

2021

Flow of oil price

[유가 빅데이터 분석]



오일플로우(Oil Flow)

정상훈, 이동현, 김태우

Contents

1 개요

- 주제 및 목적
- 연구 의의
- 업무 분장
- 일정 관리
- 개발 환경

2 데이터 수집 및 전처리

- 데이터 출처
- 데이터 수집
- 전처리

3 데이터 분석

- 국내 유가
- 국제 유가
- 관계성 확인

4 결론

- 결론
- 연구의 한계점

개요

주제 및 목적

연구 의의

업무 분장

일정 관리

개발 환경

SUBJECT & PURPOSE

Subject :

유류비 변동성

Purpose :

국제유가에 따른 국내 유류비

SIGNIFICANCE

[illegible]

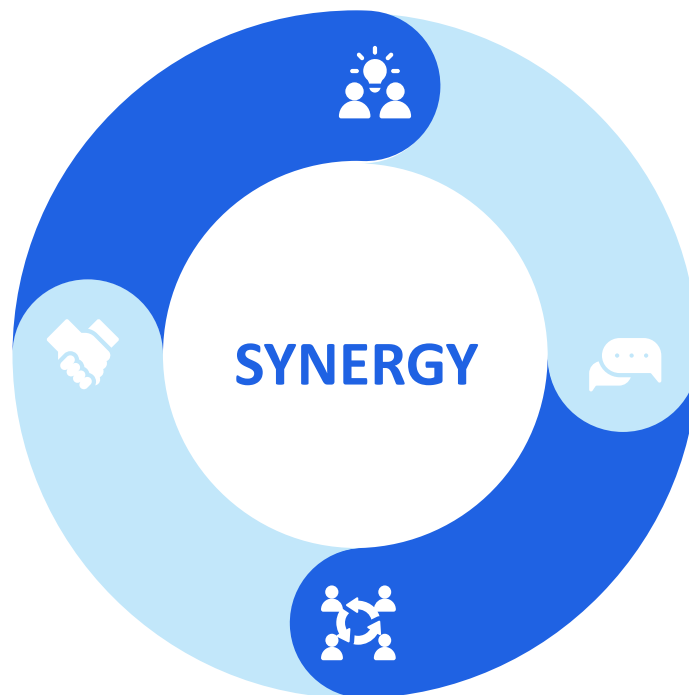
업무 분장

이 동 현

분석 과제 도출
데이터 탐색
데이터 준비
분석 알고리즘 작성
분석 모델 평가
발표

정 상 훈

분석 과제 도출
데이터 탐색
데이터 준비
분석 알고리즘 작성
분석 모델 평가
GitHub 관리



김 태 우

분석 과제 도출
데이터 준비
분석 알고리즘 작성
분석 모델 평가
PPT 작성

업무 분장



GitHub interface showing the repository **kkkerry / teamproject** (Public).

Navigation tabs: <> Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights.

Repository details: main branch, 1 branch, 0 tags. Buttons: Go to file, Add file, Code.

Commit history:

Commit	Message	Time
kkkerry	웹크롤링 수정본_1	f9dec32 3 days ago 6 commits
data	Add files via upload	4 days ago
etc_source_file	Add files via upload	4 days ago
img	Add files via upload	4 days ago
master_source_file	Add files via upload	4 days ago
ppt_files	Add files via upload	4 days ago
2012.04.02 - 2012.04.16.txt	웹크롤링 수정본_1	3 days ago
README.md	Update README.md	4 days ago
oilflow_정상훈.ipynb	정상훈 웹크롤링 추가 최신업데이트	3 days ago
oilflow_정상훈0907_2.ipynb	Add files via upload	4 days ago
oilflow_정상훈0910_수정본.ipynb	웹크롤링 수정본_1	3 days ago

SCHEDULE

활동 내용	9 / 10	9 / 13	9 / 14	9 / 15	9 / 16	9 / 17	9 / 23	9 / 24	9 / 27	9 / 28
<div> <div></div> <div>계획</div> <div></div> <div>완료</div> </div>										
1. 기획 및 설계										
1-1. 주제 선정 및 요구사항	<div></div>									
1-2. 설계, 데이터 수집	<div></div>									
2. 서론 및 이론적 배경										
2-1. 연구 배경 및 목적	<div></div>	<div></div>								
2-2. 연구 내용 및 방법	<div></div>	<div></div>								
2-3. 이론적 배경	<div></div>	<div></div>								
3. 연구 가설 및 조사 설계										
3-1. 연구 가설		<div></div>								
3-2. 변수의 조작적 정의		<div></div>	<div></div>	<div></div>						
4. 실증 분석 및 결론										
4-1. 자료 수집 및 표본의 특성			<div></div>	<div></div>	<div></div>					
4-2. 기술 통계 분석 및 시각화				<div></div>	<div></div>	<div></div>	<div></div>			
4-3. 데이터 분석 및 시각화				<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	
4-4. 추가 분석 및 시각화				<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	
5. 발표 자료 작성										
5-1. ppt 작성							<div></div>	<div></div>	<div></div>	<div></div>

OS

Windows 10 Pro

Language

Python 3.9.6

IDE

Jupyter Notebook 6.3.0

Open Source

Pandas, Numpy, Matplotlib, wordcloud,
BeautifulSoup, geopy, pyautogui

데이터 수집 및
전처리

데이터 출처
데이터 수집
전처리

데이터 수집

한국은행 - 경제통계시스템



<https://ecos.bok.or.kr/>

- 환율 정보

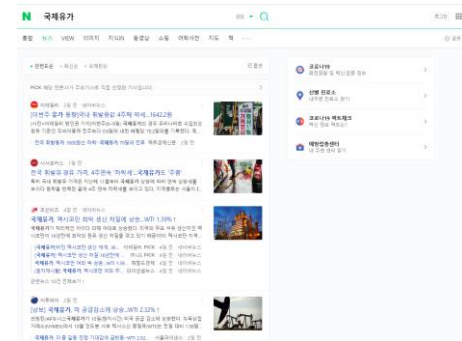
한국석유공사 - 오피넷



<https://www.opinet.co.kr/>

- 국제 유가 요금 정보
- 정유사 공급가격 정보
- 주유소 판매가격 정보

네이버 - 네이버 뉴스



<https://www.naver.com>

- '국제유가' 관련 기사

Download

Opinet [싼 주유소찾기](#) [국내유가통계](#) [유가관련정보](#) [불법행위공표](#) [이용안내](#) [로그인](#) [회원가입](#)

