              soup.find('div',{'class': 'link_news'}).decompose()
          #본문 내용 추출
          content = soup.find('div',{'id':'content'}).text.strip()
      except AttributeError as e:
          #본문이 없을 경우, NaN로 채우기
          print('error : ',e)

      print(content)
      print("-------다음뉴스------")
      #본문내용을 리스트에 저장
      contentlist.append(content)
```

→ 본문이 담긴 div내에 불필요한 데이터(link_news)가 존재할 경우 decompose 함수를 사용하여 제거 후 본문 내용 추출

목록보기

머니투데이 주요뉴스 | 해당 언론사에서 선정하며 **언론사 페이지(아웃링크)**로 이동해 볼 수 있습니다.

- '재벌2세와 이혼' 최정윤 "딸, 낯선 男에 '아빠'…
- 여배우 반효정, 면전서 허영만 저격? "짜증나"
- 30억 별장·130억 빌라…아이유 클래스가 다르네
- 목디스크 터진 소유 "속옷도 안입고 병원행"
- '사진 한 장=43억' 제니, 한남동 빌라·명품침…

예) link_news

```
# 데이터셋 만들기
NaverFinance = pd.DataFrame({
    'stock' : stocklist,
    'title' : titlelist,
    'company' : companylist,
    'date' : datelist,
    # 'time' : timelist,
    'content' : contentlist
})
```

```
# csv 파일로 저장
    # 최초 생성 이후 mode는 append로 변경
    if not os.path.exists('news_dataset.csv'):
        NaverFinance.to_csv('news_dataset.csv', mode='w', encoding='utf-8-sig')
    else:
        NaverFinance.to_csv('news_dataset.csv', mode='a', encoding='utf-8-sig', header=False)
```

→ csv 파일로 저장

뉴스 데이터 추출 결과

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		stock	title	company	date	content									
2	0	삼성전자	삼성전자, 인도 '1위' 통신사에 5G 장비 공급한다	이데일리	2022-12-09	인도 1·2위 사업자에 1조 규모 장비 공급5G 개화한 세계 2위 이동통신 시장 '정조준'[이데일리 이다원 기지									
3	1	삼성전자	삼성전자, 인도1위 통신사와 5G 네트워크 장비 공급 계약	매일경제	2022-12-09	삼성전자가 인도 가입자 1위 통신사에 5세대(5G) 네트워크 장비를 공급한다. 9일 전자업계에 따르면 삼성									
4	2	삼성전자	한종희 삼성전자 부회장, 내년에도 TV,가전 지휘..15일 위기 타개 전략 짠다	서울신문	2022-12-09	VS사업부 신설 부사업부장에 용석우 부사장삼성전자가 9일 조직 개편, 보직 인사를 마무리 지은 가운데 D									
5	3	삼성전자	'겸직 유지' 한종희 삼성전자 부회장, 내년에도 가전·TV 이끈다	머니투데이	2022-12-09	한종희 삼성전자 부회장/사진제공=삼성전자한종희 삼성전자 DX(디바이스 경험)부문장(부회장)이 영상디									
6	4	삼성전자	한종희 삼성전자 부회장, 생활가전·VD사업부장 겸직 유지	SBSBiz	2022-12-09	삼성전자 DX(디바이스경험)부문장 한종희 부회장이 내년에도 영상디스플레이(VD), 생활가전(DA)사업부장									
7	5	삼성전자	인사 끝낸 삼성전자 15일 글로벌전략회의...복합위기 대응책 모색(종합2보)	연합뉴스	2022-12-09	조직 개편·보직 인사 마무리...한종희 부회장, 종전처럼 생활가전사업부장 겸임다른 그룹도 내년 경영 환경									
8	6	삼성전자	삼성전자 조직개편...한종희 부회장, TV·가전 계속 챙긴다	이데일리	2022-12-09	글로벌 시황 악화에 안정적 리더십 택해비상경영체제 돌입한 삼성, 전략회의 연다[이데일리 이다원 기자]									
9	7	삼성전자	삼성전자, 인도 1위 통신사업자에 5G 장비 공급계약	연합뉴스	2022-12-09	세계 2위 통신시장 인도 1·2위 사업자에 1조원 5G 장비 공급(서울=연합뉴스) 조성미 기자 = 삼성전자가 인									
10	8	삼성전자	[인사] 삼성전자	SBSBiz	2022-12-09	삼성전자 DS부문▲ 한진만(1966년생) 미주총괄(DSA) 부사장▲ 김재준(1968년생) 메모리 전략마케팅실장									
11	9	삼성전자	삼성전자, 3Q 파운드리 점유율 15.5%...1위 TSMC는 56.1%	데일리안	2022-12-09	TSMC, '아이폰 효과' 수혜...삼성전자, 원화 약세 등으로 매출 감소ⓒ트렌드포스[데일리안 = 조인영 기자] 글									
12	10	삼성전자	경영안정에 힘 실은 삼성전자...한종희, VD·생활가전사업부장 겸직 유지	아이뉴스24	2022-12-09	삼성전자, 조직개편·보직인사 마무리...생활가전사업부장 후임 인선 없어삼성전자가 사장단 인사를 시작으									
13	11	삼성전자	삼성전자 한종희, 내년에도 TV·가전 모두 챙긴다	한국경제	2022-12-09	삼성전자 조직개편 확정디스플레이·가전 사업부장 겸직"실적부진에 강력한 리더십 필요"유럽·서남아 지역									
14	12	삼성전자	경계현 삼성전자 사장, 팻 겔싱어 인텔 CEO 만나	헤럴드경제	2022-12-09	삼성전자 화성캠퍼스에서 경계현 사장과 만나경계현(왼쪽) 삼성전자 DS부문 사장과 팻 겔싱어 인텔 최고경									
15	13	삼성전자	삼성전자, 인텔 CEO와 반도체 협력 논의...15일 전략회의선 '경영 새판 짜기'	서울경제	2022-12-09	■내년 전략 마련 속도전경계현 등 경영진, 겔싱어와 회동이재용 붙참…M&A 거론 안한 듯내주 한종희 등									
16	14	삼성전자	[단독] 삼성전자, 전략마케팅실장·영업팀장 등 예비 CEO '물갈이'	SBSBiz	2022-12-09	美·中 법인장 등 '수백명' 대규모 인력 재배치삼성전자가 반도체사업부 예비 CEO에 대한 고강도 쇄신 인사									
17	15	삼성전자	삼성전자, 베트남 국가주석에게 부산 엑스포 지지 당부	한경비즈니스	2022-12-09	[비즈니스 플라자]삼성전자 DX부문장 한종희 부회장(사진 왼쪽)이 6일 서울시 중구 롯데호텔에서 응우옌									
18	16	삼성전자	겔싱어 인텔 CEO, 경계현 삼성전자 사장과 회동...반도체 협력 논의	아이뉴스24	2022-12-09	시스템·메모리반도체 사업 논의...귀국한 이재용 회장은 붙참한 듯방한한 팻 겔싱어 인텔 최고경영자(CEO)									
19	17	삼성전자	[단독] 삼성전자 조직개편...DS미래건설혁신·디지털트윈TF 신설	뉴스1	2022-12-09	한종희 부회장, 생활가전·VD사업 부장 겸직 유지미래 먹거리 위한 미래건설혁신·디지털트윈TF 출범서울 삼									
20	18	삼성전자	삼성전자 파운드리 점유율 15%대로...TSMC와 격차 확대	이데일리	2022-12-09	트렌드포스, 3분기 글로벌 10대 파운드리 매출·점유율 분석TSMC 매출 2분기 대비 11% 상승, 점유율도 56									
21	19	삼성전자	삼성전자, 15일부터 글로벌전략회의...글로벌 위기 돌파 머리 맞댄다	중앙일보	2022-12-09	서울 서초구 삼성전자 서초사옥. 연합뉴스 삼성전자가 오는 15일부터 글로벌전략회의를 열어 내년									
22	20	삼성전자	인텔 CEO, 경계현 삼성전자 사장과 회동...반도체 협력 논의	노컷뉴스	2022-12-09	핵심요약중동 출장을 마치고 9일 오전 귀국한 이재용 회장이 겔싱어 CEO를 만났는지 여부는 확인되지 않									
23	21	삼성전자	'친구이자 적' 인텔 CEO, 삼성전자 사장 만나...이재용 회장 만날까	매일경제	2022-12-09	팻 겔싱어 CEO, 삼성 화성캠퍼스 들러경계현 사장 만나 반도체 협력 논의뒤이어 이재용·겔싱어 만남여부									
24	22	삼성전자	삼성전자 줄고, TSMC 늘고...파운드리 격차 더 커졌다	머니투데이	2022-12-09	삼성전자와 TSMC간의 글로벌 파운드리(시스템반도체 위탁생산)시장 점유율 격차가 올해 3분기 더 벌어졌									
25	23	삼성전자	삼성전자, 2024년부터 해외법인과 거래시 국제표준가격 준수해야	헤럴드경제	2022-12-09	다국적기업 간 이전가격 책정 '어마운트B' 논의[연합][헤럴드경제=배문숙 기자] 삼성전자를 비롯한 다국적									

예) '삼성전자' 뉴스 데이터

FinanceDataReader 를 사용하여 종목별 1년치(2021.12.09 ~ 2022.12.09) 주가 추출

