WASSUP AI 모델 개발 8기

공공데이터(정형) 활용 AI 모델링 프로젝트

EST

서울시 알뜰 주유소 가격 예측 모델

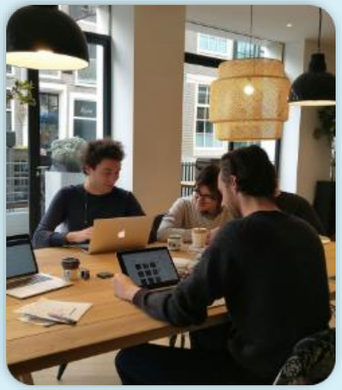
7조 김은서 이동인 이종화 최에셀

AI MODELING PROJECT

CONTENTS

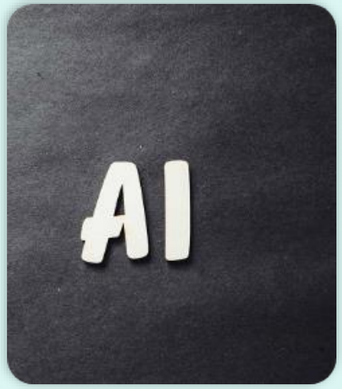
서울시 알뜰 주유소 가격 예측 모델

Business Understanding



Data Understanding

Data Preparation



Modeling

Evaluation

Business Understanding

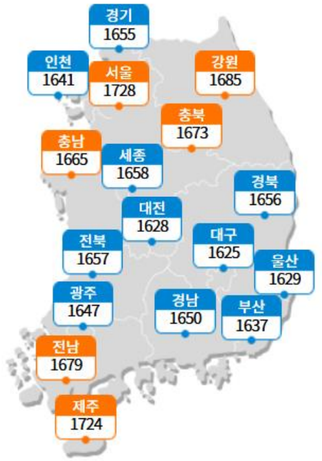
프로젝트 배경

서울시 주유소 가격은 제주도를 제외하고 **전국에서 가장 높음**

2011년 부터 시행된 **알뜰주유소 제도**는 정책적으로 안정화

그러나 운영주체의 주유가격 30~40원 인하 노력과 달리,

알뜰주유소의 가격이 **일반 주유소를 앞지르는 문제**가 지속적으로 발생



Business Understanding

프로젝트 배경

서울시 주유소 가격은 제주도를 제외하고 전국에서 가장 높음

2011년 부터 시행된 알뜰주유소 제도는 정책적으로 안정화

그러나 운영주체의 주유가격 30~40원 인하 노력과 달리,

알뜰주유소의 가격이 일반 주유소를 앞지르는 문제가 지속적으로 발생

프로젝트 목표

머신러닝을 통해

알뜰 주유소의 휘발유 적정가격을 도출

기존 주유소와의 가격 차별성을 통해

주유요금 안정화에 기여

Business Understanding

Project Plan

Step 1	Step 2	Step 3	Step 4
데이터 이해	**데이터 전처리**	**모델링**	**모델 평가**
데이터 수집 데이터 탐색 데이터 시각화 이상치 탐지	특성 선택 데이터 정제 데이터 변환 훈련/테스트 데이터 분할	모델 선택 모델 훈련 하이퍼파라미터 튜닝	성능 지표 결과 해석 최종 평가
3/31 ~ 4/7	4/2 ~ 4/7	4/8 ~ 4/9	4/9 ~ 4/10

Business Understanding

WBS (Work Breakdown Structure)

■ TODO LIST (ver 2.0) ver. 2025-04-09

번호	항목	담당자	중요도	비고
1	데이터 탐색	이종화	***	
2	주제 도출	최에셀	*****	
3	추가 목표탐색	김은서	**	
4	전처리 및 모델링	김은서	****	
5	코딩	모든 조원	*****	
6	시각화	최에셀	***	
7	상관계수 분석	이동인	**	
8	이상치 탐지&하이퍼파라미터 분석	이종화	****	
9	주제 선정발표 및 분담 브리핑	이종화	**	
10	중간 발표 프리젠테이션	김은서	***	
11	최종 시연 및 평가	이종화	*****	
12	주제(데이터) 선정발표	이종화	**	
13	주제(데이터) 선정명세	이동인	***	
14	프리젠테이션 작성	이동인	*****	
15	검증	최에셀	***	
16	적용모델 및 하이퍼파라미터 기록	이동인	*****	
17	일반화(escalation)	김은서	****	
18	분석 결과 요약	김은서	*****	

■ COMPLETE (ver 2.0) ver. 2025-04-09

번호	항목	담당자	중요도	비고
1	데이터 탐색	이종화	***	
2	주제 도출	최에셀	*****	
3	추가 목표탐색	김은서	**	
4	전처리 및 모델링	김은서	****	
5	코딩	모든 조원	*****	
6	시각화	최에셀	***	
7	상관계수 분석	이동인	**	
8	이상치 탐지&하이퍼파라미터 분석	이종화	****	
9	주제 선정발표 및 분담 브리핑	이종화	**	
10	중간 발표 프리젠테이션	김은서	***	
12	주제(데이터) 선정발표	이종화	**	
13	주제(데이터) 선정명세	이동인	***	
14	프리젠테이션 작성	이동인	*****	
15	검증	최에셀	***	
16	적용모델 및 하이퍼파라미터 기록	이동인	*****	
17	일반화(escalation)	김은서	****	
18	분석 결과 요약	김은서	*****	

번호	항목	담당자	시작일	종료일	3.31	4.1	4.2	4.3	4.4	4.7	4.8	4.9	4.10
	DataMining + Coding												
1	데이터 탐색	이종화	2025-03-31	2025-04-01									
2	주제 도출	최에셀	2025-03-31	2025-04-01									
3	추가 목표탐색	김은서	2025-03-31	2025-04-04									
4	전처리 및 모델링	김은서	2025-04-02	2025-04-08									
5	코딩	모든 조원	2025-03-31	2025-04-09									
6	시각화	최에셀	2025-04-07	2025-04-09									
7	상관계수 분석	이동인	2025-04-07	2025-04-08									
8	이상치 탐지&하이퍼파라미터 분석	이종화	2025-04-08	2025-04-09									
	Presentation												
9	주제 선정발표 및 분담 브리핑	이종화	2025-04-02	2025-04-02									
10	중간 발표 프리젠테이션	김은서	2025-04-08	2025-04-08									
12	최종 시연 및 평가	이종화	2025-04-10	2025-04-10									
	Documentation												
13	주제 선정 발표 자료 작성	이동인	2025-04-01	2025-04-01									
14	프레젠테이션 작성	이동인	2025-04-02	2025-04-09									
15	검증	최에셀	2025-04-07	2025-04-09									
16	적용모델 및 하이퍼파라미터 기록	이동인	2025-04-08	2025-04-09									
17	일반화(escalation)	김은서	2025-04-09	2025-04-09									
18	분석 결과 요약	김은서	2025-04-09	2025-04-09									

Data Understanding

데이터 수집

- 데이터 출처 식별
- 수집 방법 설명
- 데이터 수집 도구

데이터 탐색

- 기본 통계치 분석
- 분포 분석
- 결측치 분석

데이터 시각화

- 시각화 도구 선택
- 시각화 기법
- 패턴 및 특성 파악

이상치 탐지

- 이상치 정의
- 이상치 탐지 기법
- 이상치 처리

Data Understanding

데이터 수집

유가 데이터

출처: 오피넷 (한국석유공사)

경로: 국내유가통계 → 유가 내려받기

기간: 2024년 1월 ~ 12월

주요 내용:

주유소별 유가, 날짜별 유가 변동 정보,
환율, 국제유가 등

유가정보서비스 Opinet 주유소 가격비교 사이트

공시지가 데이터

출처: 서울열린데이터광장

경로: 서울시 개별공시지가 정보

기간: 2024년 1월 ~ 12월

주요 내용:

