# 양성심 포트폴리오

파이썬 Backend & Data Analysis

Developer

# INDEX

01
Experience

- 맡은 업무
- 개발

02
Data Science

- 데이터 분석

03
Web

- Web
  - Front end
  - Back end

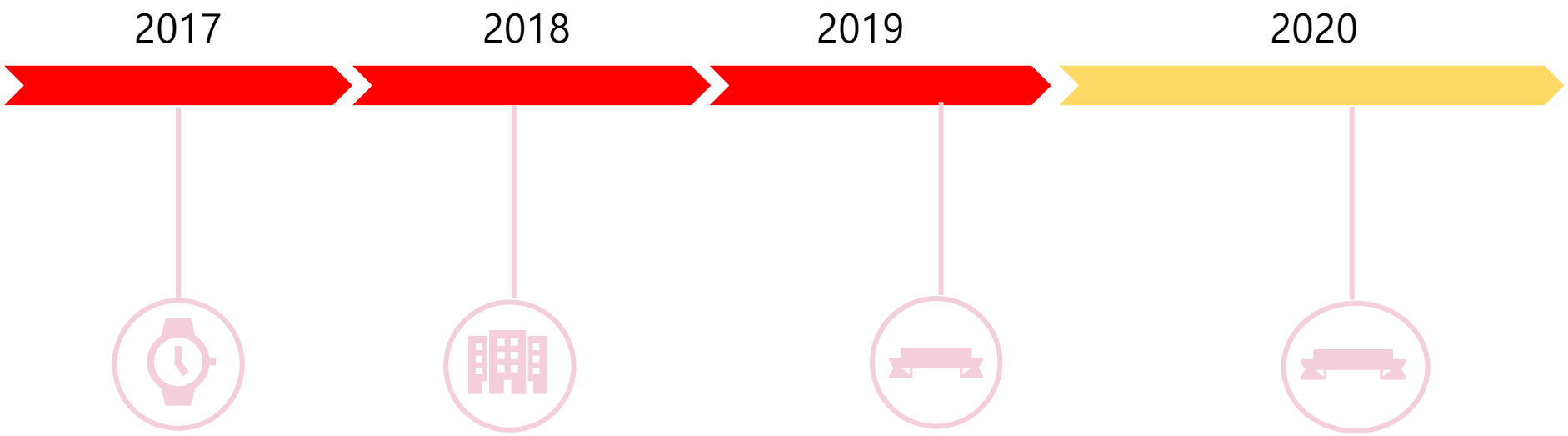
# 양성심

- 1993.01.03 생(만 27세)
- 홍익대학교 금융보험학과 (2017.03 졸업)
- 자산 운용사 시스템 트레이더 & 데이터 분석(2017.10 ~2019.12)
- 리서치 회사 개발팀 웹 개발 및 데이터 분석
  (2020.04~2020.08)

- 관리 : Github
- 파이썬 데이터 분석(머신러닝, nlp, python, node.js)
- Web(JaveScript, Vue.js, Django, Flask)

기술 스택

# 01 Experience




## 입사

- 파생상품 운용팀
- 달러 및 주가지수 선물 시스템 트레이더

## 분석

-프랍 트레이딩 용
-매매 규칙 데이터 분석

## 분석 & 개발

- 백 오피스 정산 작업 자동화
  - 시세 크롤링
  - 계좌 스크래핑
- 암호화폐 트레이딩 로직
- 평균 주가 회귀 종목 분석
- 금융텍스트 분석

## 개발팀

- 리서치 회사 R & D 개발팀
- 소셜 미디어 및 뉴스 크롤링
- 키워드 추출 작업
- 웹사이트 프론트엔드 (vue.js, mongodb)
- 설문지 웹 작업 및 실사 진행 (ASP, javascript, mssql)

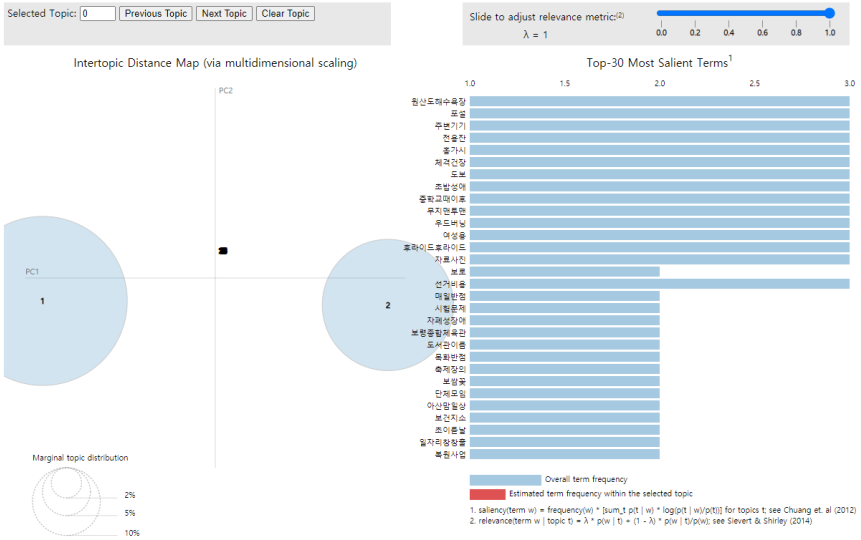
# 키워드 추출 프로세스

Node.js python puppeteer 와 selenium 을 이용하여
1. 관공서
2. 뉴스
3. 소셜미디어
크롤링으로 텍스트 데이터 수집