Home [국내유가통계](#) [주유소](#)

평균판매가격 [연세유평균가격](#)

평균판매가격 전체 주유소의 제품별 평균 판매가격(부가세포함)

제품별 지역별 상표별 형태별 지역상표별

구분 ☒ 일간 ☐ 주간 ☐ 월간 ☐ 분기 ☐ 연간

기간 2010년 1월 1일 ~ 2021년 8월 31일

제품 전체선택 ☒ 고급휘발유 ☒ 보통휘발유 ☒ 자동차용경유 ☒ 실내등유 ☒ 보일러등유

유가자유화 이전보기: 1964년 ~ 1996년 유가이력: 1964년 ~ 현재

2010년 01월 01일 ~ 2021년 08월 31일 (원/리터) ☒ 최고값 ☐ 최저값 ☐ 차트보기 ☐ 화면연세 ☐ 역셀저장 ☒ CSV저장

구분	고급휘발유	보통휘발유	자동차용경유	실내등유	보일러등유
2010년01월01일	1,848.26	1,641.20	1,433.24	1,019.88	1,001.23
2010년01월02일	1,848.43	1,641.12	1,433.12	1,020.11	1,005.76
2010년01월03일	1,853.39	1,641.03	1,433.11	1,021.32	1,004.32
2010년01월04일	1,850.71	1,641.28	1,432.97	1,020.85	1,004.81
2010년01월05일	1,866.87	1,644.89	1,435.79	1,026.02	1,011.73
2010년01월06일	1,866.39	1,647.25	1,438.20	1,028.59	1,013.76
2010년01월07일	1,867.79	1,649.43	1,440.07	1,030.25	1,015.46
2010년01월08일	1,867.56	1,651.81	1,442.09	1,031.89	1,021.56
2010년01월09일	1,867.80	1,654.95	1,444.78	1,035.86	1,023.70

```
# 다운로드한 파일 불러오기
station = pd.read_csv('data/주유소_평균판매가격_제품별.csv', encoding='cp949')

# row데이터 : 2010년 1월 2일 ~ 2021년 9월 1일 (날짜 조정) // 주유소 금액은 하루전 가격으로 발표
station['날짜'] = pd.to_datetime(station['구분'].str.replace('년', '-').str.replace('월', '-').str.replace('일', ''))
station['날짜'] = station['날짜'] - datetime.timedelta(days=1)
station.drop(['구분'], axis=1, inplace=True)

# 보통휘발유의 30일 이전대비 증감비를 계산 // 30일 이전 데이터 없는 경우 NaN 처리
for idx in range(len(station)):
    try:
        temp = ((station.loc[idx, '보통휘발유'] - station.loc[idx-30, '보통휘발유']) / station.loc[idx-30, '보통휘발유']).round(6)
        station.loc[idx, '보회-30'] = temp
    except KeyError as e:
        station.loc[idx, '보회-30'] = np.nan

station = station[['날짜', '고급휘발유', '보통휘발유', '자동차용경유', '실내등유', '보회-30']]
station_df = station.copy() # 새로운 DF에 복사

# 파일 저장
station_df.to_csv('data/station_df.csv', index=None)
```

```
# 다운로드한 파일 불러오기
refinery = pd.read_csv('data/정유사_주간공급가격_제품별.csv', encoding='cp949')

# 정유사 금액은 전주의 월요일(-10)~토요일(-4) 기준 -> 전주 수요일 기준으로 원의날짜 조정
refinery['날짜'] = '2009-12-30'
refinery['날짜'] = pd.to_datetime(refinery['날짜'])
for idx in range(1, len(refinery)):
    refinery['날짜'][idx] = refinery['날짜'][0] + datetime.timedelta(days=(7 * idx))
refinery.drop(0, inplace=True)
refinery.index = range(refinery.shape[0])

# 보통휘발유의 4주전 대비 증감비를 계산 : 30일 이전 데이터 없는 경우 NaN 처리
for idx in range(len(refinery)):
    try:
        temp = ((refinery.loc[idx, '보통휘발유'] - refinery.loc[idx-4, '보통휘발유']) / refinery.loc[idx-4, '보통휘발유']).round(6)
        refinery.loc[idx, '보회-4w'] = temp
    except KeyError as e:
        refinery.loc[idx, '보회-4w'] = np.nan

refinery = refinery[['날짜', '고급휘발유', '보통휘발유', '자동차용경유', '실내등유', '보회-4w']]
refinery_df = refinery.copy() # 새로운 DF에 복사

# 파일 저장
refinery_df.to_csv('data/refinery_df.csv', index=None)
```

Web crawling

Opinet [싼 주유소찾기](#) [국내유가통계](#) [유가관련정보](#) [불법행위공표](#) [이용안내](#) [로그인](#) [회원가입](#)

Home [국내유가통계](#) [유가내려받기](#) ▼

유가내려받기

사업자 정보 및 유가 정보

일반 현재

사업자 기본정보 (현재기준)

구분 ☒ 주유소 ☐ 자동차충전소 [엑셀저장](#) [CSV저장](#)

- 제공정보: 고유번호, 상호, 상표, 주소, 전화번호, 셀프구분, 품질인증주유소, 전산보고주유소 (셀프, 품질인증주유소, 전산보고주유소 여부는 주유소에만 해당됨)

사업자별 현재 판매가격

구분 ☒ 주유소 ☐ 자동차충전소 [엑셀저장](#) [CSV저장](#)

- 제공정보: 고유번호, 상호, 상표, 주소, 제품가격, 셀프여부

사업자별 과거 판매가격 (2008.4.15~)

구분 ☒ 주유소 ☐ 자동차충전소

유가 ☒ 일일유가 ☐ 주간평균 ☐ 월간평균

기간 20210907 ~ 20210907

지역 시/도 시/군/구 [텍스트저장](#) [CSV저장](#)