지역별 개별공시지가,
날짜별 지가 변동 정보 등

서울 열린데이터 광장

Data Understanding

데이터 탐색 - 데이터 특성 파악

유가 데이터

columns=[번호, 지역, 상호, 주소, 기간, 상표, 셀프여부, 고급휘발유, 휘발유, 경유, 실내등유]

	번호	지역	상호	주소	기간	상표	셀프여부	고급휘발유	휘발유	경유	실내등유
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	20241001.0	SK에너지	셀프	1967.0	1797.0	1667.0	0.0
2	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	20241002.0	SK에너지	셀프	1967.0	1797.0	1667.0	0.0
3	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	20241003.0	SK에너지	셀프	1967.0	1797.0	1667.0	0.0
4	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	20241004.0	SK에너지	셀프	1967.0	1797.0	1667.0	0.0

공시지가 데이터

columns=[시도명, 시군구명, 법정동명, 토지코드, 공시지가(원/㎡), 시군구코드, 법정동코드, 필지구분코드, 필지구분명, 본번, 부번, 기준년도, 기준년월]

	시도명	시군구명	법정동명	토지코드	공시지가(원/㎡)	시군구코드	법정동코드	필지구분코드	필지구분명	본번	부번	기준년도	기준년월
0	서울특별시	성북구	하월곡동	1129013600100901610	3112000	11290	13600	1	토지	90	1610	2024	2024-01-01
1	서울특별시	성북구	하월곡동	1129013600100901609	3112000	11290	13600	1	토지	90	1609	2024	2024-01-01
2	서울특별시	성북구	하월곡동	1129013600100901608	3112000	11290	13600	1	토지	90	1608	2024	2024-01-01
3	서울특별시	성북구	하월곡동	1129013600100901607	3112000	11290	13600	1	토지	90	1607	2024	2024-01-01
4	서울특별시	성북구	하월곡동	1129013600100901606	3143000	11290	13600	1	토지	90	1606	2024	2024-01-01

Data Understanding

데이터 탐색 - 데이터 특성 파악

원/달러 환율, 국제유가 데이터

columns=[날짜, 환율], [날짜, 두바이유]

	날짜	환율
0	2024-01-01	1293.530029
1	2024-01-02	1293.540039
2	2024-01-03	1307.619995
3	2024-01-04	1309.530029
4	2024-01-05	1311.250000
5	2024-01-06	1312.190002
6	2024-01-07	1312.190002
7	2024-01-08	1313.130005
8	2024-01-09	1311.800049
9	2024-01-10	1319.790039

	날짜	두바이유
0	2024-01-01	78.100
1	2024-01-02	78.100
2	2024-01-03	75.280
3	2024-01-04	78.640
4	2024-01-05	77.780
5	2024-01-06	77.825
6	2024-01-07	77.825
7	2024-01-08	77.870
8	2024-01-09	76.840
9	2024-01-10	77.990

Data Understanding

데이터 탐색 - 유형, 결측치, 통계 파악

유가 데이터 columns=[번호, 지역, 상호, 주소, 기간, 상표, 셀프여부, 고급휘발유, 휘발유, 경유, 실내등유]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 157225 entries, 0 to 157224
Data columns (total 11 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   번호       157225 non-null  object
 1   지역       157225 non-null  object
 2   상호       157225 non-null  object
 3   주소       157225 non-null  object
 4   기간       157225 non-null  int64
 5   상표       157225 non-null  object
 6   셀프여부     157225 non-null  object
 7   고급휘발유    157225 non-null  int64
 8   휘발유      157225 non-null  int64
 9   경유       157225 non-null  int64
 10  실내등유     157225 non-null  int64
dtypes: int64(5), object(6)
memory usage: 13.2+ MB
```

	기간	고급휘발유	휘발유	경유	실내등유
count	1.572250e+05	157225.000000	157225.000000	157225.000000	157225.000000
mean	2.024066e+07	1135.516216	1715.344296	1590.435618	447.883155
std	3.457540e+02	992.966337	183.471260	190.000128	715.258207
min	2.024010e+07	0.000000	0.000000	0.000000	0.000000
25%	2.024033e+07	0.000000	1617.000000	1489.000000	0.000000
50%	2.024063e+07	1817.000000	1675.000000	1548.000000	0.000000
75%	2.024093e+07	1918.000000	1729.000000	1614.000000	1430.000000
max	2.024123e+07	3215.000000	2871.000000	2840.000000	2370.000000

휘발유의 **최소값에 0**
→ 특정 석유를 판매하지 않는 주유소

또한 주유소별로 석유 가격의 편차가 크다는 것을 확인

Data Understanding

데이터 탐색 - 유형, 결측치, 통계 파악

공시지가 데이터

columns=[시도명, 시군구명, 법정동명, 토지코드, 공시지가(원/㎡), 시군구코드, 법정동코드, 필지구분코드, 필지구분명, 본번, 부번, 기준년도, 기준년월]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 894236 entries, 0 to 894235
Data columns (total 13 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   시도명         894236 non-null  object
 1   시군구명        894236 non-null  object
 2   법정동명        894236 non-null  object
 3   토지코드        894236 non-null  object
 4   공시지가(원/㎡)  894236 non-null  int64
 5   시군구코드       894236 non-null  int64
 6   법정동코드       894236 non-null  int64
 7   필지구분코드      894236 non-null  int64
 8   필지구분명       894236 non-null  object
 9   본번          894236 non-null  object
 10  부번          894236 non-null  int64
 11  기준년도        894236 non-null  int64
 12  기준년월        894236 non-null  object
dtypes: int64(6), object(7)
memory usage: 88.7+ MB
```

	0
시도명	0
시군구명	0
법정동명	0
토지코드	0
공시지가(원/㎡)	0
시군구코드	0
법정동코드	0
필지구분코드	0
필지구분명	0
본번	0
부번	0
기준년도	0
기준년월	0

dtype: int64

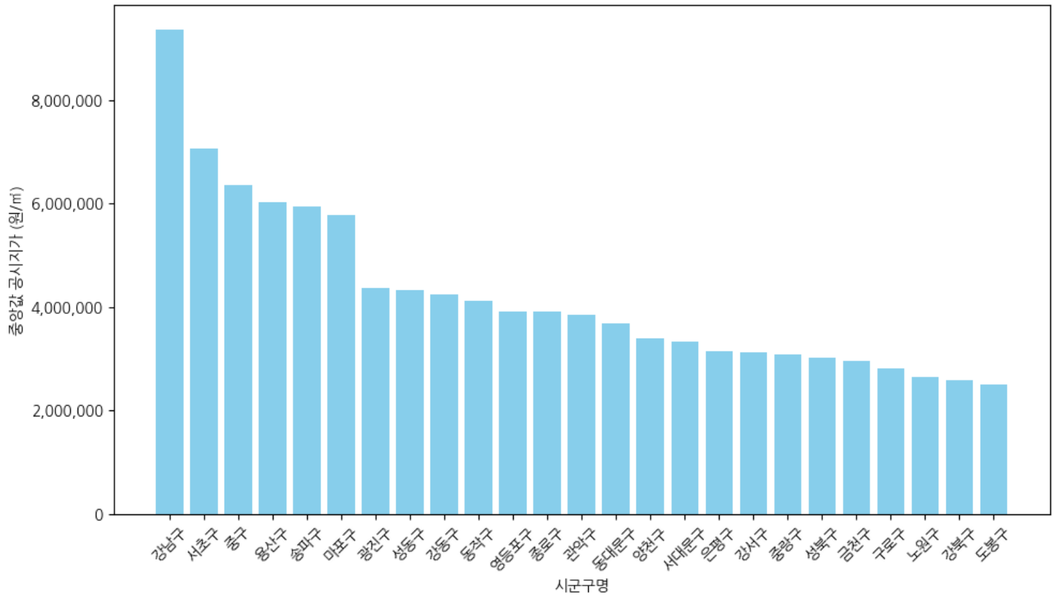
```
1 # 서울 공시지가 통계
2 land_price.describe()
```