KoNLPy

Konlpy 의 형태소 분해를 통해 명사를 추출 후 wordBag 생성
Python 의 counter 함수를 이용하여 단어 frequency DB 생성

mongoDB




Node.js stopword 모듈 및 수작업으로 불용어 리스트 작업

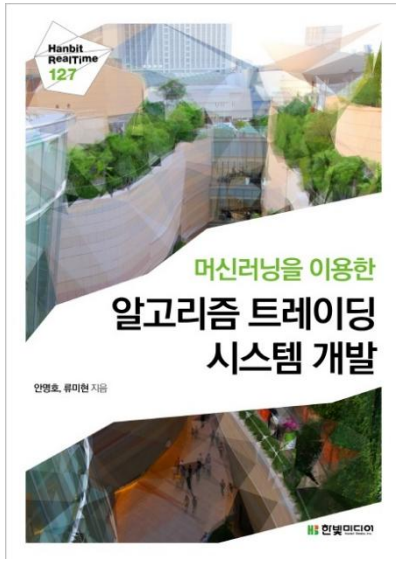
Django 에서 Chart.js 를 이용한 wordCloud 시각화

잠재 디리클레할당으로 단어를 그룹별로 묶어서 시각화

Chart.js

# 회귀 종목 추출 및 트레이딩 과정

참고자료



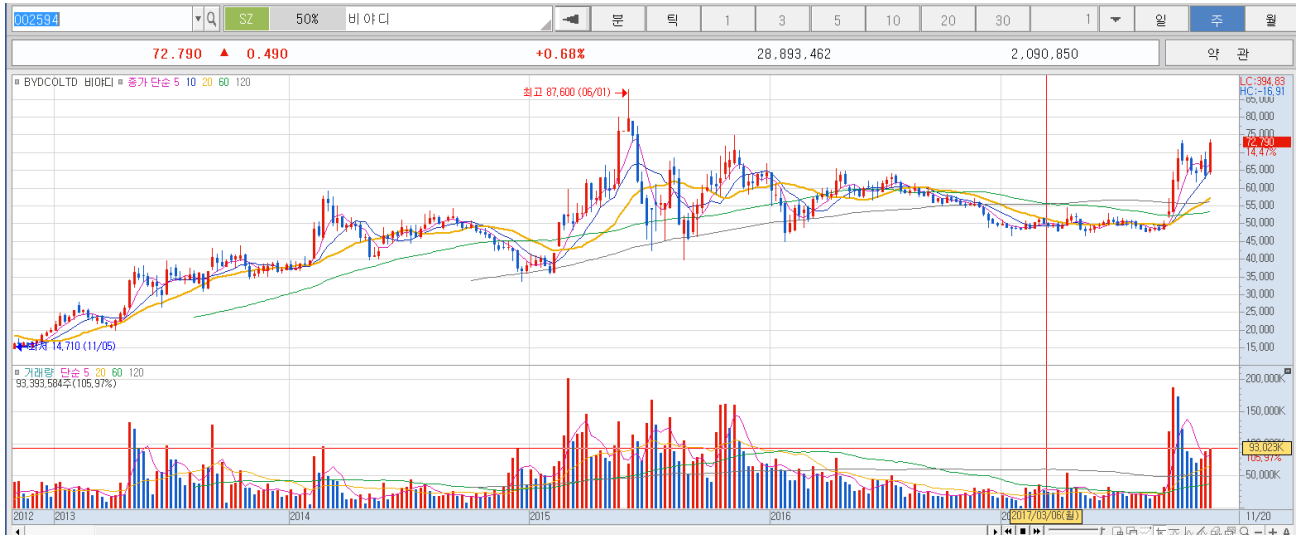


처음 파이썬을 배우기 시작했던 동기

**"나만의 로직으로 트레이딩 시스템을 개발해보자"**

-퀀트 트레이딩 아이디어를 머신러닝, 파이썬으로 구현

- 서적1. 머신러닝 개념, 평균회귀모델로 포트폴리오 빌더 구현
- 서적2. 증권사 api에서 데이터를 불러와 거래량 , 배당률 기반 투자 알고리즘을 구현하고 실제매매까지 다룬 서적




평균회귀를 보이는 종목들을 추출하기 위해 두 서적의 기술 아이디어를 혼합.

+평균회귀 성향을 보이는 종목을 추출하게 전 종목 대상으로 확대 재검색 및 주가 특이점 필터 추가
+데이터 수집 경로를 크롤링과 Api로 다각화

# 회귀 종목 추출 및 트레이딩 과정

전종목 1년치 데이터 (2019,03,20~ 2018,03,20) 의 종가와 거래량 데이터를 가지고 주가방향예측을 한다.
중간에 거래가 정지되거나 거래량이 전무해서 데이터가 전부 똑같은 3가지 종목(드림텍, 세화이앤씨..) 등을 제외시키고 진행했다.