```
#colab에서 사용 시 실행 필요
!pip install -U finance-datareader

import FinanceDataReader as fdr
# 버전 확인
fdr.__version__

# KRX는 KOSPI,KOSDAQ,KONEX 전체를 가져온다
# KOSPI 종목 데이터 전체 얻기
df_KOSPI = fdr.StockListing('KOSPI')

stock_names=['삼성전자','카카오','현대차','SK하이닉스','기아','LG에너지솔루션']
stock_codes=[] #종목코드를 담을 list
#종목코드 얻기
for name in stock_names:
  stock_codes.append(df_KOSPI.loc[df_KOSPI['Name'] == name]['Code'].values[0])
# print(stock_codes)
# => ['005930', '035720', '005380', '000660', '000270', '373220']
```

```
'''
칼럼 정보
'Open'   : '시가'
'High'   : '고가'
'Low'    : '저가'
'Close'  : '종가'
'Volume' : '거래량'
'Change' : '변화율'
'''

import pandas as pd
# 6개 종목별 '2021-12-09'~'2022-12-09' 데이터 가져오기 및 csv 생성
for name,code in zip(stock_names,stock_codes):
  data = fdr.DataReader(code, '2021-12-09', '2022-12-09')

  #종목별 csv 파일로 저장
  data.to_csv(name+"주가.csv", mode='w')

  #하나의 csv 파일로 저장
  data.to_csv("주가통합.csv", mode='a', header=False)
```

주가 보조데이터 : FinanceDataReader 를 사용하여 독립변수 7개 추가

```
#보조 데이터 기간 설정
start_date = '2021-12-09'
end_date = '2022-12-09'

#stock_data의 Date 칼럼 type 변경
stock_data['date'] = stock_data['date'].astype('datetime64')
# print(stock_data.info())

#FinanceDataReader를 이용해 보조 데이터 가져오기
df_dict = {
    'KS11' : fdr.DataReader('KS11', start_date, end_date), #코스피지수
    'KQ11' : fdr.DataReader('KQ11',start_date, end_date), #코스닥지수
    'DAU' : fdr.DataReader('DJI', start_date, end_date), #다우지수
    'NASDAQ' : fdr.DataReader('IXIC', start_date, end_date), #나스닥지수
    'SP500':  fdr.DataReader('US500', start_date, end_date), #S&P 500
    'USD' : fdr.DataReader('USD/KRW', start_date, end_date), #달러당 원화 환율
    'JPY' : fdr.DataReader('JPY/KRW', start_date, end_date) #엔화 원화 환율
}

#각 보조데이터의 수정주가를 추출해서 stock_data에 추가하기
for key in df_dict:
  #df_dict[key] 인덱스 해제
  extra_df = df_dict[key].rename_axis('Date').reset_index()

  #사용할 칼럼명 생성
  newName= key+'_Adj_Close'

  #Date와 Adj Close컬럼만 추출
  extra_df_cut=extra_df.loc[:,['Date','Adj Close']]
  #Adj Close컬럼명을 newName으로 변경
  extra_df_cut.rename(columns={'Adj Close':newName},inplace=True)

  #stock_data의 Date를 기준으로 extra_df_cut과 merge
  merge_outer = pd.merge(stock_data,extra_df_cut, how='left',left_on='date', right_on='Date')

  #stock_data에 merge_outer의 newName(key+'_Adj_Close')칼럼 데이터 추가
  stock_data[newName] = merge_outer[newName]
print(stock_data)
```

변수 5개로는 데이터의 정보가 부족하다고 판단하여 보조지표 추가 및 주가 데이터와 merge
(코스피지수, 코스닥지수, 다우지수, 나스닥지수, S&P 500, 달러당 원화 환율, 엔화 원화 환율)

	date	Open	High	Low	Close	Volume	Change	sent_index	KS11_Adj_Close	KQ11_Adj_Close	DAU_Adj_Close	NASDAQ_Adj_Close	SP500_Adj_Close	USD_Adj_Close	JPY_Adj_Close
2021-12-09	82400	84800	82400	84100	1774961	0.010817	0.002200	3029.570068	1022.869995	35754.691406	15517.370117	4667.450195	1172.560059	10.314114
2021-12-10	83800	85900	83300	85400	1690388	0.015458	-0.002666	3010.229980	1011.570007	35970.988281	15630.599609	4712.020020	1177.329956	10.378255
2021-12-13	86500	88200	85900	86000	2232198	0.007026	0.004862	3001.659912	1005.960022	35650.949219	15413.280273	4668.970215	1178.969971	10.386440
2021-12-14	85200	86700	84700	85300	1006657	-0.008140	-0.001426	2987.949951	1002.809998	35544.179688	15237.639648	4634.089844	1185.099976	10.434101
2021-12-15	84500	85300	84200	84300	651974	-0.011723	0.017720	2989.389893	1003.520020	35927.429688	15565.580078	4709.850098	1184.229980	10.411362

데이터 추출 결과

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	date	Open	High	Low	Close	Volume	Change	KS11_Adj_Close	KQ11_Adj_Close	DAU_Adj_Close	NASDAQ_Adj_Close	SP500_Adj_Close	USD_Adj_Close	JPY_Adj_Close
0	2021-12-09	82400	84800	82400	84100	1774961	0.010817308	3029.570068	1022.869995	35754.69141	15517.37012	4667.450195	1172.560059	10.314114
1	2021-12-10	83800	85900	83300	85400	1690388	0.015457788	3010.22998	1011.570007	35970.98828	15630.59961	4712.02002	1177.329956	10.378255
2	2021-12-13	86500	88200	85900	86000	2232198	0.007025761	3001.659912	1005.960022	35650.94922	15413.28027	4668.970215	1178.969971	10.38644
3	2021-12-14	85200	86700	84700	85300	1006657	-0.008139535	2987.949951	1002.809998	35544.17969	15237.63965	4634.089844	1185.099976	10.434101
4	2021-12-15	84500	85300	84200	84300	651974	-0.011723329	2989.389893	1003.52002	35927.42969	15565.58008	4709.850098	1184.22998	10.411362
5	2021-12-16	85400	86000	84400	85700	1014992	0.016607355	3006.409912	1007.859985	35897.64063	15180.42969	4668.669922	1184.400024	10.376639
6	2021-12-17	84700	85800	84500	84600	1300076	-0.012835473	3017.72998	1001.26001	35365.44141	15169.67969	4620.640137	1186.150024	10.43549
7	2021-12-20	84500	84500	82800	82900	924100	-0.020094563	2963	990.51001	34932.16016	14980.94043	4568.02002	1186.390015	10.445495
8	2021-12-21	83500	83600	82400	83200	977687	0.003618818	2975.030029	996.599976	35492.69922	15341.08984	4649.22998	1190.119995	10.47051
9	2021-12-22	83700	84800	83400	83900	1002461	0.008413462	2984.47998	1000.130005	35753.89063	15521.88965	4696.560059	1191.27002	10.450149
10	2021-12-23	84500	84700	83700	84200	834876	0.003575685	2998.169922	1003.309998	35950.55859	15653.37012	4725.790039	1187.77002	10.407714
11	2021-12-24	84700	85800	84500	85100	1210818	0.010688836	3012.429932	1007.419983	35950.55859	15653.37012	4725.790039	1185.5	10.357149
12	2021-12-27	85400	85800	84800	84800	627895	-0.003525264	2999.550049	1011.359985	36302.37891	15871.25977	4791.189941	1185.920044	10.363698
13	2021-12-28	85000	85200	83800	84500	1209343	-0.003537736	3020.23999	1027.439941	36398.21094	15781.71973	4786.350098	1185.650024	10.334326
14	2021-12-29	83700	84100	83300	83400	1061718	-0.013017751	2993.290039	1028.050049	36488.62891	15766.21973	4793.060059	1187.849976	10.349199
15	2021-12-30	83300	83900	82100	82200	1252096	-0.014388489	2977.649902	1033.97998	36398.07813	15741.55957	4778.72998	1183.670044	10.297173
16	2022-01-03	82900	83200	82300	82600	778020	0.00486618	2977.649902	1033.97998	36585.05859	15832.79981	4796.560059	1187.780029	10.307657
17	2022-01-04	83000	84000	82700	83500	959153	0.010895884	2989.23999	1031.660034	36799.64844	15622.71973	4793.540039	1194.680054	10.358659
18	2022-01-05	85100	87100	84800	85900	3678719	0.028742515	2953.969971	1009.619995	36407.10938	15100.16992	4700.580078	1196.5	10.298985
19	2022-01-06	85000	86600	84700	85600	1767632	-0.003492433	2920.530029	980.299988	36236.46875	15080.86035	4696.049805	1199.25	10.326608
20	2022-01-07	85800	87200	85700	86700	1905888	0.012850467	2954.889893	995.159973	36231.66016	14935.90039	4677.029785	1205.780029	10.397956
21	2022-01-10	86100	86500	83500	83800	1818497	-0.033448674	2926.719971	980.380005	36068.87109	14942.83008	4670.290039	1196.880005	10.340882

뉴스 데이터 전처리

본문 내용에 기사 없거나 신문사의 내용만 담겨져 있는 본문 제거

제목과 내용의 중복이 있다면 첫번째 중복된 내용만 남기고 제거

NaN 값이 존재하는지 체크