	공시지가(원/㎡)	시군구코드	법정동코드	필지구분코드	부번	기준년도
count	8.942360e+05	894236.000000	894236.000000	894236.000000	894236.000000	894236.0
mean	4.704708e+06	11406.070786	11320.778855	1.024224	76.349238	2024.0
std	5.124744e+06	186.537230	1734.194962	0.220964	235.104717	0.0
min	4.520000e+03	11110.000000	10100.000000	1.000000	0.000000	2024.0
25%	2.376000e+06	11230.000000	10300.000000	1.000000	5.000000	2024.0
50%	3.618000e+06	11410.000000	10700.000000	1.000000	18.000000	2024.0
75%	5.452000e+06	11560.000000	11500.000000	1.000000	54.000000	2024.0
max	1.754000e+08	11740.000000	18700.000000	9.000000	4825.000000	2024.0

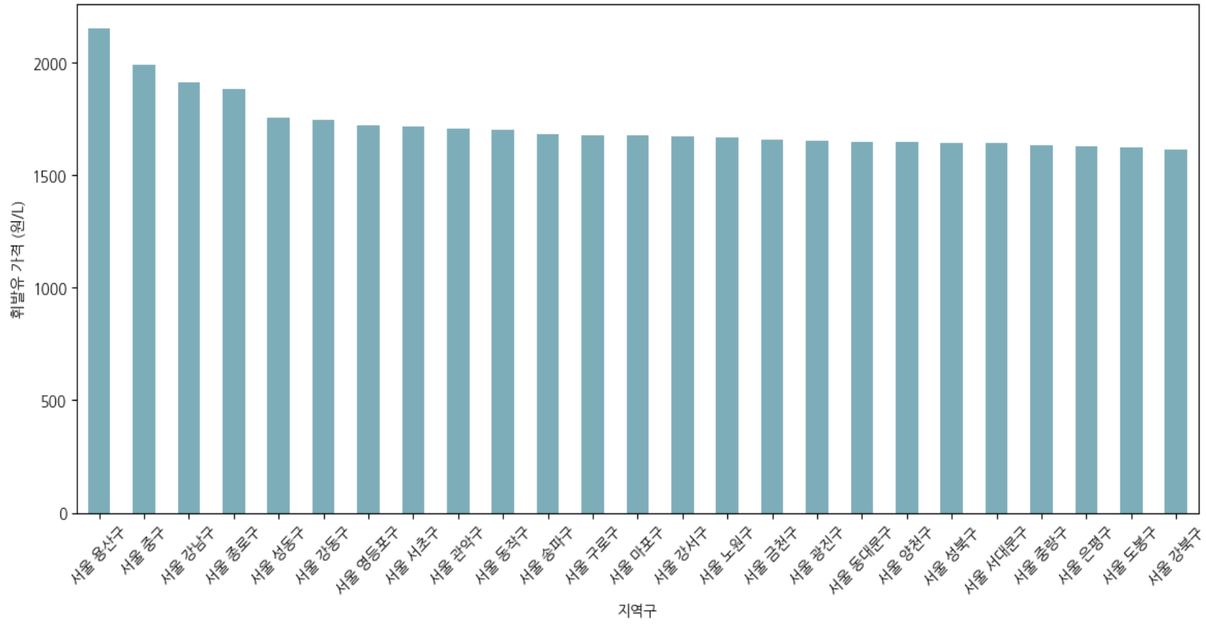
Data Understanding

데이터 시각화

공시지가 (중앙값)



휘발유가 (평균값)



Data Preparation

특성 선택

- 변수 중요도 평가
- 차원 축소

데이터 정제

- 결측치 처리
- 이상치 처리
- 데이터 형식 변환

데이터 변환

- 스케일링
- 인코딩
- 파생 변수 생성

훈련/테스트 데이터 분할

- 분할 비율 설정
- 무작위 분할

Data Preparation

데이터 병합

두바이 유가 추가

```python
# Dubai 값을 가져오기 위한 함수 정의
def get_nearest_dubai_value(date, oil_prices):
    # 기준 날짜보다 14일 전
    target_date = date - pd.Timedelta(days=14)

    # 14일 전 또는 그 이전 중 가장 가까운 날짜 찾기
    while target_date not in oil_prices['Date'].values:
        # 하루 전으로 계속 감소
        target_date -= pd.Timedelta(days=1)
        if target_date < oil_prices['Date'].min():
            # 대상 날짜가 데이터를 초과하면 None 반환
            return None

    # 해당 날짜의 Dubai 값 반환
    return oil_prices.loc[oil_prices['Date'] == target_date, 'Dubai'].values[0]

# gas_data_df의 Dubai 값을 추가
gas_data_df['Dubai'] = gas_data_df['기간'].apply(get_nearest_dubai_value, oil_prices=oil_prices_df)

```

	번호	지역	상호	주소	기간	상표	셀프여부	고급휘발유	휘발유	경유	실내등유	법정동명	공시지가(원/㎡)	Dubai
0	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	2024-01-01	SK에너지	셀프	1967	1737	1697	0	논현동	9670000.0	76.17
1	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	2024-01-02	SK에너지	셀프	1967	1737	1697	0	논현동	9670000.0	76.83
2	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	2024-01-03	SK에너지	셀프	1967	1737	1697	0	논현동	9670000.0	79.06
3	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	2024-01-04	SK에너지	셀프	1967	1737	1697	0	논현동	9670000.0	78.84
4	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	2024-01-05	SK에너지	셀프	1967	1737	1697	0	논현동	9670000.0	79.12

원/달러 환율 추가

```python
# 날짜 열 형식 변환 (문자열 -> datetime)
gas_data_copy['기간'] = pd.to_datetime(gas_data_copy['기간'])
fx_data['통계표'] = pd.to_datetime(fx_data['통계표'])

# 새로운 열 추가
def find_exchange_rate(target_date, fx_data):
    # 14일 전 날짜를 계산
    query_date = target_date - pd.Timedelta(days=14)

    # 가장 가까운 과거 날짜로 이동하여 환율 찾기
    while query_date >= fx_data['통계표'].min():
        rate_row = fx_data.loc[fx_data['통계표'] == query_date]
        if not rate_row.empty:
            return float(rate_row['계정항목'].values[0].replace(',', ''))  # 환율 값 반환
        query_date -= pd.Timedelta(days=1)
    return None  # 데이터가 없는 경우

# 환율 계산 및 열 추가
gas_data_copy['14일 전 환율'] = gas_data_copy['기간'].apply(lambda x: find_exchange_rate(x, fx_data))
```

	번호	지역	상호	주소	기간	상표	셀프여부	고급휘발유	휘발유	경유	실내등유	법정동명	공시지가(원/㎡)	Dubai	14일 전 환율
0	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	2024-01-01	SK에너지	셀프	1967	1737	1697	0	논현동	9670000.0	76.17	1294.7
1	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	2024-01-02	SK에너지	셀프	1967	1737	1697	0	논현동	9670000.0	76.83	1298.7
2	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	2024-01-03	SK에너지	셀프	1967	1737	1697	0	논현동	9670000.0	79.06	1304.7
3	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	2024-01-04	SK에너지	셀프	1967	1737	1697	0	논현동	9670000.0	78.84	1300.2
4	A0006039	서울 강남구	(유)동하석유 힐탑셀프주유소	서울 강남구 논현로 640	2024-01-05	SK에너지	셀프	1967	1737	1697	0	논현동	9670000.0	79.12	1303.8

데이터 정제

```python
# 결측치 확인
missing_data = gas_data_copy.isnull().sum()  # 각 열의 결측치 개수 계산
total_missing = gas_data_copy.isnull().sum().sum()  # 전체 결측치 개수 계산

print("각 열의 결측치 개수:₩n", missing_data)
print(f"전체 결측치 개수: {total_missing}")
```

결측치 확인

```python
# 휘발유가 0인값을 그 달의 평균값으로 채워넣기

zero_values=df['휘발유']==0 #df['휘발유'] != 0

month_avg=df[df['휘발유'] != 0].groupby('month')['휘발유'].mean() #0값을 제외한 월별 평균값

for idx in df[zero_values].index :
    month = df.loc[idx, 'month']
    df.loc[idx, '휘발유'] = month_avg[month]
```