· 제공정보: 고유번호, 상호, 상표, 주소, 기간, 제품가격, 셀프여부

· 데이터량에 따라 아래와 같이 지역 선택시 1회 최대 조회기간이 제한됨

 - (전국)1개월, (시도)6개월, (시군구)1년

· 텍스트 파일의 내용을 복사하여 엑셀에 붙여넣기 하시면 엑셀파일로 저장 가능

· 가격정보를 제외한 업체 기본정보는 조회 시점의 자료 기준이므로 과거 특정 시점과는 차이가 있을 수 있으며, 기본정보가 불확실한 경우 공란으로 표시됨

```
import pyautogui
import time

pyautogui.click(x=650, y=809, clicks=2)
date_list = pd.read_csv('data/date.csv')
date_list = np.array(date_list[3:140])

for date in date_list:
    # 시작날짜 입력창 더블클릭
    start = str(date[0])
    end = str(date[1])
    pyautogui.click(x=1045, y=547, clicks=1)
    pyautogui.click(x=655, y=809, clicks=2)
    pyautogui.typewrite('')
    pyautogui.typewrite(start)
    time.sleep(1)
    # 종료날짜 입력창 더블클릭
    pyautogui.click(x=775, y=807, clicks=2)
    time.sleep(1)
    pyautogui.typewrite('')
    pyautogui.typewrite(end)
    # CSV저장 버튼 클릭
    pyautogui.click(x=1477, y=859, clicks=1)
    time.sleep(3)
    # 계속진행하시겠습니까? 확인버튼 클릭
    pyautogui.click(x=1063, y=208, clicks=1)
    time.sleep(80)
```

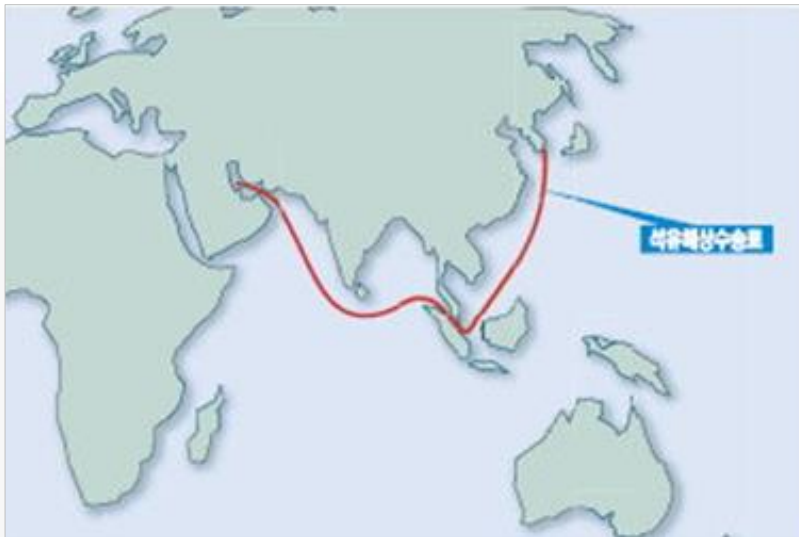
```
price = pd.read_csv('data/Data_oilstation.csv', encoding='CP949')
price = price.dropna()
pivot = price.pivot_table(index='기간', values='회발유', aggfunc='mean').reset_index()
pivot.columns = ('기간', '회발유_평균')
merged = pd.merge(price, pivot, on='기간')
merged['회발유_편차'] = list(map(gap, merged['회발유'], merged['회발유_평균']))

# high_price_rank = 높은가격순 // low_price_rank = 낮은가격순
high_price_rank = merged.pivot_table(index=['상호', '주소'], values='회발유_편차', aggfunc='mean').sort_values(by='회발유_편차',
low_price_rank = merged.pivot_table(index=['상호', '주소'], values='회발유_편차', aggfunc='mean').sort_values(by='회발유_편차')
low_price_rank = low_price_rank.reset_index()

# 파일 저장
high_price_rank.to_csv('data/high_price_rank.csv', encoding='CP949', index=None)
low_price_rank.to_csv('data/low_price_rank.csv', encoding='CP949', index=None)
```

X축(날짜) 조정

원유의 수송 : 약 6주 소요



정유사 정제 : 약 1주 소요



Data Preprocessing

1. 날짜 단위 : 주로 통일
2. 비교 단위 : 리터(l)로 통일
3. 금액 단위 : 한화(₩)로 통일
4. 정확성 : 각종 세금 포함
5. 세밀함 : 운송 및 정제 기간 감안

```

local_price = pd.read_csv('data/주유소_지역별_평균판매가격.csv', encoding='cp949') # 지역별 휘발유 평균가격
global_price = pd.read_csv('data/국제_원유가격20100102_20210901.csv', encoding='cp949') # 국제원유가격

global_price['기간'] = global_price['기간'].str.replace('년', '-').str.replace('월', '-').str.replace('일', '')
global_price['기간'] = global_price['기간'].apply(lambda x : "20"+x)
global_price['날짜'] = pd.to_datetime(global_price['기간'])
global_price = global_price.replace('-', np.nan)
global_price['Dubai'] = global_price['Dubai'].astype('float')
global_price['Brent'] = global_price['Brent'].astype('float')
global_price['WTI'] = global_price['WTI'].astype('float')

local_price['날짜'] = local_price['구분'].str.replace('년', '-').str.replace('월', '-').str.replace('일', '')
local_price['날짜'] = pd.to_datetime(local_price['날짜'])
local_price = local_price.replace('-', np.nan)
for col in local_price.columns[1:-1]:
    local_price[col] = local_price[col].astype('float')

local_price = pd.merge(local_price, global_price, how='left').iloc[:, 1:-4]
global_price = pd.merge(local_price, global_price, how='left').iloc[:, -5:].drop(columns='기간')

# merge 이후 발생한 결측치 처리
global_price['Dubai'][0] = global_price['Dubai'].mean()
global_price['Brent'][0] = global_price['Brent'].mean()
global_price['WTI'][0] = global_price['WTI'].mean()
global_price = global_price.fillna(method='ffill')

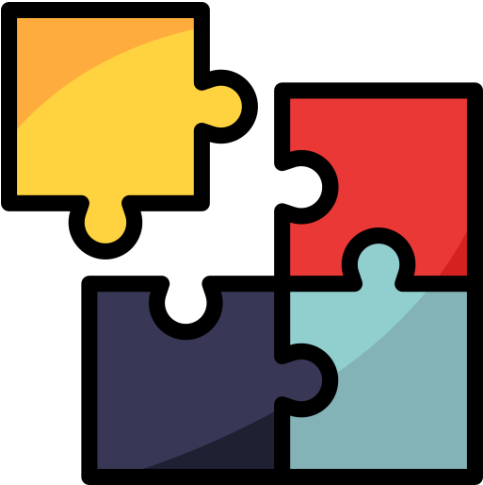
local_price['평균'] = local_price.mean(axis=1)
global_price['평균'] = global_price.mean(axis=1)

price1 = local_price.pivot_table(index='날짜')['평균']
price2 = global_price.pivot_table(index='날짜')['평균']

# 파일 저장
price1.to_csv('data/local_price.csv')
price2.to_csv('data/global_price.csv')