```python
import pandas as pd

test_df = pd.read_csv("/content/drive/MyDrive/news/삼성_dataset.csv") # 삼성 뉴스.csv 불러오기

print(test_df.shape)  # 전처리 전에 데이터 수 확인
test_df = test_df.loc[test_df['content'].str.len() >= 15] # 본문 내용의 길이가 15이상인 데이터만 추
# test_df['content'].str.strip()
# test_df['content'].nunique()

# 제목의 중복이 있을경우 첫번째 데이터를 제외하고 삭제
test_df = test_df.drop_duplicates(['title'], keep='first')

# 본문의  중복이 있을경우 첫번째 데이터를 제외하고 삭제
test_df = test_df.drop_duplicates(['content'], keep='first')

# 널값 체크
print(test_df.isnull().sum())

# 전 처리 후 데이터 저장
test_df.to_csv('/content/drive/MyDrive/Sam_dataset.csv', encoding='utf-8-sig', index = None)
test_df.shape # 전처리 후에 데이터 수 확인
```

가공된 각 기업별 1년치 뉴스데이터 사용
제목과 본문을 연결 (News , Content로 사용)

```
# 뉴스 제목과 본문 합친 후 list로 변환
total = np.array(test_df['title'].str.cat(test_df['content']))
print(total[0])
```

Pororo의 abstractive model을 활용하여 뉴스 요약

pororo는 텍스트 요약을 위해 3개의 모델을 제공합니다. 'abstractive' 모델은 하나의 완전한 문장으로 텍스트 내용을 요약합니다. 이 모델은 SKT에서 개발한 KoBART모델을 사용합니다. 학습에 사용한 데이터는 데이콘(DACON)의 문서 추출요약 경진 대회 데이터 AI 허브에서 공개한 AI 학습용 데이터입니다.

```
# 본문 요약 (Pororo)
text_sum = Pororo(task="text_summarization", lang="ko", model="abstractive")

# 요약 내용을 담을 리스트
summary_list = []

# 본문 요약 Pororo를 사용해 요약
for i in total:
  print('원본 : ',i)
  print()
  print('요약 : ',text_sum(i))
  #리스트에 저장
  summary_list.append(text_sum(i))
```

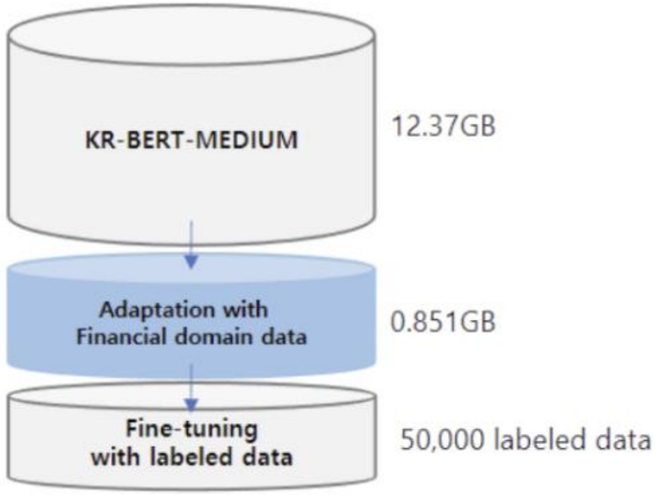
News Content

하반기 디지털 손해보험사 출범 목표보험업계 게임체인저 될까.
카카오뱅크와 카카오페이가 금융 소비자들의 손에 익어가자 카카오가 다음 금융 플랫폼 타깃으로 보험을 겨냥했다. 하반기 빅테크와 보험이 결합한 디지털 손해보험사가 약 4500만명에 달하는 카카오톡이 카카오의 보험영업을 뒷받침해줄 전망이다. 또 코로나19 사태로 증폭된 시대적 변화를 등에 업고 정부가 추진하고 있는 마이데이터 사업이 잘개를 달아주며 전통 보험사들을 긴장하게 할 것으로 보인다. 이미 업계에서는 카카오톡을 무기로 카카오뱅크를 은행권 메기로 성공시킨 카카오의 보험사 설립을 주시하는 분위기다. 카카오뱅크는 불과 2년 만인 2019년에는 연간 기준 흑자전환에도 성공했고 은행업을 바라보는 소비자의 인식을 전환하는 데도 기여했다는 평가를 받기 때문이다. 카카오페이 역시 가입자 수 3500만명 MAU 2000만명을 넘어섰고 2020년 3분기까지 누적거래액 47조원으로 2019년 연간 거래액을 3분기만에 달성했다. 카카오의 새로운 금융 플랫폼이 될 보험은 자동차보험과 단기소액보험 분야부터 사업을 전개할 것으로 예상된다. 전통 보험사를 카카오페이가 단숨에 따라잡기는 어렵지만 고객 접근성을 바탕으로 20,30대부터 차근차근 세력을 넓혀나갈 것으로 보인다. 금융권 관계자는 카카오뱅크가 은행권의 디지털화에 방아쇠를 당겼다고 얘기할 정도로 카카오의 금융 서비스는 영향력이 크다며 번뜩이는 상품을 내놓는 것도 카카오 금융 서비스의 강점으로 작용할 것이다고 말했다. 권지예 기자 kwon.jiyejoongang.co.kr

Summarize

카카오뱅크와 카카오페이가 금융 소비자들의 손에 익어가자 카카오가 다음 금융 플랫폼 타깃으로 보험을 겨냥해 디지털 손해보험사가 약 4500만명에 달하는 카카오톡이 카카오의 보험 영여업을 뒷받침해줄 전망이며 카카오의 새로운 금융 플랫폼이 될 보험은 자동차보험과 단기소액보험 분야부터 사업을 전개할 것으로 예상되어 전통 보험사들을 긴장하게 할 것으로 보인다.

KR-FinBert 모델



데이터

이 모델의 학습 데이터는 KR-BERT-MEDIUM , 한국어 위키백과의 텍스트, 일반 뉴스 기사, 국가법령정보센터에서 크롤링한 법률 텍스트 및 한국어 댓글 데이터 세트 에서 확장 됩니다. 전이학습 에는 파이낸셜타임스, 한국경제 등 **72개 매체의 기업 관련 경제기사 기사** 와 키움증권, 삼성증권 등 **16개 증권사의 애널리스트 리포트 가 추가됐다.** 데이터 세트에는 콘텐츠가 포함된 440,067개의 뉴스 제목과 11,237개의 분석가 보고서가 포함되어 있습니다. **총 데이터 크기는 약 13.22GB입니다.** mlm 교육의 경우 데이터를 한 줄씩 나누고 **총 수를 나눕니다. 줄 수는 6,379,315입니다.** KR-FinBert는 최대 512개, 훈련 배치 크기 32개, 학습률 5e-5로 550만 단계에 대해 훈련되며 NVIDIA TITAN XP를 사용하여 모델을 훈련하는 데 67.48시간이 걸립니다.