누락값(0) 치환
: 휘발유 → 평균값(월별)

```python
# '기간' 열에서 월(month)을 추출하여 'month' 열 추가
main['month'] = pd.to_datetime(main['기간']).dt.month
```

'기간' 열에서 월(month)을 추출하여 'month' 열 추가

최종 데이터셋

Features

```
1 # 데이터 파악
2 data.head()
```

	상호	상표	셀프여부	휘발유	경유	법정동명	month	공시지가(원/㎡)	평균 공시지가(원/㎡)	Dubai 14일 전 환율	
0	(유)동하석유 힐탑셀프주유소	SK에너지	셀프	1737	1697	논현동	1	9670000.0	12473509.2	76.17	1294.7
1	(유)동하석유 힐탑셀프주유소	SK에너지	셀프	1737	1697	논현동	1	9670000.0	12473509.2	76.83	1298.7
2	(유)동하석유 힐탑셀프주유소	SK에너지	셀프	1737	1697	논현동	1	9670000.0	12473509.2	79.06	1304.7
3	(유)동하석유 힐탑셀프주유소	SK에너지	셀프	1737	1697	논현동	1	9670000.0	12473509.2	78.84	1300.2
4	(유)동하석유 힐탑셀프주유소	SK에너지	셀프	1737	1697	논현동	1	9670000.0	12473509.2	79.12	1303.8

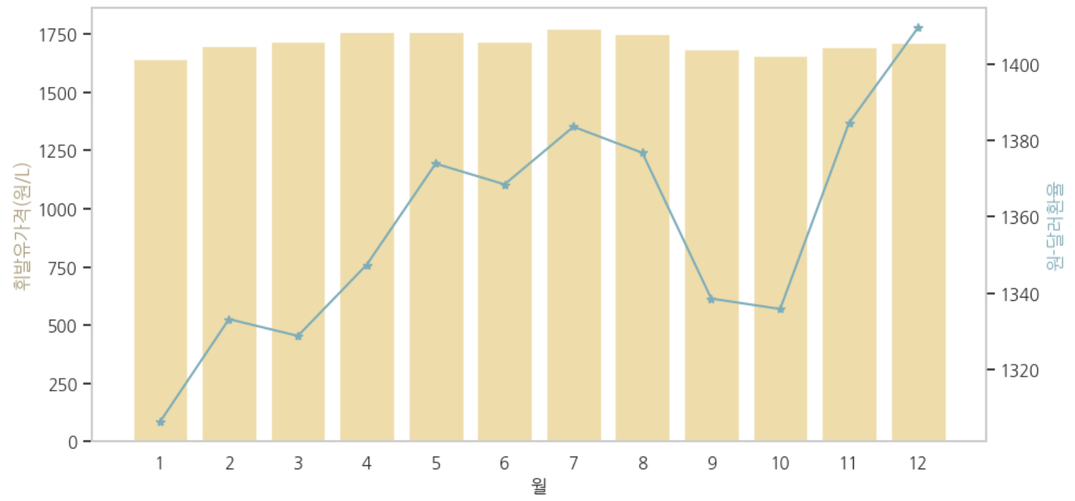
- **상표:** 브랜드(SK에너지, GS칼텍스, S-OIL, HD현대오일뱅크, 알뜰주유소, 자가상표)
- **셀프여부:** 셀프/비셀프
- **법정동명:** 지역 정보 (동 단위)
- **month:** 월 (계절성 반영 가능)
- **공시지가(원/m^2):** 법정동명 기준 중앙값
- **Dubai:** 국제 유가 (두바이유, 2주 전 데이터)
- **14일 환율:** 국제 거래와 관련된 환율 영향