→ 전처리 과정 생략 사용한 3가지 모형지표

↓

```
#[기본 import]
import bs4
from urllib.request import urlopen
import webbrowser
import requests
import sqlite3
import win32com.client
import timeit
import datetime
import statsmodels.tsa.stattools as ts
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import numpy as np
import pandas as pd
from warnings import simplefilter
import time
```

1.단계 전종목의 이름과 코드 넘버를 따올 웹사이트를 찾아서 크롤링 해온다. 저장은 딕셔너리에 한다.

```
stockcode = 'http://vip.mk.co.kr/newSt/rate/item_all.php'
source = urlopen(stockcode).read()
source = bs4.BeautifulSoup(source,'lxml')

td = source.findAll('td',class_='st2')
td1 = [str(i)[101:].replace("</a></td>","") for i in list(td)]
code = {}
for i in td1:
```

5. ADF , Hurst, Halflife 지수 계산하는 3가지 함수

ADF: 검정통계량이 5% , 10 % 보다 작아야 평균회귀 의미가 있다
Hurst : 0 에 가까울 수록 평균회귀현상을 보임
Half : 평균으로 되돌아오는 데 걸리는 시간. 짧을 수록 회귀성향이 강하다고 봄.

```
def calcADF(df):
    df.fillna(method = 'bfill')
    df.dropna(how='any')
    try:
        adf_result = ts.adfuller(df)
        critical_value = adf_result[4]
    except ZeroDivisionError:
        adf_result = 1
        critical_value = 1
    return [adf_result[0], critical_value['1%'],critical_value['5%'],critical_value['10

def calcHurstExponent(df):
    lags = range(2,100)
    ts = np.log(df)

    tau = [np.sqrt(np.std(np.subtract(ts[lag:],ts[:-lag]))) for lag in lags]
                        x축          y축          1차방정식
    poly = np.polyfit(np.log(lags),np.log(tau),1)
    result = poly[0]*2.0    기울기의 2배는?

    return result

def calcHalfLife(df):
    price = pd.Series(df)
    lagged_price = price.shift(1).fillna(method='bfill')
    delta = price - lagged_price
```

# 회귀 종목 추출 및 트레이딩 결과

## 결과

```
cursor1.execute('''select * from stocks''')
rows1 = cursor1.fetchall()
stock_data1 = pd.DataFrame(rows1,columns=['id','code','company','logistic','rf','svm','
stock_data1.set_index('id')
del stock_data1['id']
stock_data1['svm'].fillna('0')
stock_data1['total_score'] = stock_data1['logistic']+stock_data1['rf']+stock_data1['svm
stock_data1['rank'] = stock_data1['total_score'].rank(ascending=False)
stock_data1['way'] = np.where(stock_data1['lrpred']+stock_data1['rfpred']+stock_data1['
print(stock_data1[stock_data1['rank']<=10])
```

[결과]

| | code | company | logistic | rf | ... | svmpred | total_score | rank | way |
|---|---|---|---|---|---|---|---|---|---|
| 137 | 000370 | 한화손해보험 | 0.5517 | 0.5230 | ... | 1.0 | 1.6149 | 10.0 | B |
| 240 | 180640 | 한진칼 | 0.5230 | 0.5632 | ... | 1.0 | 1.6322 | 9.0 | B |
| 321 | 298690 | 에어부산 | 0.5152 | 0.5758 | ... | 1.0 | 1.6365 | 8.0 | B |
| 505 | 103130 | 웅진에너지 | 0.5515 | 0.5515 | ... | 1.0 | 1.6545 | 6.0 | B |
| 509 | 298040 | 효성중공업 | 0.5892 | 0.5021 | ... | 1.0 | 1.6888 | 4.0 | B |
| 678 | 074610 | 나노메딕스 | 0.5599 | 0.5543 | ... | 1.0 | 1.6741 | 5.0 | B |
| 708 | 006110 | 삼아알미늄 | 0.5320 | 0.5710 | ... | 1.0 | 1.6490 | 7.0 | B |



결과로 나온 종목 10개

1위: 삼화 페인트
5위: 한진칼

해석:
주가 움직임을 보면
모델의 정확도가 50%인
것처럼 반은 우상향하는
움직임을 보이고
반은 우하향하는 확률로
나뉘는 것을 확인.

# 02 DATA SCIENCE

## 뉴스 기사 와 주가 상관관계

## N LP 처리

01 네이버 뉴스 크롤링과 종가 데이터를 Trade Station 툴에서 가져오기

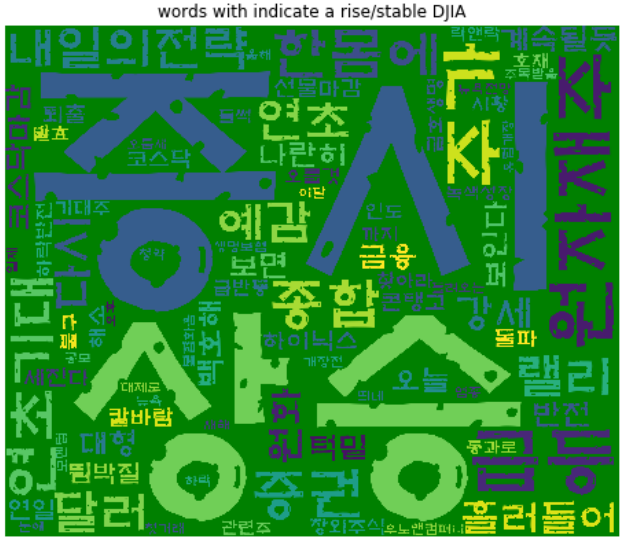
02 K200 ETF 의 시장 수익률은 kospi 를 추종하는 것, 뉴스와 kospi 와의 연관도가 얼마나 되는 지 분석

03 공휴일 제거 및 형태소 분해 및 불용어 제거 처리.