```

Preprocessed Data



	날짜	주유소	보통휘발유	정유사	보통휘발유	원유	평균	환율	환율반영	원유
0	2010-01-06		1649.43		1597.16		81.64	1144.3		587.921032
1	2010-01-13		1666.68		1583.76		78.55	1124.2		555.732599
2	2010-01-20		1670.36		1556.60		75.54	1124.3		534.484720

	날짜	Dubai	Brent	WTI	평균	평균-30
0	2010-01-03	78.27	80.12	81.51	79.97	NaN
1	2010-01-04	79.83	80.59	81.77	80.73	NaN
2	2010-01-05	79.92	81.89	83.18	81.66	NaN

	지역	주유소 수	위도	경도
0	강원 강릉시	85	37.752531	128.875952
1	강원 고성군	18	38.380800	128.467800
2	강원 동해시	34	37.524514	129.114630

	날짜	고급휘발유	보통휘발유	자동차용경유	실내등유	보휘-30
0	2010-01-01	1848.43	1641.12	1433.12	1020.11	NaN
1	2010-01-02	1853.39	1641.03	1433.11	1021.32	NaN
2	2010-01-03	1850.71	1641.28	1432.97	1020.85	NaN

	상호	주소	휘발유_편차
0	뉴서울(강남)	서울 강남구 언주로 716	592.571216
1	서남주유소	서울 중구 통일로 30	551.899559
2	지에스칼텍스(주)직영 역전점	서울 중구 퇴계로 15	538.125669

index	번호	지역	상호	주소	기간	상표	셀프여부	고급휘발유	휘발유	경유	실내등유	
0	1	A0011352	강원 강릉시	(주)대성길	강원 강릉시 구정면 칠성로 187	2021년 08월	S-OIL	셀프	0.0	1632.19	1438.71	913.87
1	2	A0010562	강원 강릉시	(주)동해에너지주유소	강원 강릉시 경강로 2101	2021년 08월	SK에너지	셀프	0.0	1615.00	1415.81	898.06
2	3	A0032684	강원 강릉시	(주)명진에너지 사천지점	강원 강릉시 사천면 동해대로 3576	2021년 08월	SK에너지	셀프	0.0	1630.00	1430.00	0.00

데이터 분석

국내 유가
국제 유가
관계성 확인

🗄️ 종류 : 산점도 그래프

🗄️ 대상 : 전국 주유소의 분포도

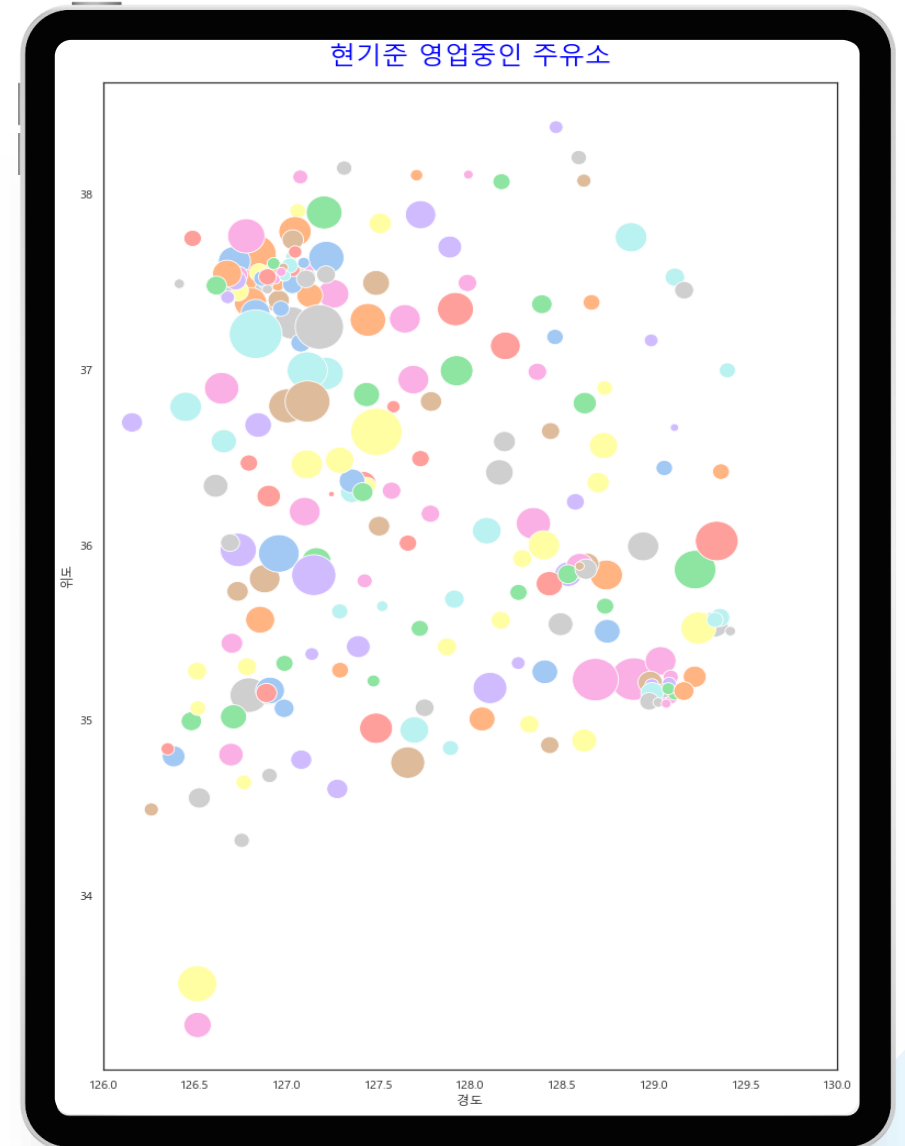
```
temp = pd.read_csv('data/202108_area.csv')
dl = temp

sns.set(style='white')
plt.rc("font", family="Malgun Gothic")
plt.figure(figsize = (12,18))

maxsize_p = max(dl['주유소 수'] * 10)
maxsize_k = max(dl['주유소 수'] * 2)

gl = sns.scatterplot(data=dl, x='경도', y='위도', size='주유소 수', hue='주유소 수', sizes=(0, maxsize_p),
                    palette='pastel')

gl.set_title('현기준 영업중인 주유소', fontdict={'fontsize': 24, 'color':'blue'}, pad=17)
gl.get_legend().remove()
gl.set_xlim(left=126, right=130)
```