Dubai, 환율 : 한국 유가에 반영되기까지 걸리는 기간이 대략 2~3주

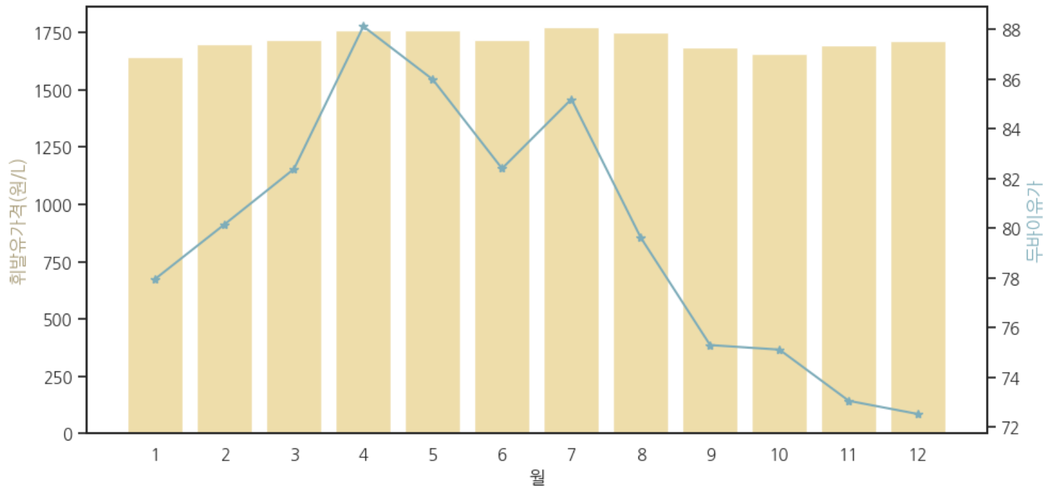
→ 기준을 14일 전으로 채택

최종 데이터셋 - 시각화

월별 휘발유 가격 및 환율



월별 휘발유 가격 및 두바이 유가

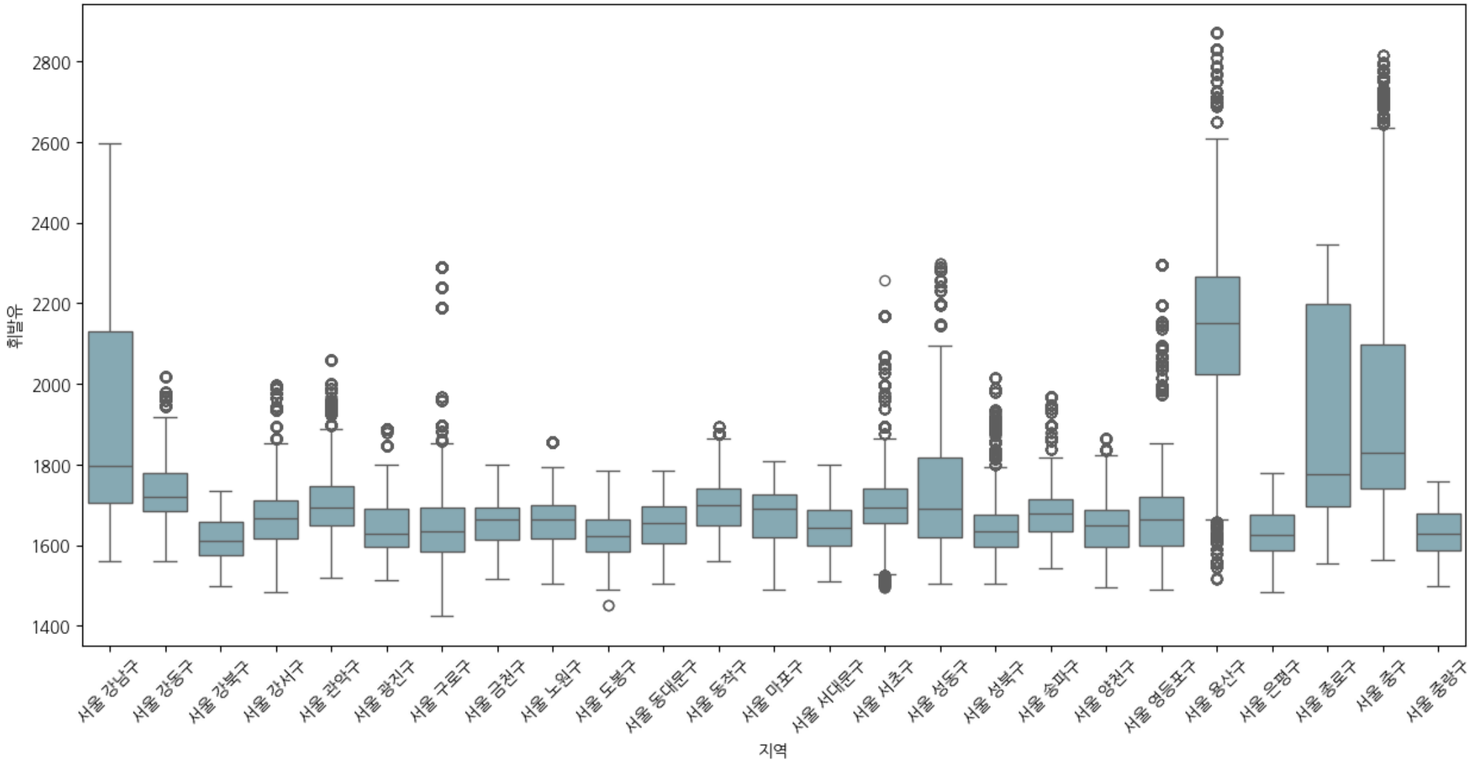


최종 데이터셋 - 상관관계 파악

상관관계 Heatmap

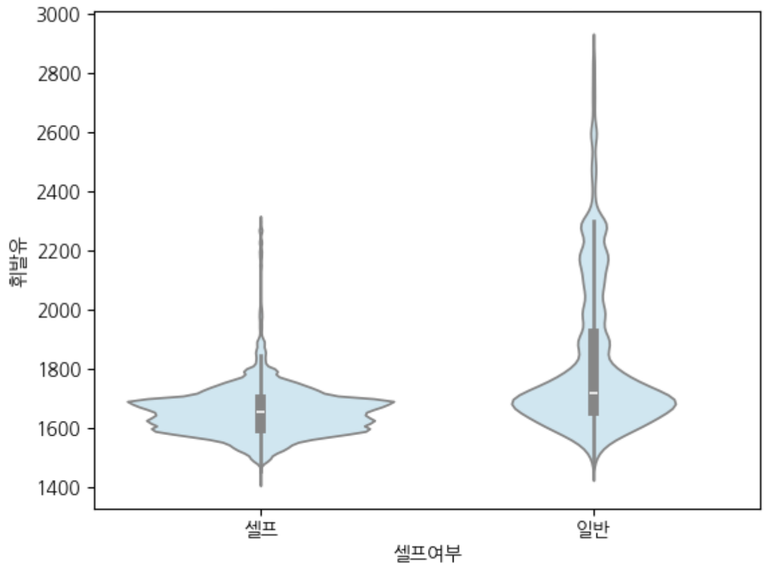
최종 데이터셋 - 시각화

지역구별 휘발유 Boxplot - 이상치 파악

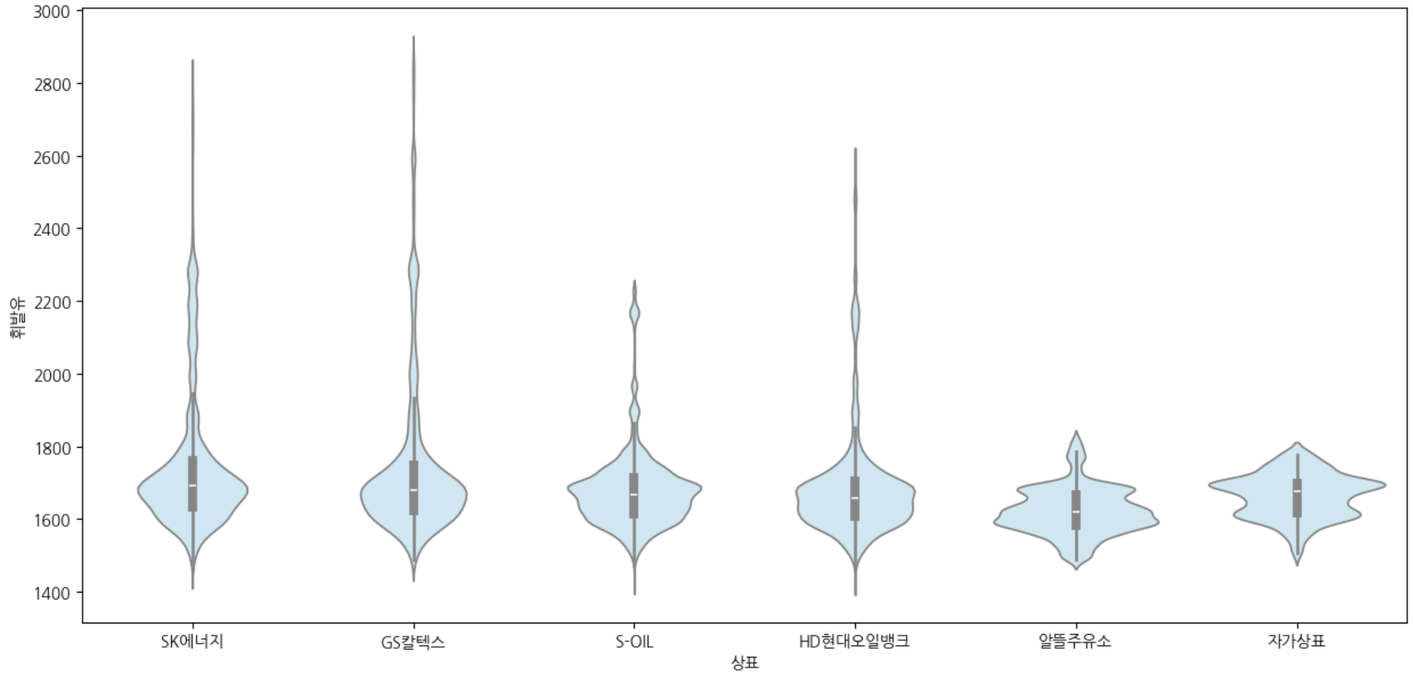


최종 데이터셋 - 시각화

셀프여부별 휘발유 가격 분포



상표별 휘발유 가격 분포



훈련/테스트 데이터 분할

```
#데이터 x, y 나누기

x=df.drop(columns=['상호','휘발유','경유','평균 공시지가(원/㎡)']) # 독립변수 설정
y=df['휘발유']
```

상호(주유소 명), 경유, 평균 공시지가
→ 모델링에 필요 없는 변수 drop

독립변수(X): 상표, 셀프여부, 법정동명, month, 공시지가(원/㎡), Dubai, 14일 전 환율
종속변수(y): 휘발유

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

train셋 / test셋 8:2 분할

데이터 변환 - 인코딩 및 스케일링

인코딩 (범주형 변수 → 수치형)

```
#독립변수의 법주형 변수 원핫인코딩
x=pd.get_dummies(x, columns=['상표','셀프여부', '법정동명' ,'month'],drop_first=True)
```

스케일링
(기존의 수치형 데이터만 선택 = num_col 리스트 생성)

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
num_col=['공시지가(원/㎡)','Dubai', '14일 전 환율']
x_train[num_col]=scaler.fit_transform(x_train[num_col])
x_test[num_col] = scaler.transform(x_test[num_col])