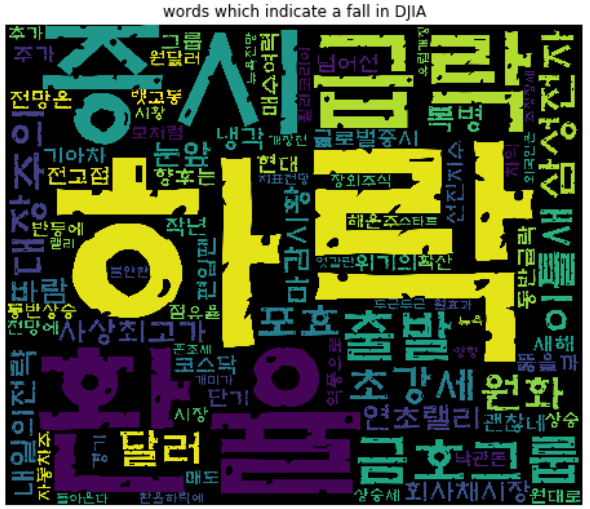
04 LogisticRegression , KNeighborsClassifier , RandomForestClassifier 세 가지 모델을 사용.


2010.01.01~2019.08.14까지
네이버 증권 섹션에서 seleninum
크롤링을 함

Word cloud
주가 상승 시 키워드




주가 하락 시 키워드



# 뉴스 기사와 주가와 상관도

참조 자료: Kaggle 의 뉴스와 주가 데이터를 분석하는 대회에서 아이디어를 착안.

미국 주가 데이터와 텍스트 데이터를 바탕으로 제시한 train data Feature에서 감성 분석한 열과 이동평균 열을 발견.

한국 주가 데이터와 주요 포털 사이트의 뉴스기사를 바탕으로 분석하고자 함.

## 베타 조정된 이동평균선 계산 위한 밑작업

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Date | Close | | | | | Date | Close | | | 베타조정된 수익 | 베타계수 |
| | 2009-12-30 | 1682.77 | | | | | 2009-12-30 | 225.75 | | | | |
| | 2010-01-04 | 1696.14 | 0.79% | | | | 2010-01-04 | 227.1 | 0.60% | | 0.27% | 0.73 |
| | 2010-01-05 | 1690.62 | -0.33% | | | | 2010-01-05 | 226.9 | -0.09% | | -0.32% | |
| | 2010-01-06 | 1705.32 | 0.87% | | | | 2010-01-06 | 228.45 | 0.68% | | 0.25% | |
| | 2010-01-07 | 1683.45 | -1.28% | | | | 2010-01-07 | 225.65 | -1.23% | | -0.08% | |
| | 2010-01-08 | 1695.26 | 0.70% | | | | 2010-01-08 | 226.5 | 0.38% | | 0.44% | |
| | 2010-01-11 | 1694.12 | -0.07% | | | | 2010-01-11 | 226.5 | 0.00% | | -0.09% | |
| | 2010-01-12 | 1698.64 | 0.27% | | | | 2010-01-12 | 226.75 | 0.11% | | 0.21% | |
| | 2010-01-13 | 1671.41 | -1.60% | | | | 2010-01-13 | 223.05 | -1.63% | | 0.04% | |
| | 2010-01-14 | 1685.77 | 0.86% | | | | 2010-01-14 | 225.7 | 1.19% | | -0.45% | |
| | 2010-01-15 | 1701.8 | 0.95% | 0.011309 | | | 2010-01-15 | 226.75 | 0.47% | 0.44% | 0.66% | |
| | 2010-01-18 | 1711.78 | 0.59% | 0.009221 | | | 2010-01-18 | 227.9 | 0.51% | 0.35% | 0.11% | |
| | 2010-01-19 | 1710.22 | -0.09% | 0.011593 | | | 2010-01-19 | 228.1 | 0.09% | 0.53% | -0.24% | |
| | 2010-01-20 | 1714.38 | 0.24% | 0.005313 | | | 2010-01-20 | 228.35 | 0.11% | -0.04% | 0.18% | |
| | 2010-01-21 | 1722.01 | 0.45% | 0.022905 | | | 2010-01-21 | 230.3 | 0.85% | 2.06% | -0.56% | |
| | 2010-01-22 | 1684.35 | -2.19% | -0.00644 | | | 2010-01-22 | 224.35 | -2.58% | -0.95% | 0.54% | |
| | 2010-01-25 | 1670.2 | -0.84% | -0.01412 | | | 2010-01-25 | 223.5 | -0.38% | -1.32% | -0.63% | |
| | 2010-01-26 | 1637.34 | -1.97% | -0.03609 | | | 2010-01-26 | 218.85 | -2.08% | -3.48% | 0.15% | |
| | 2010-01-27 | 1625.48 | -0.72% | -0.02748 | | | 2010-01-27 | 217 | -0.85% | -2.71% | 0.16% | |
| | 2010-01-28 | 1642.43 | 1.04% | -0.02571 | | | 2010-01-28 | 219.8 | 1.29% | -2.61% | -0.34% | |
| | 2010-01-29 | 1602.43 | -2.44% | -0.05839 | | | 2010-01-29 | 213.85 | -2.71% | -5.69% | 0.37% | |
| | 2010-02-01 | 1606.44 | 0.25% | -0.06154 | | | 2010-02-01 | 214.65 | 0.37% | -5.81% | -0.17% | |
| | 2010-02-02 | 1595.81 | -0.66% | -0.0669 | | | 2010-02-02 | 213.4 | -0.58% | -6.44% | -0.11% | |
| | 2010-02-03 | 1615.02 | 1.20% | -0.05796 | | | 2010-02-03 | 215.85 | 1.15% | -5.47% | 0.08% | |
| | 2010-02-04 | 1616.42 | 0.09% | -0.06132 | | | 2010-02-04 | 215.6 | -0.12% | -6.38% | 0.28% | |
| | 2010-02-05 | 1567.12 | -3.05% | -0.0696 | | | 2010-02-05 | 209.45 | -2.85% | -6.64% | -0.27% | |