🗄️ 종류 : 막대 그래프

🗄️ 대상 : 주유소 유종별 평균가

```
station_df = pd.read_csv('data/station_df.csv')
st_temp = station_df.copy()
st_temp['년'] = pd.Series('int')
st_temp['월'] = pd.Series('int')
for i, y in enumerate(st_temp['날짜']):
    st_temp['년'][i] = st_temp['날짜'][i].split('-')[0]
    st_temp['월'][i] = st_temp['날짜'][i].split('-')[1]
st_temp.pivot_table(index='년').drop(columns=['보회-30']).plot(kind='bar', rot='60', figsize=(12,8))
```



🗄️ 종류 : 히트맵 그래프

🗄️ 대상 : 주유소 유종별 평균가

```
station_df = pd.read_csv('data/station_df.csv')
st_temp = station_df.copy()
st_temp['년'] = pd.Series('int')
st_temp['월'] = pd.Series('int')
for i, y in enumerate(st_temp['날짜']):
    st_temp['년'][i] = st_temp['날짜'][i].split('-')[0]
    st_temp['월'][i] = st_temp['날짜'][i].split('-')[1]
plt.figure(figsize=(16,8))
temp = st_temp.pivot_table(index='년').drop(columns=['보회-30']).T
sns.heatmap(temp, annot=True, fmt='.2f', cmap='Blues', annot_kws={'size':12})
```



🗄️ 종류 : 라인 그래프

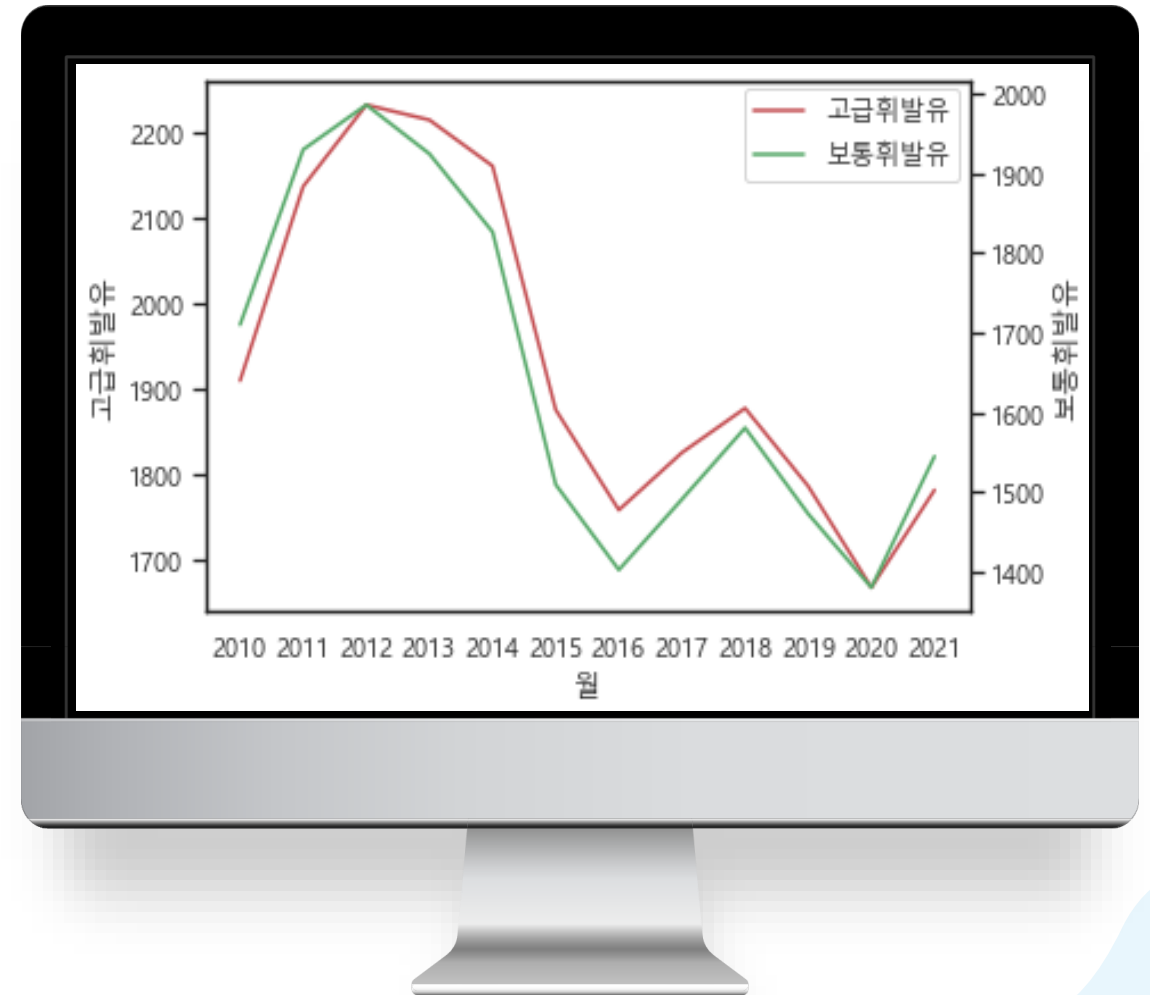
🗄️ 대상 : 연도별 휘발유 평균가

```
# 보통 & 고급 (트윈)
station_df = pd.read_csv('data/station_df.csv')
st_temp = station_df.copy()
st_temp['년'] = pd.Series('int')
st_temp['월'] = pd.Series('int')
for i, y in enumerate(st_temp['날짜']):
    st_temp['년'][i] = st_temp['날짜'][i].split('-')[0]
    st_temp['월'][i] = st_temp['날짜'][i].split('-')[1]
price1 = st_temp.pivot_table(index='월', values='고급휘발유')
price2 = st_temp.pivot_table(index='월', values='보통휘발유')

fig, ax1 = plt.subplots()
ax1.plot(price1, 'r', label='고급휘발유')
ax1.set_xlabel('월')
ax1.set_ylabel('고급휘발유')

ax2 = ax1.twinx()
ax2.plot(price2, 'g', label='보통휘발유')
ax2.set_ylabel('보통휘발유')

fig.legend(bbox_to_anchor=(0.35, .87), loc=1, borderaxespad=0.)
plt.savefig('temp2')
```