#확인
print(x_train[num_col].head())
```

```
        공시지가(원/㎡)     Dubai  14일 전 환율
36611     1.283893  1.220348 -0.352830
85328    -0.262033 -1.072210  2.470971
53580    -0.281077  0.807800  0.689248
37934    -0.547702  0.323670 -0.773606
107682    3.675234 -0.049318  0.889774
```

Modeling

모델 선택

- 문제 유형 파악
- 모델 후보 선정

모델 훈련

- 훈련 데이터 준비
- 모델 훈련

하이퍼파라미터 튜닝

- 튜닝 기법 선택
- 과적합 확인
- 성능 확인

Modeling

모델 선택

Linear Regression

입력 변수들과 출력 변수 사이의
선형적인 관계를 가정하고 예측하는 모델

모델 선택 이유

- 휘발유 가격은 경제지표(공시지가, 환율, 국제유가)와
  선형적인 관계가 있을 가능성이 높음 → 이를 정량적으로 분석 가능
- 각 변수의 계수를 통해
  “어떤 변수가 가격에 얼마나 영향을 미치는지” 해석할 수 있음
- 빠르고 계산이 간단해 Baseline 모델로 적합

Random Forest Regressor

여러 개의 결정 트리(Decision Tree)를
랜덤하게 학습시키고, 예측 값을 평균 내는 앙상블 모델

모델 선택 이유

- 휘발유 가격은 단순히 환율이나 유가 하나로 결정되지 않고,
  복합적인 변수 조합(ex. 상표 + 지역 + 환율)이 영향을 줌
  → 복잡한 패턴 포착
- ‘셀프 여부’, ‘상표’, ‘법정동명’ 등 범주형 변수 처리에 강점
- 이상치나 결측치에 강점

Modeling

모델 선택

K-NN (K-Nearest Neighbors)

새로운 데이터를 예측할 때,
가장 가까운 K개의 데이터를 참고하여 평균을 내어 예측하는 모델

모델 선택 이유

- 휘발유 가격은 유사한 조건(지역, 셀프 여부, 유가, 환율 등)에서 비슷한 값을 가질 가능성이 큼 → K-NN이 적합
- 모델이 단순하고 직관적이며, 학습 없이 바로 예측 가능 (Lazy Learning)
- 복잡한 수식 없이 “비슷한 상황에선 가격이 얼마였는가?”라는 아이디어가 잘 통함

XGBoost (Extreme Gradient Boosting)

Gradient Boosting 방식의 고성능 앙상블 회귀 모델
트리를 순차적으로 학습시키며,
이전 모델의 오차를 보완해 나가는 방식의 부스팅 알고리즘

모델 선택 이유

- 휘발유 가격 예측 정확도를 최대한 끌어올리고 싶을 때 최적의 선택
- ‘공시지가’, ‘환율’, ‘두바이 유가’, ‘셀프 여부’ 등 다양한 변수 간의 복잡한 비선형 관계를 정밀하게 반영 가능
- 결측값 자동 처리 기능과 정규화로 과적합 방지
  → 데이터 품질이 완벽하지 않아도 안정적인 성능
- 변수 중요도 시각화 가능
  → 어떤 요소가 가격에 가장 큰 영향을 주는지 분석 가능

Modeling

모델 훈련

Model 1

선형 회귀

```
from sklearn.linear_model import LinearRegression

linear_regression=LinearRegression(fit_intercept=False)
linear_regression.fit(x_train,y_train)

```

LinearRegression

LinearRegression(fit_intercept=False)

Model 2

랜덤 포레스트

```
from sklearn.ensemble import RandomForestRegressor
random_forest=RandomForestRegressor(
    n_estimators=200,
    random_state=42,
    max_depth=20,
    min_samples_split=4,
    min_samples_leaf=2,
    )
```

```
random_forest.fit(x_train, y_train)
```

RandomForestRegressor

RandomForestRegressor(max_depth=20, min_samples_leaf=2, min_samples_split=4, n_estimators=200, random_state=42)

Model 3

KNN 근접회귀

```
# 1. KNN 모델 초기화
from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor(
    n_neighbors=3,          # 고려할 이웃의 개수
    weights='uniform',      # 이웃 간 동일한 가중치
    metric='minkowski',     # 거리 측정 기준 (디폴트: 유클리드 거리)
    p=1
)
```

```
# 2. 모델 학습
knn.fit(x_train, y_train)
```

KNeighborsRegressor

KNeighborsRegressor(n_neighbors=3, p=1)

Model 4

XG부스트

```
# 1. XGBoost 모델 초기화
from xgboost import XGBRegressor

xgboost = XGBRegressor(
    n_estimators=200,       # 생성할 트리의 개수
    max_depth=10,           # 트리의 최대 깊이
    learning_rate=0.01,      # 학습률
    random_state=42         # 재현성을 위한 난수 고정
)
```