## 거래량과 이동평균선 feature 계산

```
import pandas as pd
import numpy as np
import datetime
from keras.models import Sequential
from keras.layers import Activation, LSTM, Dense, BatchNormalization
from keras.optimizers import sgd
import pandas as pd
import numpy as np

data = pd.read_csv('C:/Users/FOS_08/Documents/k200.csv')
pd.options.display.max_columns=20



windows = [5,10,20,60,120]
for i in windows:
    data['close_ma{}'.format(str(i))] = data['Close'].rolling(window=i).mean()
    data['volume_ma{}'.format(str(i))] = data['Vol'].rolling(window=i).mean()

거래량과 종가의 이동평균선을 계산
```

# 뉴스 기사와 주가와 상관도

```
from sklearn.metrics import roc_curve
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split, KFold

#한국어 텍스트 konlpy 로 형태소 추출한다음
#불용어 리스트에서 전부 제거시키기
okt = Okt()


trainheadline = []
for i in range(0,len(before_train.index)):
    trainheadline.append(' '.join(str(x) for x in before_train.iloc[i,1:39]))

#형태소를 다 분리시킴 불용어 제거

stopwords = []
with open('../Documents/stop_words.txt','r') as res:
    k = res.readlines()
    for i in k:
        i = i.replace('\n','')
        stopwords.append(i)


trainhead = []
for i in trainheadline:
    letters_only = okt.morphs(i)
    temp = []
    for ii in letters_only:
        if ii not in stopwords:
            letters = re.sub("[^ㄱ-ㅎㅏ-ㅣ가-힣]"," ",ii)
            letterss = re.sub('[\s]','',letters)
            if letterss != '':
                temp.append(letterss)
            temp2 = ' '.join(str(x) for x in set(temp))
    trainhead.append(temp2)


#test 도 전처리

testheadline = []
```

3. 모델 학습

LogisticRegression, kNeighborsClassifiers, RandomForestClassifier 이 세모델로 학습 시킨 후 가장 정확도가 높게 나온 모델을 채택하기

```
basicvectorizer = CountVectorizer()
basictrain = basicvectorizer.fit_transform(trainhead)
basictest = basicvectorizer.transform(testhead)


#tfidvectorizer로  할때
basicvectorizer = TfidfVectorizer()
basictrain = basicvectorizer.fit_transform(trainhead)
basictest = basicvectorizer.transform(testhead)

Classifiers = [
        LogisticRegression(C=1,solver='liblinear',max_iter=5000)
        ,KNeighborsClassifier(3),
        RandomForestClassifier(n_estimators=2000,max_depth=9)]
```

불용어 사전을 텍스트 파일로 만들어서 거른후

정규식으로 2차 필터

vectorizer  위해 리스트를 문자열로 묶기.

## 결론

feature_importance

| | imp | col |
|---|---|---|
| 8 | 2 | sentiment |
| 11 | 18 | close_rolling10 |
| 7 | 19 | close_beta10 |
| 10 | 20 | open_raw10 |
| 12 | 20 | close_rolling5 |
| 3 | 43 | beta_close_return01 |
| 5 | 57 | beta_open_return01 |
| 0 | 59 | Open |
| 9 | 64 | close_raw10 |
| 2 | 70 | Vol |
| 14 | 88 | JustCloseReturn01 |
| 1 | 94 | Close |
| 13 | 97 | BetaCloseReturn01 |
| 16 | 171 | JustOpenReturn |
| 6 | 184 | just_open_return |
| 15 | 259 | BetaOpenReturn01 |
| 4 | 485 | just_close_return01 |

결론: 하지만, 뉴스 기사 분석을 feature로한 분석에서 정확도가 제일 낮게 나옴.
하루 전 수익률이 다음날 방향과 제일 유사도가 높다고 나옴

하루 전의 종가가 다음날의 움직임에 영향을 미친다.