종류 : 라인 그래프

대상 : 주유소휘발유 가격

```
def detail(a=201002, b=202108, figsize=(20,10), name=None):
    price1 = station_correct_date[['날짜', '주유소 보통휘발유']]
    price2 = station_correct_date[['날짜', '원유반입 원유']]

    N = str(a)[-4] + '-' + str(a)[-4:]
    M = str(b)[-4] + '-' + str(b)[-4:]

    A = pd.Series(price1[['날짜']], str.contains(M).idxmax())
    B = pd.Series(price2[['날짜']], str.contains(N).idxmax())

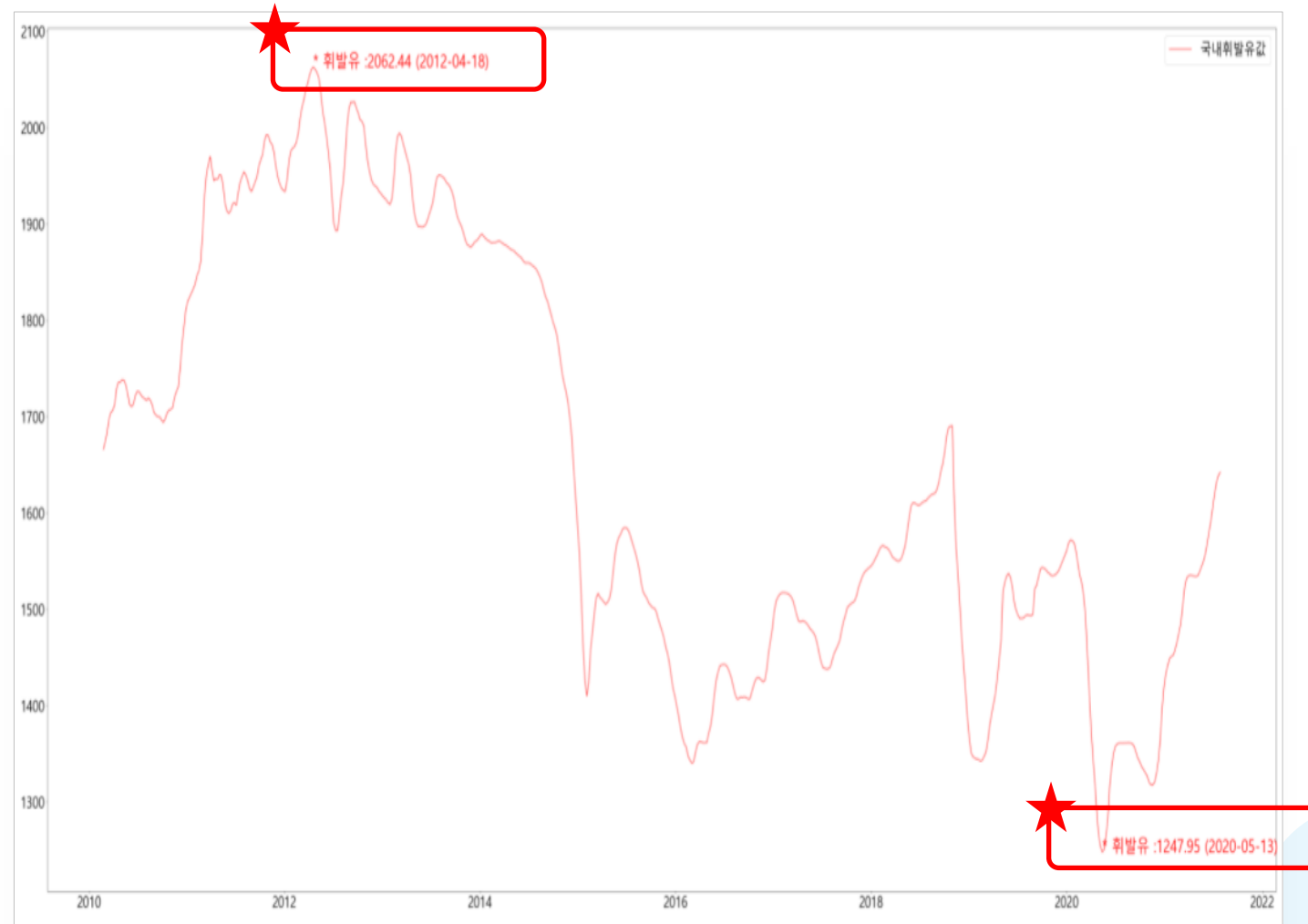
    price1[['날짜']] = pd.to_datetime(price1[['날짜']])
    price1.index = price1[['날짜']]
    price1 = price1[['주유소 보통휘발유']]
    price2[['날짜']] = pd.to_datetime(price2[['날짜']])
    price2.index = price2[['날짜']]
    price2 = price2[['원유반입 원유']]

    plt.rc('font', size=30) # 기본 폰트 크기
    plt.rc('axes', labelsize=30) # x, y축 label 폰트 크기
    plt.rc('xtick', labelsize=30) # x축 눈금 폰트 크기
    plt.rc('ytick', labelsize=30) # y축 눈금 폰트 크기
    plt.rc('legend', fontsize=30) # 범례 폰트 크기

    plt.figure(figsize=figsize)
    plt.plot(price1[A:B], 'r', label='국내휘발유값')
    plt.legend()
    plt.text(x=price1[A:B].idxmax(), y=price1[A:B].max(), s=f'휘발유 : {str(round(price1[A:B].max(),2))} ({str(price1[A:B].idxmax())})')
    plt.text(x=price1[A:B].idxmin(), y=price1[A:B].min(), s=f'휘발유 : {str(round(price1[A:B].min(),2))} ({str(price1[A:B].idxmin())})')

    if name:
        plt.savefig(name+'.png')

    print('최대')
    print('원유 : {str(round(price2[A:B].max(),2))} ({str(price2[A:B].idxmax())})')
    print('원유 : {str(round(price2[A:B].min(),2))} ({str(price2[A:B].idxmin())})')
    print('최소')
    print('원유 : {str(round(price2[A:B].max(),2))} ({str(price2[A:B].idxmax())})')
    print('원유 : {str(round(price2[A:B].min(),2))} ({str(price2[A:B].idxmin())})')
```