```
from argparse import Namespace
from sklearn.metrics import classification_report
from transformers import pipeline

from transformers import AutoTokenizer, AutoModelForSequenceClassification
from transformers import pipeline

# 사전학습 모델 로드
tokenizer = AutoTokenizer.from_pretrained("snunlp/KR-FinBert-SC")
model = AutoModelForSequenceClassification.from_pretrained("snunlp/KR-FinBert-SC")
sa = pipeline("sentiment-analysis", model=model, tokenizer=tokenizer) # 감성분류 모델 객체

totalsa_list = [] # 뉴스(제목+내용) sa(sentiment analysis) 배열 초기화
for news in summary_list:
    inputs = tokenizer(news, return_tensors='pt')
    output = model(**inputs)
    output = output.logits.tolist()[0]
    totalsa_list.append(output)

import torch
import torch.nn as nn

outputs = torch.tensor(totalsa_list) # 출력값을 텐서로 변환 # outputs

predictions = nn.functional.softmax(outputs, dim=-1) # 출력값을 확률 형태로 변환 # predict
```

```
# 출력값 확인
df_sd = pd.DataFrame(predictions.detach().numpy())
df_sd.columns = ['negative','neutral','positive']
df_sd
```

	negative	neutral	positive
0	0.153839	0.096778	0.749383
1	0.028643	0.023599	0.947758
2	0.723889	0.028944	0.247167
3	0.666395	0.292137	0.041468
4	0.133914	0.005486	0.860600
...	...	...	...

감성지수

		negative	neutral	positive	logit	intensity2	sent_index2
date	stock						
2022-05-10	카카오	0.432839	0.433353	0.133808	-1.173958	0.740866	-0.869746
2022-05-11	카카오	0.111808	0.585046	0.303146	0.997431	0.641936	0.640287
2022-05-12	카카오	0.483357	0.334482	0.182161	-0.975865	0.570090	-0.556331
2022-05-13	카카오	0.084016	0.806401	0.109583	0.265670	0.233854	0.062128
2022-05-15	카카오	0.162628	0.550157	0.287215	0.568764	0.142961	0.081311
...	...	...	...	...	...	...	...
2022-12-05	카카오	0.040662	0.686783	0.272556	1.902560	0.740866	1.409543
2022-12-06	카카오	0.311604	0.465031	0.223365	-0.332926	0.886688	-0.295201
2022-12-07	카카오	0.363960	0.503242	0.132798	-1.008211	1.195521	-1.205338
2022-12-08	카카오	0.174616	0.511377	0.314007	0.586824	0.687118	0.403217
2022-12-09	카카오	0.020227	0.723377	0.256396	2.539726	0.437530	1.111205

$$logit = \ln(\frac{P}{N})$$

$$intensity_t = \ln\left(1 + \frac{n_t}{avg\ of\ N}\right)$$

$$sentiment\_index_t = logit * intensity_t$$

기업의 특정 일자에 대한 감성 지표는
딥러닝 주가 예측 모델의 입력값으로 활용된다.

주가 데이터 및 뉴스 데이터 병합

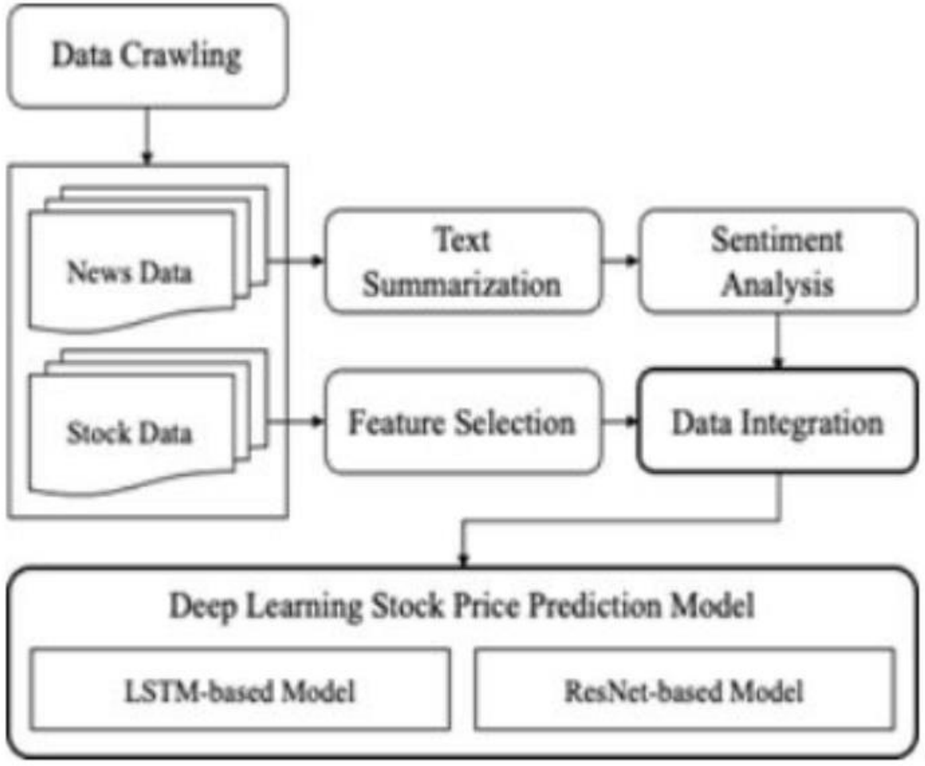


Fig. 1. Overall architecture

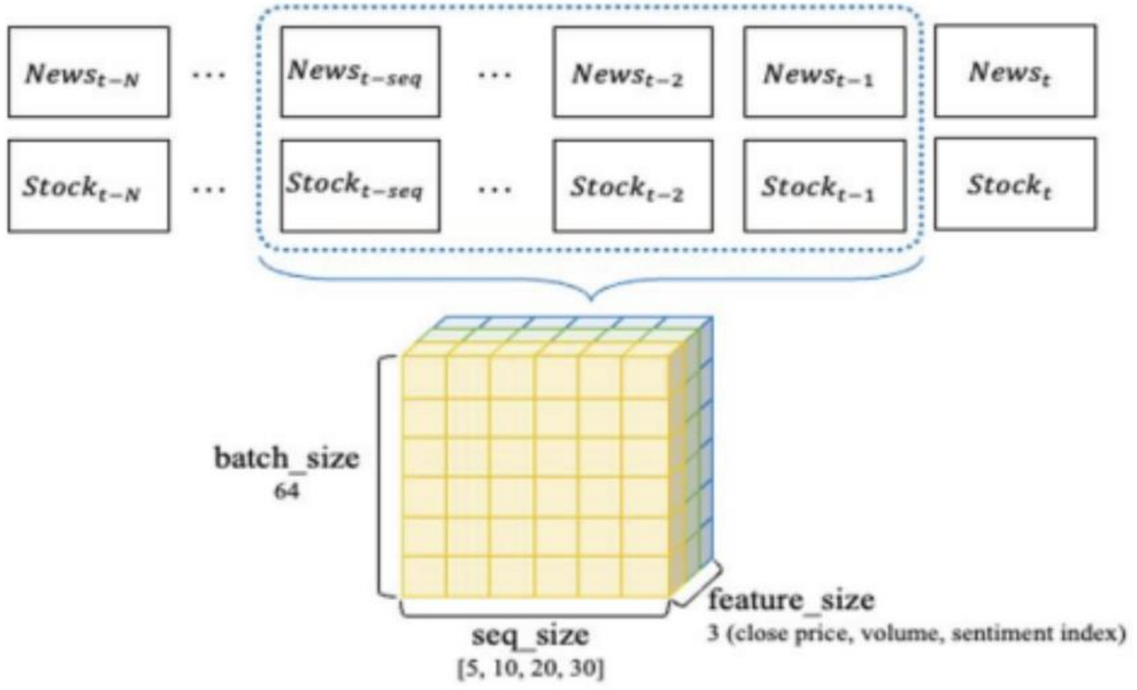


Fig. 5. Data integration based on seq_size

주가 데이터 및 뉴스 데이터 병합

날짜 column의 중복을 이용한 병합을 실시하였습니다.

date	Open	High	Low	Close	Volume	Change
2022-01-27	597000	598000	450000	505000	15946992	
2022-01-28	476000	483000	445000	450000	4559773	-0.10891
2022-02-03	458000	495500	441000	477000	2918435	0.06
2022-02-04	476500	505000	476000	504000	2088996	0.056604
2022-02-07	520000	548000	511000	548000	1911176	0.087302
2022-02-08	554000	577000	524000	542000	2583335	-0.01095
2022-02-09	550000	557000	505000	511000	2070704	-0.0572
2022-02-10	507000	507000	471500	474500	2306815	-0.07143
2022-02-11	463500	484000	457000	482000	1411055	0.015806
2022-02-14	473000	482000	459500	463000	3285402	-0.03942
2022-02-15	460000	468000	450000	451500	825406	-0.02484
2022-02-16	464000	465000	449500	455500	961669	0.008859
2022-02-17	454500	460000	447000	454500	711183	-0.0022
2022-02-18	448500	454500	448000	454500	418728	0
2022-02-21	450000	453000	447500	453000	327509	-0.0033
2022-02-22	444500	448000	440000	440000	557179	-0.0287
2022-02-23	443500	446000	438000	442000	381560	0.004545
2022-02-24	435000	438000	416000	416500	883314	-0.05769
2022-02-25	425000	430000	418500	420000	465140	0.008403
2022-02-28	404500	422000	404000	412000	970088	-0.01905

주가데이터 (+보조 데이터)

date	negative	positive	logit	intensity	sent_index
2021-12-09	0.000796	0.646682	6.700464	0.002452	0.016429
2021-12-10	0.00536	0.738768	4.925962	0.000818	0.004029
2021-12-14	0.000797	0.774153	6.878093	0.009772	0.067213
2021-12-15	0.000143	0.999253	8.84936	0.002452	0.021698
2021-12-18	0.00727	0.991713	4.915714	0.000818	0.004021
2021-12-20	3.96E-05	0.999752	10.13695	0.004083	0.041392
2021-12-21	0.001411	0.037976	3.292984	0.000818	0.002694
2021-12-22	0.000859	0.935668	6.993504	0.000818	0.005721
2021-12-23	0.052512	0.549443	2.347869	0.004898	0.0115
2021-12-27	0.000699	0.995353	7.260779	0.002452	0.017803
2022-01-02	0.000468	0.000763	0.488278	0.000818	0.000399
2022-01-03	0.021505	0.624397	3.368515	0.005712	0.019241
2022-01-04	0.001745	0.452375	5.557477	0.002452	0.013627
2022-01-05	0.001546	0.314463	5.315169	0.004083	0.021703
2022-01-06	0.006242	0.081889	2.574017	0.001635	0.004209
2022-01-09	0.384641	0.407731	0.058298	0.005712	0.000333
2022-01-10	0.003093	0.521008	5.126483	0.014623	0.074962
2022-01-11	0.111222	0.281811	0.929708	0.00815	0.007577

요약된 뉴스데이터로 뽑아낸 감성분석 데이터