# 그 외 데이터 분석

| | |
|---|---|
| Titanic_korean | deleted |
| [illegible]e_date | New_folder |
| fifa | New_folder |
| funda_sales | add |
| jeju_transportation | add newone |
| job_recommandation | newly added |
| keras_image_classification | newly added |
| movie | add |
| smishing | newly added |

**Readme**

1. Dda-reong

서울시 날씨와 따릉이 데이터로 1시간 후의 자전거 대수를 예측

2. Financial_visualization

네이버 각종 상품선물 및 국제주가지수 등의 데이터를 시각화

3. NLP_pizza

kaggle pizza 데이터를 자연어 처리해서 주문에 성공여부를 예측

4. NLP_practice1~ NLP_stock_news

네이버 신문기사를 크롤링해서 실제 종합주가지수와 얼마나 연관성이 있는지 예측

5. Titanic_Korean

타이타닉 승객 데이터로 생존여부 예측

6. Fifa

해외 축구 선수 이적료 예측 미션

7. Funda_sales

펀다 상점의 고객 매출 데이터를 바탕으로 1분기 매출액 예상

8. jeju-transporation

제주도 버스 승차시간 별 승객 수 예측

9. keras-image-classification

케라스로 사람 얼굴 이미지 분류

10. Movie

영화 감독, 관객 수등의 데이터를 가지고 영화 관객 수 예측

11. smishing

금융 문자 텍스트를 분석하여 스미싱 여부 판단

양성심
(promise6424)
쉽게 까먹어서 적는 글 EDIT
프로필
글쓰기 관리·통계

카테고리
전체보기 (239) EDIT
python (154)
Django (19)
data science (52)
tensorflow (17)
QUI
그린 연습 (35)
크롤링 (21)
정규표현식 (4)
웹
Excel (22)
VBA (8)
SQL (14)
기타
hack (3)
JAVA (14)
jsp (0)
안드로이드 (6)
리눅스(백엔드) (1)

data science 52개의 글 목록닫기

| 글 제목 | 조회수 | 작성일 |
|---|---|---|
| LSTM 시계열 예측 검색비허용 | 1 | 2019. 11. 1. |
| naive bayes 분류기 검색비허용 | 0 | 2019. 9. 18. |
| movie review prediction 검색비허용 | 0 | 2019. 9. 6. |
| 손실함수와 활성화 함수의 조합 검색비허용 | 0 | 2019. 9. 6. |
| stock_news_final 검색비허용 | 1 | 2019. 9. 5. |

글관리 열기 5줄 보기

1 2 3 4 5 6 7 8 9 10 다음 >

data science

## movie review prediction

양성심 2019. 9. 6. 14:06 URL 복사 통계

#karas의 LSTM과 embedding을 어떻게 쌓아올리는 것인지 알기 위함.
# natural language processs 가공을 어떻게 해서 모델에 집어넣어야 하는 지 알기 위함.

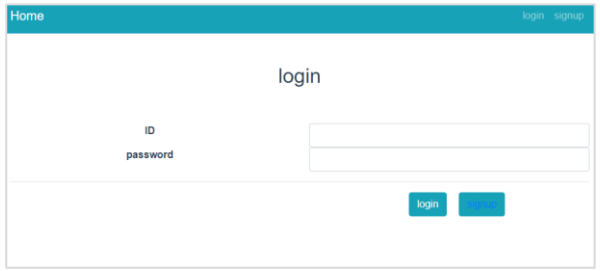
데이터 구성
train.csv
test.csv
로 구성

```
import numpy as np
```