```
# 2. 모델 학습
xgboost.fit(x_train, y_train)
```

XGBRegressor

XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=0.01, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=10, max_leaves=None, min_child_weight=None, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=200, n_jobs=None, num_parallel_tree=None, random_state=42, ...)

하이퍼파라미터 튜닝

Linear Regression

- Hyperparameter: Fit_intercept

선형 회귀 모델에서 절편(intercept)을 포함할지 여부를 결정

- TRUE → 절편을 포함하여 모델이 데이터를 학습. 즉, 직선이 원점을 통과할 필요가 없음
- FALSE → 절편을 포함하지 않고 학습하며 직선이 원점을 지나도록 제한

Modeling

하이퍼파라미터 튜닝

Random Forest Regressor

하이퍼파라미터 1	하이퍼파라미터 2	하이퍼파라미터 3	하이퍼파라미터 4	하이퍼파라미터 5
n_estimators	max_depth	min_samples_split	min_samples_leaf	random_state
100	10	2	1	42
100	10	2	2	42
100	10	4	1	42
100	10	4	2	42
100	20	2	1	42
100	20	2	2	42
100	20	4	1	42
100	20	4	2	42
200	10	2	1	42
200	10	2	2	42
200	10	4	1	42
200	10	4	2	42
200	20	2	1	42
200	20	2	2	42
200	20	4	1	42

Hyperparameter

- n_estimators: 생성할 결정 트리의 개수
  → 값이 클수록 모델의 안정성과 성능이 향상, but 학습 시간도 증가
- max_depth: 각 결정 트리의 최대 깊이를 정의
  → 과적합을 방지하기 위해 적절히 설정
- min_samples_split: 내부 노드를 분할하기 위해 필요한 최소 샘플 수 지정
  → 값이 높으면 모델의 복잡도가 낮아져, 과적합 방지
- min_samples_leaf: 리프 노드에 있어야 할 최소 샘플 수 정의
  → 값이 높으면 모델이 단순해짐
- random_state: 난수 생성기의 시드를 설정하여 결과 재현

하이퍼파라미터 튜닝

K-NN (K-Nearest Neighbors)

하이퍼파라미터 1	하이퍼파라미터 2	하이퍼파라미터 3
n_neighbors	weights	metric
3	uniform	manhattan
5	distance	euclidean
10	uniform	manhattan
3	uniform	manhattan
5	uniform	euclidean
10	distance	manhattan

Hyperparameter

- n_neighbors: 이웃으로 고려할 데이터의 개수, 즉 K의 값
  → K값이 작을수록 모델이 데이터에 민감(복잡), 클수록 부드러운(단순한) 결정
- weights: 각 이웃의 중요도를 설정
  - uniform: 모든 이웃에 동일한 가중치를 부여
  - distance: 가까운 이웃에게 더 큰 가중치를 부여
- metric: 거리 계산 방식
  - 맨해튼 거리(Manhattan Distance, 절대 거리): 축을 따라 이동하는 거리
  - 유클리드 거리(Euclidean Distance, 직선 거리): 두 점 간의 직선 거리

Modeling

하이퍼파라미터 튜닝

XGBoost (Extreme Gradient Boosting)

하이퍼파라미터 1	하이퍼파라미터 2	하이퍼파라미터 3
n_estimators	max_depth	learning_rate
100	10	0.1
100	10	0.01
100	20	0.1
100	20	0.01
200	10	0.1
200	10	0.01
200	20	0.1
200	20	0.01
300	10	0.1
300	10	0.01
300	20	0.1
300	20	0.01
300	30	0.1
400	10	0.1
400	10	0.01
400	20	0.1
400	20	0.01
400	30	0.1
400	30	0.01
500	10	0.1
600	10	0.1
700	10	0.1
800	10	0.1

Hyperparameter

- n_estimators: 생성할 부스팅 트리의 개수
  값이 클수록 모델이 복잡, 과적합(overfitting) 위험 증가
- max_depth: 각 결정 트리의 최대 깊이
  값이 클수록 복잡한 패턴 학습, but 과적합 위험
- learning_rate: 학습 속도 제어 매개변수
  낮은 값(예: 0.01~0.1) 사용 → 학습이 느리지만 더 정밀한 결과

Evaluation

성능 지표 선택

- 평균 절대 오차
- 평균 제곱 오차
- 결정계수

결과 해석

- 모델 결과 분석
- 비즈니스 관점에서 해석

최종 평가

- 인사이트
- 시행착오
- 참고문헌

Evaluation

성능 지표 선택

성능 지표	간단한 설명	선택 이유
MAE (Mean Absolute Error) 평균 절대 오차	예측값과 실제값 사이의 절댓값 차이 평균 ex) 평균적으로 몇 원 정도 차이 나는지 보여줌	- 결과 해석이 직관적 (ex : 평균 42원 차이) - 이상치에 덜 민감하여 현실적인 성능 판단 가능
MSE (Mean Squared Error) 평균 제곱 오차	예측 오차를 제곱해서 평균한 값 → 큰 오차에 큰 패널티 부여	- 큰 오차가 중요한 경우에 유리 - 손실 함수로 자주 사용되며 모델 미세 조정에 민감
R^2 (결정계수)	전체 변동성 중 모델이 설명한 비율 값이 1에 가까울수록 좋은 성능	- 모델의 설명력 확인 가능 → 예측이 단순 평균보다 얼마나 나은지 설명

Evaluation

모델 결과 해석

Linear Regression

하이퍼파라미터	평가지표				
fit_intercept	MAE	MSE	R^2	mse-train_mse	train_R^2-R^2
TRUE	50.96	7413.73	0.768	-16.21	0.005
FALSE	50.96	7413.73	0.768	-16.21	0.005

K-NN (K-Nearest Neighbors)

하이퍼파라미터 1	하이퍼파라미터 2	하이퍼파라미터 3	평가지표				
n_neighbors	weights	metric	MAE	MSE	R^2	mse-train_mse	train_R^2-R^2
3	uniform	manhattan	19.51990771	2812.179844	0.9118569291	-1210.589242	0.03909490075
5	distance	euclidean	23.46879928	4923.283363	0.8456879223	-3723.909081	0.1175816462
10	uniform	manhattan	20.01493471	2140.525161	0.9329088211	-355.2897158	0.01241894757
3	uniform	manhattan	19.22346921	2758.348728	0.9135441753	-1154.331187	0.03733333031
5	uniform	euclidean	18.95435179	2403.071108	0.9246797577	-748.9041838	0.02466194
10	distance	manhattan	24.29461287	4900.115728	0.846414073	-3700.74331	0.1168555526

Evaluation

모델 결과 해석 - Random Forest

하이퍼파라미터 1	하이퍼파라미터 2	하이퍼파라미터 3	하이퍼파라미터 4	하이퍼파라미터 5	평가지표				
n_estimators	max_depth	min_samples_split	min_samples_leaf	random_state	MAE	MSE	R^2	mse-train_mse	train_R^2-R^2
100	10	2	1	42	53.18	7129.72	0.777	244.78	0.013
100	10	2	2	42	53.16	7112.49	0.777	216.08	0.012
100	10	4	1	42	53.16	7117.65	0.777	229.5	0.012
100	10	4	2	42	53.16	7112.49	0.777	216.08	0.012
100	20	2	1	42	31.06	3952.73	0.876	1423.8	0.046
100	20	2	2	42	30.75	3681.35	0.885	1089.74	0.036
100	20	4	1	42	30.89	3827.35	0.88	1273.8	0.042
100	20	4	2	42	30.75	3681.35	0.885	1089.74	0.036
200	10	2	1	42	53.19	7126.04	0.777	236.03	0.012
200	10	2	2	42	53.28	7112.54	0.777	211	0.012
200	10	4	1	42	53.17	7114.2	0.777	220	0.012
200	10	4	2	42	53.28	7112.54	0.777	211	0.012
200	20	2	1	42	31.04	3948.92	0.876	1413.29	0.046
200	20	2	2	42	30.75	3682.96	0.885	1084.34	0.036
200	20	4	1	42	30.88	3822.69	0.88	1261.94	0.041

Evaluation

모델 결과 해석 - XGBoost

하이퍼파라미터 1	하이퍼파라미터 2	하이퍼파라미터 3	평가지표				
n_estimators	max_depth	learning_rate	MAE	MSE	R^2	mse-train_mse	train_R^2-R^2
100	10	0.1	30.55	3003.15	0.906	730.63	0.025
100	10	0.01	65.57	9489.88	0.703	164.77	0.012
100	20	0.1	21.56	3912.26	0.877	2614.22	0.083
100	20	0.01	65.57	9489.88	0.703	164.77	0.012
200	10	0.1	25.6	2724.89	0.915	847.79	0.028
200	10	0.01	50.25	5630.62	0.824	512.66	0.02
200	20	0.1	21.22	4077.51	0.872	2823.96	0.089
200	20	0.01	32.21	3617.86	0.887	1265.37	0.041
300	10	0.1	22.91	2621.62	0.918	924.85	0.03
300	10	0.01	41.62	4145.36	0.87	599.24	0.021
300	20	0.1	21.45	4272.59	0.866	3043.57	0.096
300	20	0.01	25.31	3393.06	0.894	1790.75	0.057
300	30	0.1	23	4913.9	0.846	3714.28	0.117
400	10	0.1	21.81	2617.29	0.918	998.84	0.032
400	10	0.01	37.74	3656.67	0.885	654.57	0.023
400	20	0.1	21.79	4469.93	0.86	3254.84	0.103
400	20	0.01	23.01	3579.43	0.888	2164.96	0.069
400	30	0.1	23.08	4932.56	0.845	3733.14	0.118
400	30	0.01	22.5	4029.52	0.874	2760.69	0.087
500	10	0.1	20.87	2650.13	0.917	1119.93	0.036
600	10	0.1	20.51	2747.94	0.914	1269.22	0.041
700	10	0.1	20.28	2866.91	0.91	1433.3	0.046
800	10	0.1	20.28	3013.34	0.906	1614.94	0.052

Evaluation

최종 모델 - review

Random Forest

```python
input_template = pd.DataFrame(columns=x_train.columns)

#예측용 입력값을 템플릿 복사
one_input = input_template.copy()
one_input.loc[0] = 0  # 모든 값 0으로 초기화

# 필요한 컬럼만 수동으로 설정
one_input.loc[0, '공시지가(원/㎡)'] = 3717000.0
one_input.loc[0, 'Dubai'] = 74.69
one_input.loc[0, '14일 전 환율'] = 1463.50
one_input.loc[0, '상표_알뜰주유소'] = 1
one_input.loc[0, '셀프여부_일반'] = 0
one_input.loc[0, '법정동명_신림동'] = 1
one_input.loc[0, 'month_4'] = 1

# 수치형 스케일링
num_cols = ['공시지가(원/㎡)', 'Dubai', '14일 전 환율']
one_input[num_cols] = scaler.transform(one_input[num_cols])

# 예측
pred = random_forest.predict(one_input)
print(f'예측 결과: {pred[0]:.2f} 원/L')
```

```
예측 결과: 1655.56 원/L
<ipython-input-246-42802f5dd1a8>:9: FutureWarning: Setting an item of incom
  one_input.loc[0, 'Dubai'] = 74.69
<ipython-input-246-42802f5dd1a8>:10: FutureWarning: Setting an item of inco
  one_input.loc[0, '14일 전 환율'] = 1463.50
```

XGBoost

```python
one_input_xgboost = input_template.copy()
one_input_xgboost.loc[0] = 0  # 모든 값 0으로 초기화