```
import os, sys
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd


```

```
Mounted at /content/drive
```

```
news_df = pd.read_csv("/content/drive/MyDrive/SK하이닉스주가.csv") # 주가 데이터
stock_df =  pd.read_csv("/content/drive/MyDrive/SKSI.csv")   # 뉴스 데이터
```

```
print(news_df.head(3))
print(stock_df.head(3))
```

```
Merge_df = pd.merge(news_df,stock_df,on="date")
print(Merge_df.head(3))
Merge_df.to_csv('SK.csv', encoding='utf-8-sig')
```

주가 데이터 및 뉴스 데이터 병합

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	date	Open	High	Low	Close	Volume	Change	sent_index	KS11_Adj_Close	KQ11_Adj_Close	DAU_Adj_Close	NASDAQ_Adj_Close	SP500_Adj_Close	USD_Adj_Close	JPY_Adj_Close
0	2021-12-09	82400	84800	82400	84100	1774961	0.010817308	0.00219979919	3029.570068	1022.869995	35754.69141	15517.37012	4667.450195	1172.560059	10.314114
1	2021-12-10	83800	85900	83300	85400	1690388	0.015457788	-0.00266642122	3010.22998	1011.570007	35970.98828	15630.59961	4712.02002	1177.329956	10.378255
2	2021-12-13	86500	88200	85900	86000	2232198	0.007025761	0.0048624881	3001.659912	1005.960022	35650.94922	15413.28027	4668.970215	1178.969971	10.38644
3	2021-12-14	85200	86700	84700	85300	1006657	-0.008139535	-0.00142554413	2987.949951	1002.809998	35544.17969	15237.63965	4634.089844	1185.099976	10.434101
4	2021-12-15	84500	85300	84200	84300	651974	-0.011723329	0.0177197021	2989.389893	1003.52002	35927.42969	15565.58008	4709.850098	1184.22998	10.411362
5	2021-12-16	85400	86000	84400	85700	1014992	0.016607355	0.0055101966	3006.409912	1007.859985	35897.64063	15180.42969	4668.669922	1184.400024	10.376639
6	2021-12-17	84700	85800	84500	84600	1300076	-0.012835473	0.00113189563	3017.72998	1001.26001	35365.44141	15169.67969	4620.640137	1186.150024	10.43549
7	2021-12-20	84500	84500	82800	82900	924100	-0.020094563	0.00057210304	2963	990.51001	34932.16016	14980.94043	4568.02002	1186.390015	10.445495
8	2021-12-21	83500	83600	82400	83200	977687	0.003618818	0.000961227928	2975.030029	996.599976	35492.69922	15341.08984	4649.22998	1190.119995	10.47051
9	2021-12-22	83700	84800	83400	83900	1002461	0.008413462	-0.00421057898	2984.47998	1000.130005	35753.89063	15521.88965	4696.560059	1191.27002	10.450149
10	2021-12-23	84500	84700	83700	84200	834876	0.003575685	6.89E-08	2998.169922	1003.309998	35950.55859	15653.37012	4725.790039	1187.77002	10.407714
11	2021-12-24	84700	85800	84500	85100	1210818	0.010688836	0.000155324295	3012.429932	1007.419983	35950.55859	15653.37012	4725.790039	1185.5	10.357149
12	2021-12-27	85400	85800	84800	84800	627895	-0.003525264	0.000686300663	2999.550049	1011.359985	36302.37891	15871.25977	4791.189941	1185.920044	10.363698
13	2021-12-28	85000	85200	83800	84500	1209343	-0.003537736	0.00166591894	3020.23999	1027.439941	36398.21094	15781.71973	4786.350098	1185.650024	10.334326
14	2021-12-29	83700	84100	83300	83400	1061718	-0.013017751	0.020892222	2993.290039	1028.050049	36488.62891	15766.21973	4793.060059	1187.849976	10.349199
15	2021-12-30	83300	83900	82100	82200	1252096	-0.014388489	0.00252207305	2977.649902	1033.97998	36398.07813	15741.55957	4778.72998	1183.670044	10.297173
16	2022-01-03	82900	83200	82300	82600	778020	0.00486618	0.00338780956	2977.649902	1033.97998	36585.05859	15832.79981	4796.560059	1187.780029	10.307657
17	2022-01-04	83000	84000	82700	83500	959153	0.010895884	0.00106922176	2989.23999	1031.660034	36799.64844	15622.71973	4793.540039	1194.680054	10.358659
18	2022-01-05	85100	87100	84800	85900	3678719	0.028742515	0.00964130861	2953.969971	1009.619995	36407.10938	15100.16992	4700.580078	1196.5	10.298985
19	2022-01-06	85000	86600	84700	85600	1767632	-0.003492433	0.0035214618	2920.530029	980.299988	36236.46875	15080.86035	4696.049805	1199.25	10.326608
20	2022-01-07	85800	87200	85700	86700	1905888	0.012850467	0.0130007346	2954.889893	995.159973	36231.66016	14935.90039	4677.029785	1205.780029	10.397956
21	2022-01-10	86100	86500	83500	83800	1818497	-0.033448674	0.000652688161	2926.719971	980.380005	36068.87109	14942.83008	4670.290039	1196.880005	10.340882

OLS를 사용하여 회귀분석모형의 적절성을 위한 조건 확인