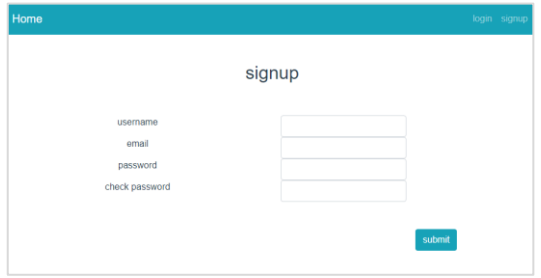
# 03 Web

Front-end: vue.js

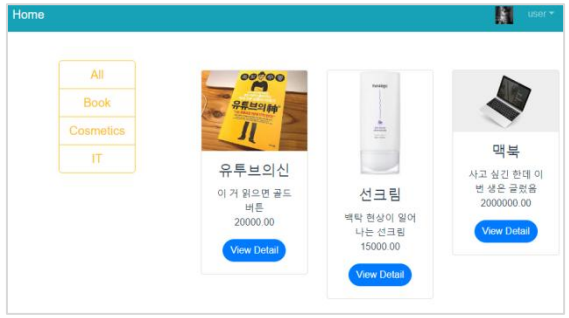
Back-end: Django




login

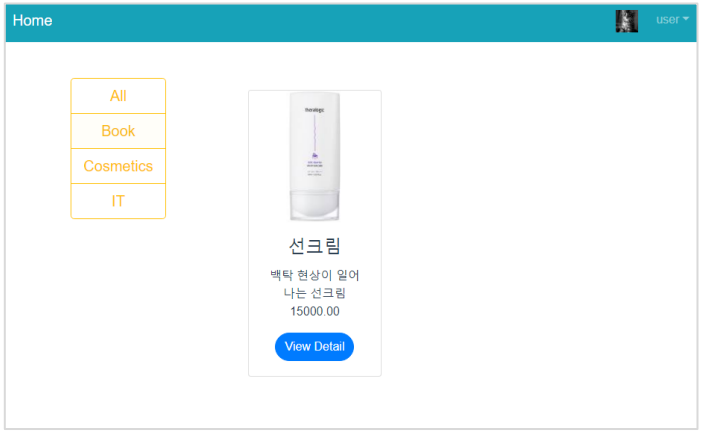



Signup




MainPage


Cart

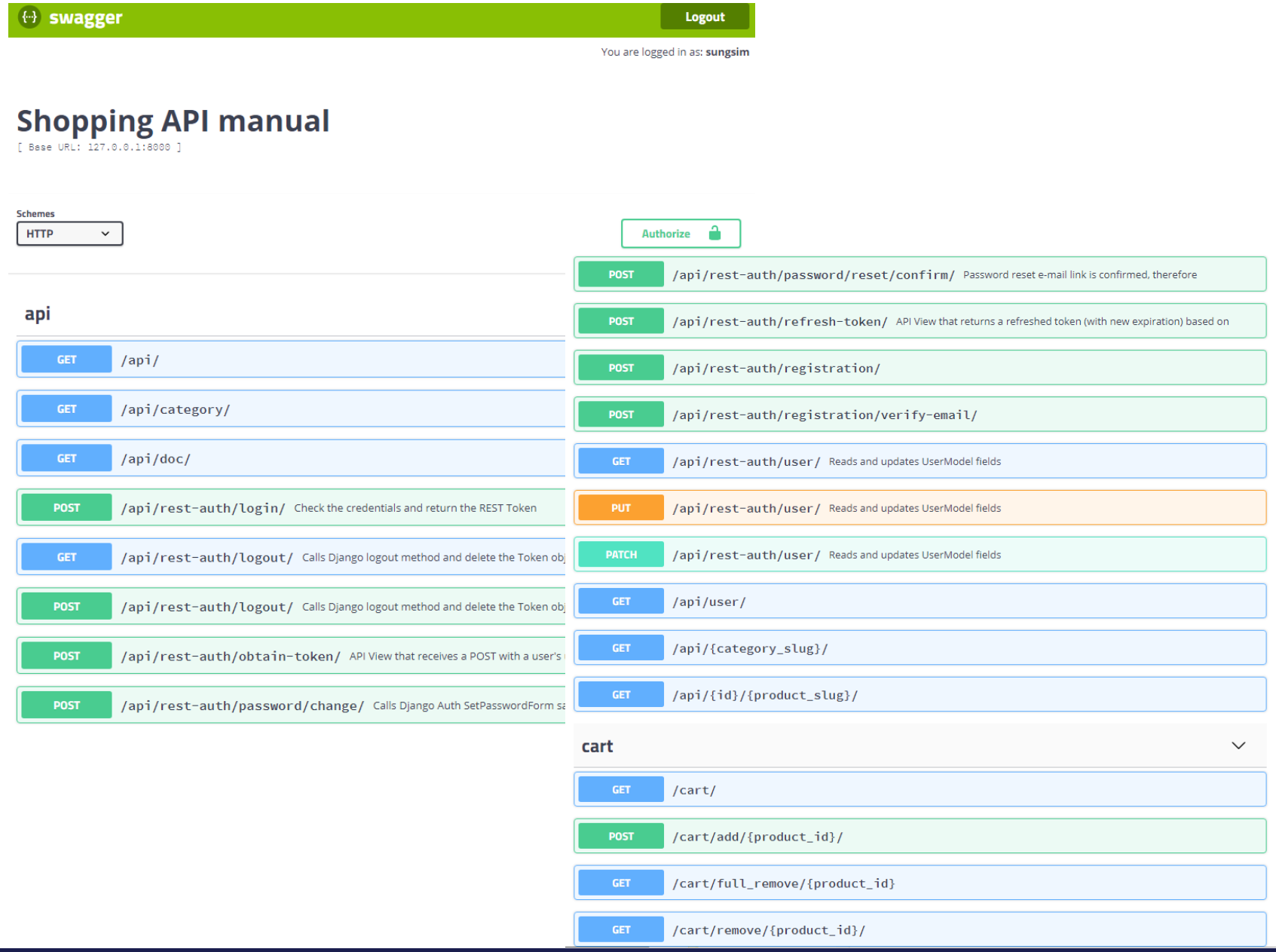



Categories

# 03 Web

Front-end: vue.js

Back-end: Django



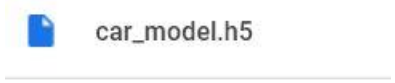
# 03 Web

## Django

### 1. 구글에서 크롤링을 통해 자동차 사진 파일 수집

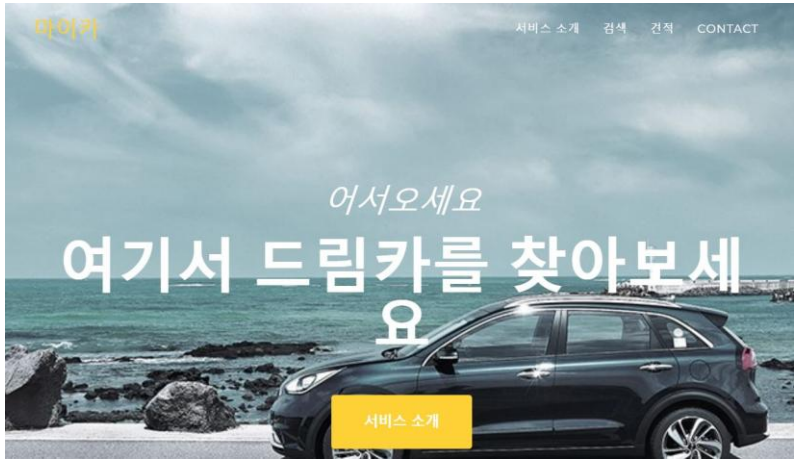
```
# 자동차 중고거래 사이트에서 자동차 사진과 가격, 주행거리 데이터들을 받기
# 사진은 따로 이미지 폴더를 만들어서 저장하고 나머지 가격, 주행거리등의 텍스트들은
# sqlite db로 만들어 놓기
# 사진이 다운로드 받아지지 않아서 클릭한번 한 후 타고 들어가 사진 다운로드 시켜야 함
# 차량 이름 과 그 외 정보들은 구글링해서 찾아야하나봄.

driver = webdriver.Chrome(r'D:\chromedriver_win32\chromedriver.exe')

driver.get(url='https://www.carisyou.com/car/')

time.sleep(10)

count = 0
while True:
    try:
        more_button = driver.find_element_by_xpath("//*[@id='moreSrhDiv']/a").click()
        count += 1
        print(count)
    except selenium.common.exceptions.ElementNotInteractableException:
        break
image_url = 'https://www.carisyou.com/car/'
source = urlopen(image_url).read()
source = bs4.BeautifulSoup(source,'html5lib')
img = source.find_all('img')
img_list = []
for i in img:
    ii = str(i)
    start = ii.find('src')+5
    k = re.compile('[.]\S{3}"')
    starts = k.search(ii[start:]).start()
    ends = k.search(ii[start:]).end()-1
    img_list.append(ii[start:][:ends])

for idx, i in enumerate(img_list):
```