종류 : 워드 클라우드

대상 : 주유소휘발유 **최저가** 시점의 기사

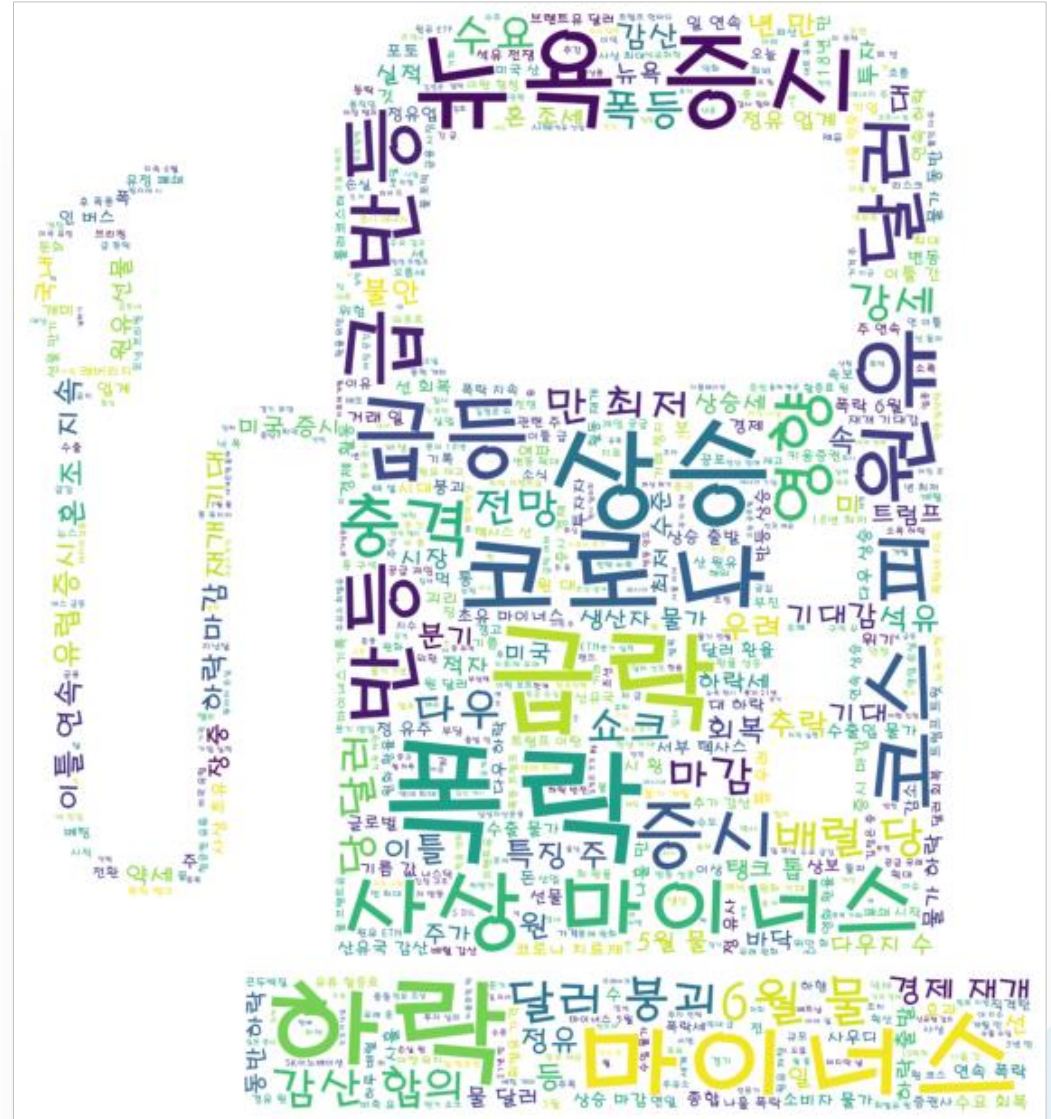
```
with open('data/2020.05.13 - 2020.05.27.txt', 'r', encoding='utf-8') as txtfile:
    data = txtfile.readlines()

from wordcloud import WordCloud
import matplotlib.pyplot as plt
from konlpy.tag import Komoran
komoran = Komoran(max_heap_size=2024)
from konlpy.corpus import kolaw
from wordcloud import STOPWORDS
불용어 = STOPWORDS | set(['국제', '유가', '2012년 4월', '4월', '3월', '2012년'])

word_list = komoran.nouns('%r' % data)
text = ' '.join(word_list)

from PIL import Image
import numpy as np
img = Image.open('img/1111.png').convert('RGBA') # RGB와 투명도
mask = np.array(img)
wordcloud = WordCloud(background_color='white',
                      max_words=600,
                      font_path='c:/Windows/Fonts/H2POPM.TTF',
                      relative_scaling=0.3,
                      mask=mask,
                      stopwords=불용어)

wordcloud.generate(text)
plt.figure(figsize=(10,20))
plt.imshow(wordcloud)
plt.axis('off')
plt.savefig('data/worldcloud_2020.05.13 - 2020.05.27.png')
plt.show()
```