```
('상관관계 : ₩n', data.corr(method='pearson'))
```

```
칼럼명 :  Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Change'], dtype='object')
상관관계 :
              Open      High       Low     Close    Volume    Change
Open      1.000000  0.993909  0.981275  0.979587  0.150309 -0.049920
High      0.993909  1.000000  0.981586  0.989091  0.160921  0.019305
Low       0.981275  0.981586  1.000000  0.992050 -0.005265  0.026938
Close     0.979587  0.989091  0.992050  1.000000  0.057515  0.103046
Volume    0.150309  0.160921 -0.005265  0.057515  1.000000 -0.129968
Change   -0.049920  0.019305  0.026938  0.103046 -0.129968  1.000000
```

독립변수들(시가,고가,저가) 간에 강한 상관관계로
다중공선성 우려

```
result = smf.ols('Close ~ Open+High+Low+Volume', data=data).fit()
print(result.summary())
```

OLS Regression Results

Dep. Variable:	Close	R-squared:	0.995
Model:	OLS	Adj. R-squared:	0.995
Method:	Least Squares	F-statistic:	9770.
Date:	Wed, 21 Dec 2022	Prob (F-statistic):	2.37e-236
Time:	03:36:54	Log-Likelihood:	-2102.4
No. Observations:	214	AIC:	4215.
Df Residuals:	209	BIC:	4232.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-381.8313	2398.031	-0.159	0.874	-5109.260	4345.597
Open	-0.6055	0.049	-12.374	0.000	-0.702	-0.509
High	0.8586	0.058	14.775	0.000	0.744	0.973
Low	0.7459	0.059	12.696	0.000	0.630	0.862
Volume	0.0006	0.001	1.052	0.294	-0.001	0.002

Omnibus:	2.386	Durbin-Watson:	1.952
Prob(Omnibus):	0.303	Jarque-Bera (JB):	2.115
Skew:	-0.134	Prob(JB):	0.347
Kurtosis:	3.407	Cond. No.	1.09e+07

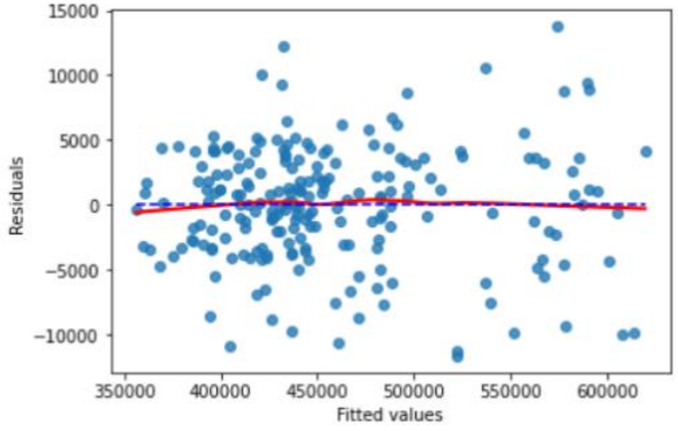
Adj. R-squared: 0.995 독립변수들의 설명력 99.5%이고
p-value: 2.37e-236 < 0.05 로 유의한 모델임을 확인

1. 독립성 확인

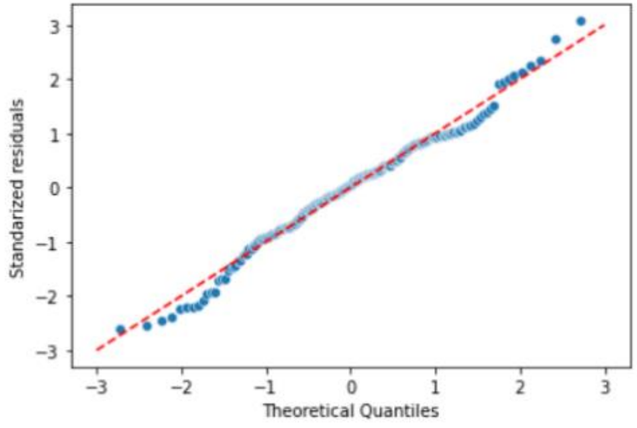
더빈왓슨검정 값이 2에 가까운 1.952이므로 자기상관이 없는
독립성을 갖고있다고 판단

OLS를 사용하여 회귀분석모형의 적절성을 위한 조건 확인

2. 선형성 확인 : 잔차가 일정하게 분포



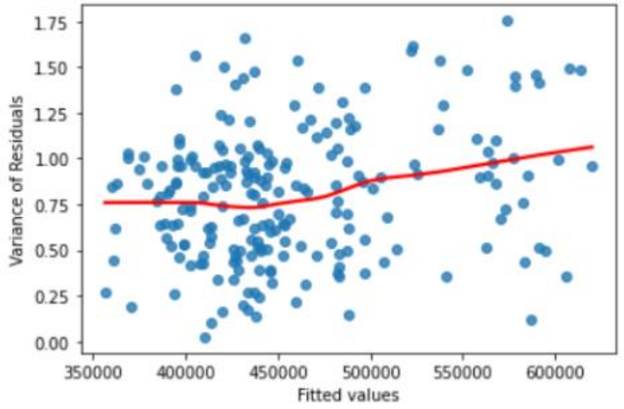
3. 정규성 확인 : Q-Q Plot, shaprio test




```
:('shaprio test : ', stats.shapiro(residual))

#pvalue=0.04236357659101486
```

Q-Q Plot 결과 값들이 참조선을 따라 선형 분포되어 보이나 shapiro 검정의 p-value 가 0.05보다 작다.

4. 등분산성 확인 : 평균선을 기준으로 일정한 패턴을 보이지 않음



5. 다중공선성 확인

VIF(Variance Inflation Factors: 분산 팽창계수)

	vif_value
0	5292.907825
1	7652.085737
2	6809.243459
3	5.382775

VIF가 10이 넘으면 다중공선성 있다고 판단하는데 시가, 고가, 저가는 자기상관이 아주높기 때문에 강한 상관관계가 있으므로 세개 변수중 하나만 채택

LSTM(Long Short-Term Memory)

장기 기억을 학습하지 못하는 기존 RNN의 단점을 보완한 모델로써 시계열 데이터 예측에 적합