### 2. google colab에서 차 이미지 처리




### 3. 마이카 사이트 메인 페이지

### 2. 서비스 소개

피부 타입별 건조한 수치를 입력 후
추천 화장 품 목록 제시

## 고객님의 피부 타입 유형을 선택해 주세요

- This field is required.

Dry: ---------

- This field is required.

Sensitive: ---------

- This field is required.

Oily: ---------

제출

## 자가 피부 타입 진단 테스트

피부타입에 맞는 화장품을 추천하겠습니다.

건조한 정도
3

민감한 정도
1

기름진 정도
1

## 추천 화장품 목록입니다.

- 356 => suncare

jungle,decoration,buffet,duty,continuation,shout

- 529 => maskpack

acquaintance,recognize,relationship,zone,circulate

- 141 => maskpack

piano paragraph nose insert layer exaggerate

# MSSQL 실습팀과제

## 데이터베이스

## 학기과제

-GS25편의점 사례연구-

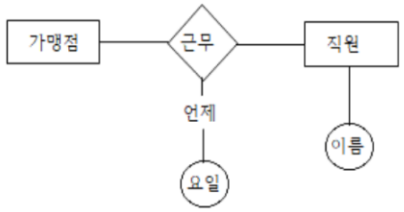
과목명 : 데이터베이스
담당교수 : 정일주 교수님
제출일자 : 2015.11.30.(월)
제출자 : 검은콩 조
조원 : 양성심, 유상미

### 3. 지역 개체연관 모델의 설계

앞서 제시한 Q1에서 Q9까지의 가시적 문의에 대하여, 각각 L1에서 L9까지의 지역 개체연관모델을 작성한다.

Q1 : 매장에서 근무하는 이름이 '전예은'인 직원이 근무하는 요일을 제시하라.

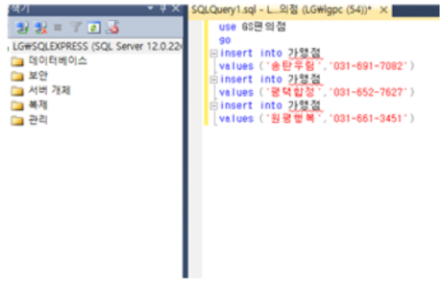
L1:




Q2 : 제품명이 첫술을 취급하고 있는 가맹점의 이름과 제품의 가격을 제시하라.

L2:


```
use GS편의점
go
insert into 가맹점
values ('송탄우림','031-691-7082')
insert into 가맹점
values ('평택합정','031-652-7627')
insert into 가맹점
values ('원평행복','031-661-3451')
```



# 능력 스킬

금융지식 50

프로그래밍 60

web 70

데이터 사이언스 80

80

* 금융지식
금융 전공입니다.

60

* 프로그래밍
파이썬 및
자바스크립트
프로그래밍

70

* Web
-front end
vue.js
-back end
django

70

* 데이터 사이언스
Numpy
Pandas
Natural language

# CONTACT

**PHONE**
**010-7650-7082**

**E-MAIL**
**promise6424@naver.com**

https://github.com/YangSungSim

https://blog.naver.com/promise6424

# THANK YOU

Promising Developer