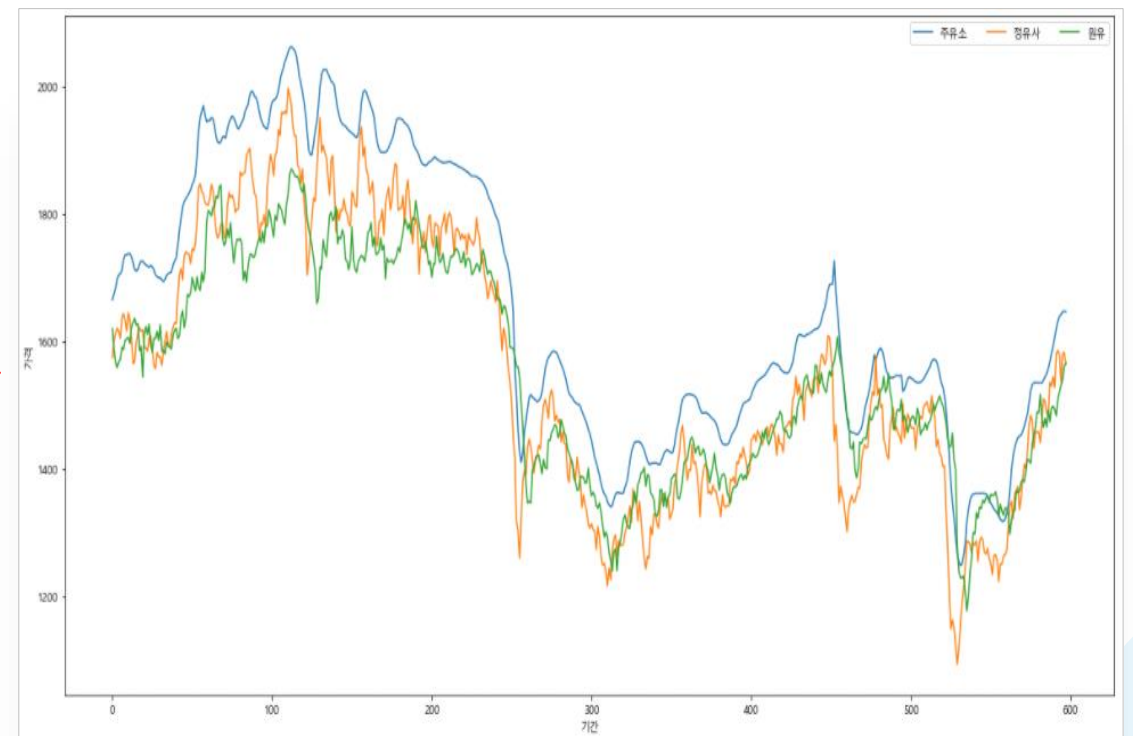
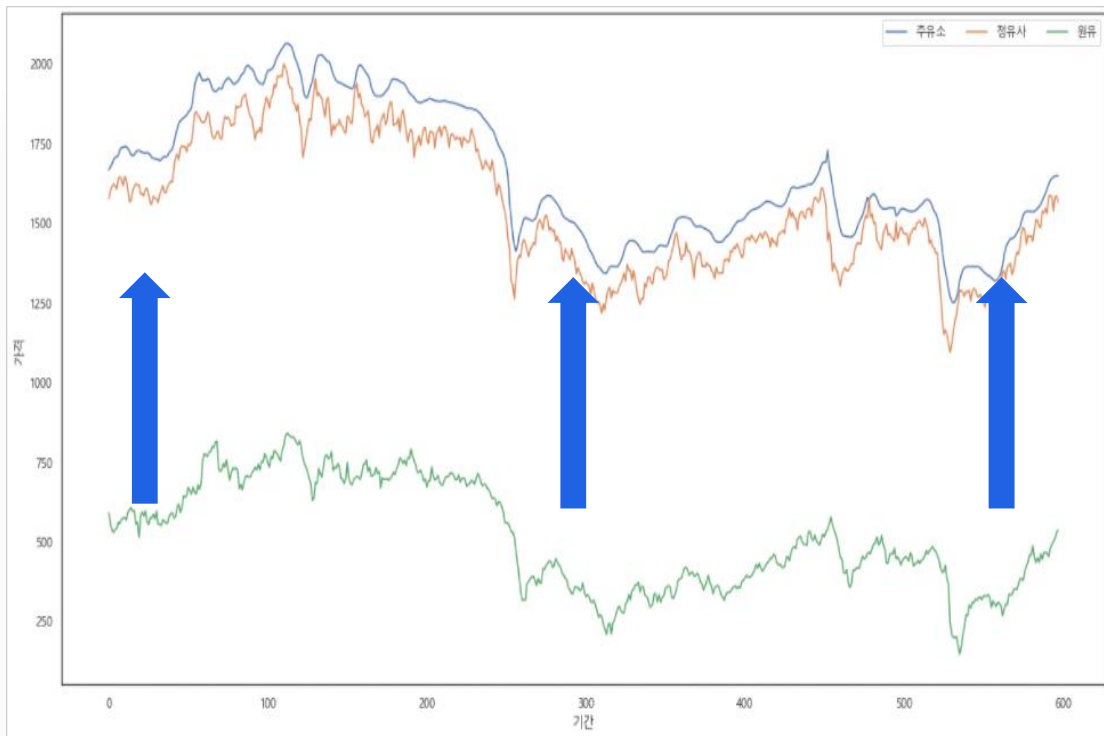
🗄️ 종류 : 라인 그래프

🗄️ 대상 : 연도별 국제원유 평균가

```
# 연도별 (라인)  
crude_df = pd.read_csv('data/crude_df.csv')  
c_temp = crude_df.copy()  
c_temp['년'] = pd.Series('int')  
c_temp['월'] = pd.Series('int')  
for i, y in enumerate(c_temp['날짜']):  
    c_temp['년'][i] = c_temp['날짜'][i].split('-')[0]  
    c_temp['월'][i] = c_temp['날짜'][i].split('-')[1]  
c_temp.pivot_table(index='년').drop(columns=['평균', '평균-30']).plot(kind='line', figsize=(12,8))  
plt.savefig('temp1')
```



🛢️ 관계 확인 : 휘발유 가격의 흐름 (주유소, 정유사, 원유)



결론

결론

연구의 한계점

🛢️ 결론 : 국내 주유소 요금 ↗ 국제 유가와 비례



연구의 한계점

- 🛢 산업에 대한 이해 부족 : 주유소의 판매가격을 결정하는 요소는 매우 다양
- 🛢 기름값 기준에 대한 오해 : 원유 시장 가격이 아닌 국제석유제품 시장 가격



Thanks !