```
#주가 데이터 가져오기
#2021-12-09부터 시작함. (오름차순)

# 주가 + 감성지수가 합쳐진 데이터셋
stock_data = pd.read_csv('KIA_final.csv')

#결측치 확인
print(stock_data.isnull().sum()) #null 없음
# print(stock_data.head(11))
# print(stock_data.tail(11))

#필요없는 칼럼 제거
stock_data.drop(['stock','negative','neutral','positive','logit','intensity'],axis=1,inplace=True)
print(stock_data) #246 rows x 8 columns

#정규화 작업
scaler = MinMaxScaler()  #객체 생성
# df_scaled
x_scaled = scaler.fit_transform(stock_data.drop(['High','Low','Close'],axis=1)) #stock_data의 'Date','High','Low','Close'를 제외한 모든 컬럼 정규화
y_scaled = scaler.fit_transform(stock_data[['Close']]) #stock_data의 Close 컬럼만 추출해서 정규화

#list type로 변경
x = x_scaled.tolist()
y = y_scaled.tolist()

#원복 테스트
print('스케일 값 : ',y_scaled[:5].ravel())
print('복원 값 : ',scaler.inverse_transform(y_scaled[:5]).ravel())
print('최초 값 : ',stock_data['Close'].values[:5])

print('x스케일 값 : ',x_scaled[:-1].ravel())
print('x복원 값 : ',scaler.inverse_transform(x_scaled[:-1]).ravel())
np.set_printoptions(precision=6, suppress=True)
xx=stock_data.drop(['High','Low','Close'],axis=1)
print('x최초 값 : ',(xx.values[-10:]))
#정규화 확인
print(x[-1])
# print('x복원 값 : ',scaler.inverse_transform(x[:-1]).ravel())
print('----------------')
print(y[-1])
```

주가 정보 와 감성지수가 합쳐진 데이터프레임 사용

scikit-learn의 MinMaxScaler()를 활용한
정규화 스케일링을 통해
feature들 간의 단위 차이 줄이기

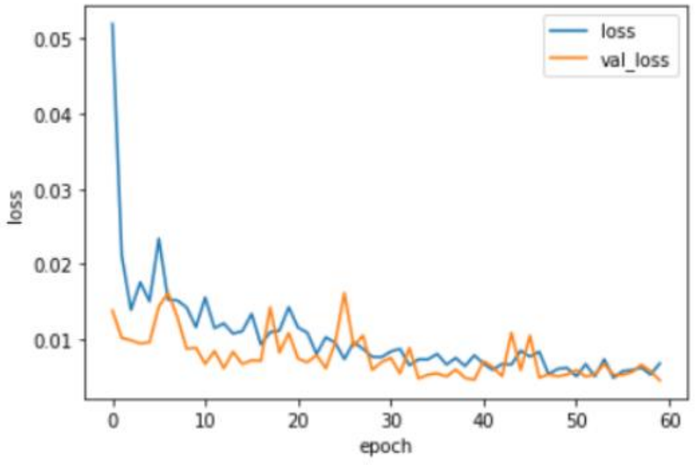
LSTM 모델 생성

```
#model 생성
model = Sequential()
# (10,11) 입력 형태를 가지는 LSTM층
model.add(LSTM(units=128, activation='tanh', return_sequences=True, input_shape=(window_size, 11)))
# 입력값의 10%를 0으로 치환하여 과적합 방지
model.add(Dropout(0.1))
model.add(LSTM(units=30, activation='tanh'))
model.add(Dropout(0.1))
model.add(Dense(units=1))
model.summary()
# 최적화 도구:adam, 손실 함수:MSE
model.compile(optimizer='adam', loss='mse', metrics=['mse'])
#train data로 학습 (과적합 방지를 위해 validation)
history = model.fit(np.array(train_x), np.array(train_y), epochs=60, batch_size=2,verbose=2, validation_split=0.2)
#evaluate
loss = model.evaluate(test_x, test_y, batch_size=2, verbose=0)
```

Dropout layer를 추가해 노드의 10%를 생략하여 과적합 방지

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 10, 50)            12400
_________________________________________________________________
dropout (Dropout)            (None, 10, 50)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 30)                9720
_________________________________________________________________
dropout_1 (Dropout)          (None, 30)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 31
=================================================================
Total params: 22,151
Trainable params: 22,151
Non-trainable params: 0
_________________________________________________________________
```

loss, val_loss 시각화



test data로 predict한 결과 시각화

r2_score : 0.77211 => 77.2% 설명력을 가진 모델

데이터분석

7. 실시간 크롤링한 10일치 데이터셋


```
# 10일치 주가 데이터 추출
def getStockData(stockName):
    start_date= datetime.today() - timedelta(20)
    end_date= datetime.today()
    df = fdr.DataReader(stockName, start_date.strftime("%Y-%m-%d"),end_date.strftime("%Y-%m-%d"))
    df_ten = df.tail(10)
    df_ten.reset_index(inplace=True)
    #FinanceDataReader를 이용해 보조 데이터 가져오기
    df_dict = {
        'KS11' : fdr.DataReader('KS11', start_date, end_date), #코스피지수
        'KQ11' : fdr.DataReader('KQ11',start_date, end_date), #코스닥지수
        'DAU' : fdr.DataReader('DJI', start_date, end_date), #다우지수
        'NASDAQ' : fdr.DataReader('IXIC', start_date, end_date), #나스닥지수
        'SP500':  fdr.DataReader('US500', start_date, end_date), #S&P 500
        'USD' : fdr.DataReader('USD/KRW', start_date, end_date), #달러당 원화 환율
        'JPY' : fdr.DataReader('JPY/KRW', start_date, end_date) #엔화 원화 환율
    }

    #각 보조데이터의 수정주가를 추출해서 stock_data에 추가하기
    for key in df_dict:
      #df_dict[key] 인덱스 해제
      extra_df = df_dict[key].rename_axis('Date').reset_index()

      #사용할 칼럼명 생성
      newName= key+'_Adj_Close'

      #Date와 Adj Close컬럼만 추출
      extra_df_cut=extra_df.loc[:,['Date','Adj Close']]

      #Adj Close컬럼명을 newName으로 변경
      extra_df_cut.rename(columns={'Adj Close':newName},inplace=True)

      #stock_data의 Date를 기준으로 extra_df_cut과 merge
      merge_outer = pd.merge(df_ten,extra_df_cut, how='left',left_on='Date', right_on='Date')

      #stock_data에 merge_outer의 newName(key+'_Adj_Close')칼럼 데이터 추가
      df_ten[newName] = merge_outer[newName]
      # df_ten.to_csv('test.csv')
    return df_ten
```

```
# 10일치 주가 데이터
stock_dataset=getStockData(stockName)
print(stock_dataset)

#결측치 확인
print(stock_dataset.isnull().sum())

#이전 행의 값으로 결측치 채우기
stock_dataset.fillna(method= 'ffill',inplace=True)

#결측치 확인
print(stock_dataset.isnull().sum()) #결측치 없음
```