# 필요한 컬럼만 수동으로 설정
one_input_xgboost.loc[0, '공시지가(원/㎡)'] = 3717000.0
one_input_xgboost.loc[0, 'Dubai'] = 74.69
one_input_xgboost.loc[0, '14일 전 환율'] = 1463.50
one_input_xgboost.loc[0, '상표_알뜰주유소'] = 1
one_input_xgboost.loc[0, '셀프여부_일반'] = 0
one_input_xgboost.loc[0, '법정동명_신림동'] = 1
one_input_xgboost.loc[0, 'month_4'] = 1

# 수치형 스케일링
num_cols = ['공시지가(원/㎡)', 'Dubai', '14일 전 환율']
one_input_xgboost[num_cols] = scaler.transform(one_input_xgboost[num_cols])

# 예측
pred = xgboost.predict(one_input_xgboost)
print(f'예측 결과: {pred[0]:.2f} 원/L')
```

```
예측 결과: 1668.51 원/L
<ipython-input-47-7147bf2ee0bb>:6: FutureWarning: Setting an item of incompatible
  one_input_xgboost.loc[0, 'Dubai'] = 74.69
<ipython-input-47-7147bf2ee0bb>:7: FutureWarning: Setting an item of incompatible
  one_input_xgboost.loc[0, '14일 전 환율'] = 1463.50
```

Evaluation

추가 진행 모델 - Ensemble Learning

```python
one_input_avg = input_template.copy()
one_input_avg.loc[0] = 0  # 모든 값 0으로 초기화

# 필요한 컬럼만 수동으로 설정
one_input_avg.loc[0, '공시지가(원/㎡)'] =3717000.0
one_input_avg.loc[0, 'Dubai'] = 74.69
one_input_avg.loc[0, '14일 전 환율'] = 1463.50
one_input_avg.loc[0, '상표_알뜰주유소'] = 1
one_input_avg.loc[0, '셀프여부_일반'] = 0
one_input_avg.loc[0, '법정동명_신림동'] = 1
one_input_avg.loc[0, 'month_4'] = 1

# 수치형 스케일링
num_cols = ['공시지가(원/㎡)', 'Dubai', '14일 전 환율']
one_input_avg[num_cols] = scaler.transform(one_input_avg[num_cols])

pred_random_forest = random_forest.predict(one_input_avg)
pred_xgboost = xgboost.predict(one_input_avg)

# 평균 앙상블
ensemble_pred = (pred_random_forest + pred_xgboost) / 2

# 결과 출력
print(f'예측 결과 (평균 앙상블): {ensemble_pred[0]:.2f} 원/L')
```

```
예측 결과 (평균 앙상블): 1662.03 원/L
<ipython-input-71-eee5922c537a>:6: FutureWarning: Setting an item of incompatible dtype is de
  one_input_avg.loc[0, 'Dubai'] = 74.69
<ipython-input-71-eee5922c537a>:7: FutureWarning: Setting an item of incompatible dtype is de
  one_input_avg.loc[0, '14일 전 환율'] = 1463.50
```

Evaluation

최종 평가 - 연구의 한계 및 개선 방향

한계

- 데이터 제한
  : 연구는 서울시 데이터에 국한
  → 전국적으로 확장 가능성을 논하기 어려움

- 시간적 제약
  : 1년간의 데이터를 기반으로 예측
  → 계절적 요인 또는 다년간의 트렌드를 반영하지 못함

- 외부 요인 부족
  : 유류세 정책, 국제정치적 요인, 소비자 행동 등의 외부 변수는 포함하지 못함
  → 실제 가격에 미치는 다양한 영향을 완벽히 반영하지 못함

- 모델 한계
  : 머신러닝 모델이 데이터를 바탕으로만 예측
  → 지정학적 리스크나 정책 변화와 같은 예기치 못한 외부 요인을 처리하지 못함

개선 방향

- 데이터 확장
  : 서울 외의 다른 도시 또는 전국 데이터를 수집하여,
  지역별 가격 패턴을 반영한 보다 정교한 모델 구축

- 시계열 분석 통합
  : 시계열 기반 유가 예측 및 실시간 가격 모니터링 시스템 구축으로,
  가격 변동의 시간적 흐름을 반영하여 장기적 트렌드를 분석할 필요

- 사회·정책적 요인 통합
  : 유류세, 소비자 행동, 정부 보조금 및 경제적 불확실성과 같은
  정책적·사회적 요소를 포함하면 모델의 실효성을 높일 수 있을 것
  + 주유소 위치 기반 GIS 데이터, 소비자 수요, 교통량 등의
  외부 변수를 통합하여 보다 정교한 예측 모델을 개발

- 새로운 알고리즘 도입
  : 딥러닝 기법(LSTM, Transformer)을 활용해
  복잡한 데이터 패턴 탐구 및 데이터 내에서 예측력 강화

프로젝트를 마치며

배운 점

실무에서는 예상치 못한 변수들이 많다.

- 유가에 영향을 미치는 변수들이 생각보다 훨씬 다양
- 단순히 상관계수만으로는 변수의 중요도를 판단하기 어렵고,
  실제로 모델을 돌려보면서 결과를 확인해야만 알 수 있는 부분이 많았음
- 히트맵 상에서 상관관계가 낮다고 해서 제외할 수는 없고,
  오히려 다양한 변수들을 종합적으로 고려하는 시도가 필요함을 느낌

적합한 모델을 고르는 것은 생각보다 어렵다.

- 데이터의 특성과 목적에 따라 어떤 모델이 적합한지는 직접 실험을 해봐야 알 수 있었음
- 선형회귀처럼 단순한 모델이 오히려 잘 맞을 때도 있고, 반대로 XGBoost 같은 복잡한 모델이 더 나은 결과를 내는 경우도 있었음
- 모델 선택 시에는 예측 목적이 무엇인지, 속도와 해석 가능성 중 무엇이 중요한지 등을 사전에 명확히 해야 한다는 점을 배웠음

변수 중요도는 모델을 통해 확인해야 한다.

- 수업에서는 보통 상관계수나 EDA(탐색적 자료 분석)를 통해 변수 중요도를 추정했지만,
- 실전에서는 모델의 피처 중요도(feature importance)를 통해,
  실제 영향력을 수치화해보는 과정이 매우 중요하다는 것을 체감

수업 데이터는 '착한 데이터'였다.

- 수업에서 다뤘던 데이터는 정제도 잘 되어 있었고, 변수들도 깔끔히 구성된 '이상적인 데이터셋'
- 반면 실제 데이터를 다루면서 결측치, 이상치, 복잡한 변수 해석 등
  여러 실무적인 문제를 경험할 수 있었고, 이 과정을 통해 한층 더 성장

'왜 이 모델을 만드는가'를 먼저 생각해야 한다.

- 단순히 "정확한 예측"만을 위한 모델인지,
- 아니면 "정책 수립"이나 "의사 결정 지원"을 위한 모델인지에 따라,
  선택할 변수, 모델, 성능 지표까지 모두 달라질 수 있습니다.
- 앞으로는 모델을 만들기 전에 목적부터 명확히 정의해야 한다는 교훈을 얻음

참고 문헌

자료 출처

- 한국석유공사 오피넷 (https://www.opinet.co.kr/)
- 서울열린데이터광장 (https://data.seoul.go.kr/)
- '셀프 주유소' 서울선 145원 싼데, 충남은 20원뿐… 왜죠? (조선일보, 2024.07.16.)
- [지식innovaiton]국제유가와 동네 주유소 가격(SK이노베이션, 2015.09.04)
- 전체 주유소보다 30~40원 싸게”…알뜰주유소 “인하 노력하겠다!(산업통상자원부, 2024.04.12.)
- 이름만 알뜰주유소…“일반 주유소 보다 더 비싸”
- 〈국내 휘발유, 경유 가격변동성 분석: 설명요인과 지렛대효과(2010)〉, 김형건 외, 한국산업경제학회
- 〈세계 경제 지표를 활용한 머신 러닝 기반 국제 경유 가격 예측 모델 개발(2023)〉, 김아린 외, JCCT