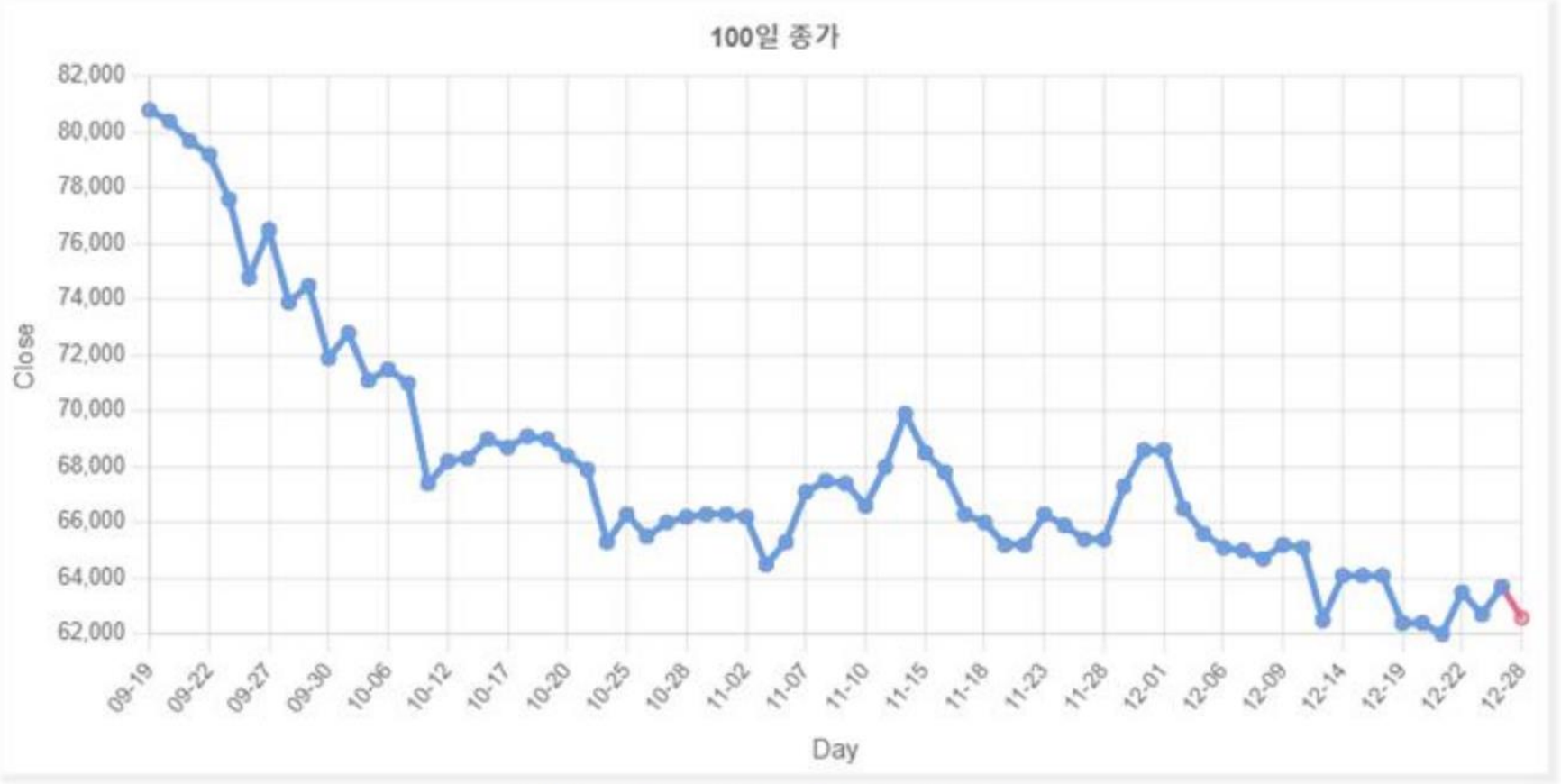
다음날 종가 예측을 위한 10일치 데이터셋

	Date	Open	High	Low	Close	Volume	Change	KS11_Adj_Close	KQ11_Adj_Close	DAU_Adj_Close	NASDAQ_Adj_Close	SP500_Adj_Close	USD_Adj_Close	JPY_Adj_Close	intensity	sent_index
0	2022-12-12	65100	65600	65000	65100	687218	-0.001534	2373.020020	715.219971	34005.039063	11143.740234	3990.560059	1303.170044	9.523555	0.648695	0.676117
1	2022-12-13	65300	65500	62100	62500	3121687	-0.039939	2372.399902	715.159973	34108.640625	11256.809570	4019.649902	1305.920044	9.497186	0.330242	-0.136421
2	2022-12-14	62200	64100	61800	64100	1960769	0.025600	2399.250000	729.000000	33966.351563	11170.889648	3995.320068	1288.979980	9.507715	0.231802	-0.341794
3	2022-12-15	63800	64800	63500	64100	878566	0.000000	2360.969971	722.679993	33202.218750	10810.530273	3895.750000	1295.680054	9.571786	0.461818	0.249656
4	2022-12-16	63200	64400	63100	64100	1126438	0.000000	2360.020020	717.409973	32920.460938	10705.410156	3852.360107	1317.099976	9.561793	1.038893	2.116147
5	2022-12-19	63700	63800	62000	62400	1296891	-0.026521	2352.169922	717.219971	32757.539063	10546.030273	3817.659912	1308.859985	9.584360	0.991640	-1.382564
6	2022-12-20	62000	62500	60900	62400	1458403	0.000000	2333.290039	703.130005	32849.738281	10547.110352	3821.620117	1301.329956	9.499869	0.889857	-2.467035
7	2022-12-21	62600	62700	61600	62000	810736	-0.006410	2328.949951	705.700012	33376.480469	10709.370117	3878.439941	1283.640015	9.732287	0.942043	2.604467
8	2022-12-22	62000	63500	62000	63500	1004592	0.024194	2356.729980	715.020020	33027.488281	10476.120117	3822.389893	1280.770020	9.675316	0.282232	0.316811
9	2022-12-23	62600	63300	62500	62700	765704	-0.012598	2313.689941	691.250000	33203.929688	10497.860352	3844.820068	1290.199951	9.747290	0.000000	0.000000

데이터분석

8. 주가 예측 그래프

기아



다음날 예측 종가는 62,582 입니다.

konlpy와 wordcloud를 사용하여 각 종목별 주요 키워드를 워드클라우드로 시각화

```
data = pd.read_csv('LG_dataset.csv')
print(data.head(3))
print(data.info())
title = data['title'].to_string()

okt = Okt()
words = okt.nouns(title) #명사추출
# print(set(words))

#불용어 제거
stopwords = ['에너지', '솔루션','특징','포토']
result =  [x for x in words if x not in stopwords and len(x) > 1]
print(result)

#빈도수 카운트
count_list = Counter(result)
print(count_list)

#배경 shape
img=Image.open('shape.png') #이미지 파일 읽어오기
imgArray=np.array(img) #픽셀 값을 배열 형태 변환

path = '/usr/share/fonts/truetype/nanum/NanumSquareRoundEB.ttf'
my_wc = WordCloud(font_path = path, relative_scaling=0.2, colormap='Reds_r',
                  background_color='white',mask=imgArray).generate_from_frequencies(count_list)
plt.figure(figsize=(12,8))
plt.imshow(my_wc)
plt.axis('off')
plt.show()
my_wc.to_file('LG_wcloud.png')
```

konlpy의 okt 형태소 분석을 사용하여 명사 추출 후
워드클라우드로 plot하여 이미지 저장

예) LG에너지 솔루션

예) 기아

프로젝트 수행 결과

프로젝트 수행 결과

자체 평가 및 보완점

자체 평가 및 보완점

자체 평가 및 보완점

한계점 & 개선방향

1. 실시간으로 뉴스데이터를 크롤링 하기때문에 출력 시간이 1분정도 걸린다.
   : multiprocessing 의 Pool을 사용하여 출력 시간 개선 필요

2. 현재 코스피 종목만 검색 가능하다.
   : 코스닥 종목까지 확대 필요

3. 다양한 모델을 사용하지 못했다.
   : Resnet, 양방향 LSTM 등 다른 모델을 활용하여 예측결과 도출 필요

4. 2020년 코로나 팬데믹 이후 코스피 지수는 약 3개월간 폭락 기간을 거친 뒤 약 1년 동안 꾸준한 상승세를 보였다. 예측 모델 학습에 영향을 주는 비대칭성 데이터의 수집을 최소화하기 위해 데이터의 수집 기간을 제한하였다.
   : TimeGAN 모델을 이용하여 데이터 증강

자체 평가

개선해야할 부분이 있지만 Pororo, KR-FinBert 등의 학습된 모델과 다양한 라이브러리를 활용해 LSTM 모델로 양호한 예측결과 도출했다.


Thank You