발표 로딩중..

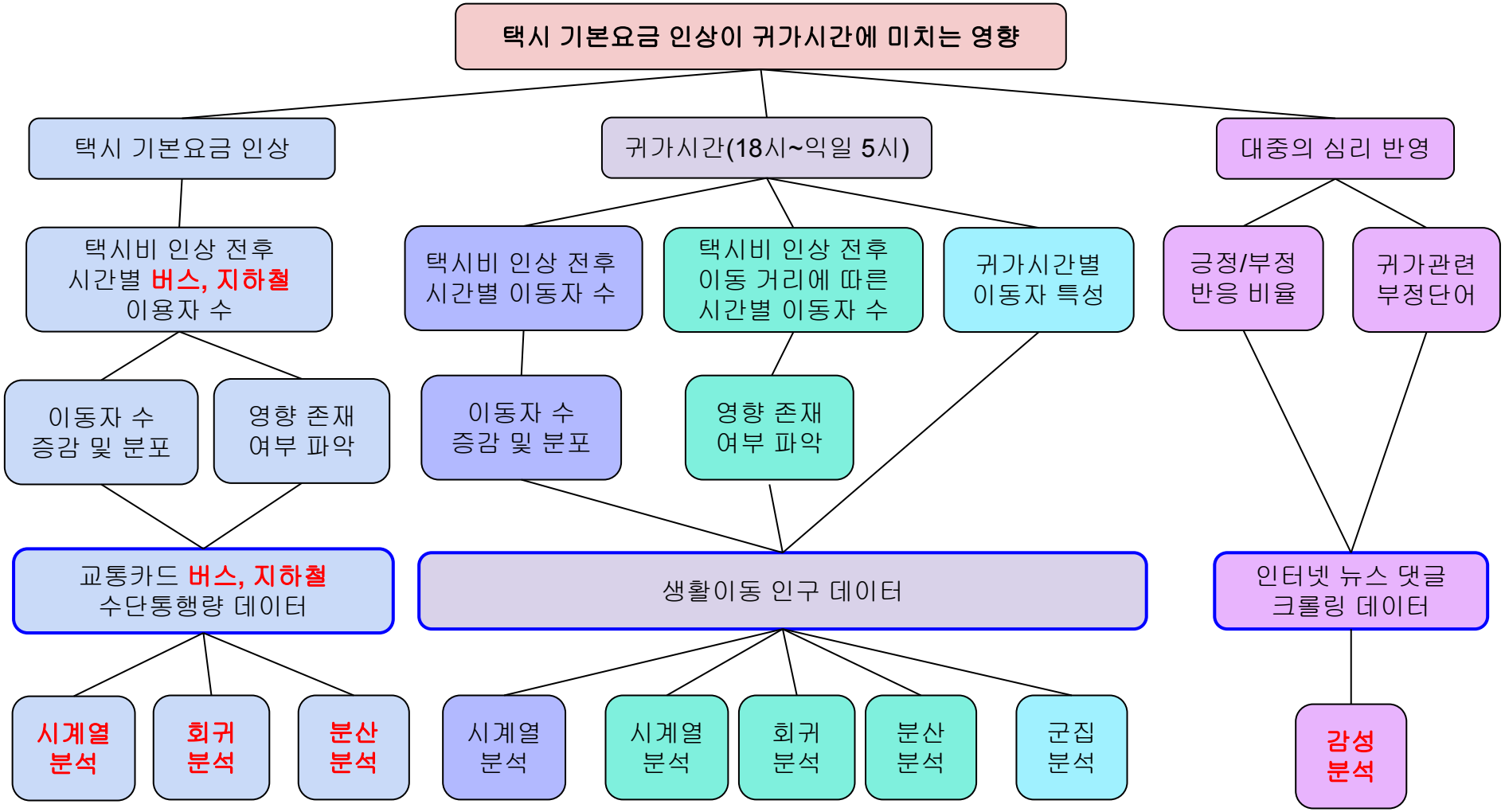
바다톤 분석 중간발표

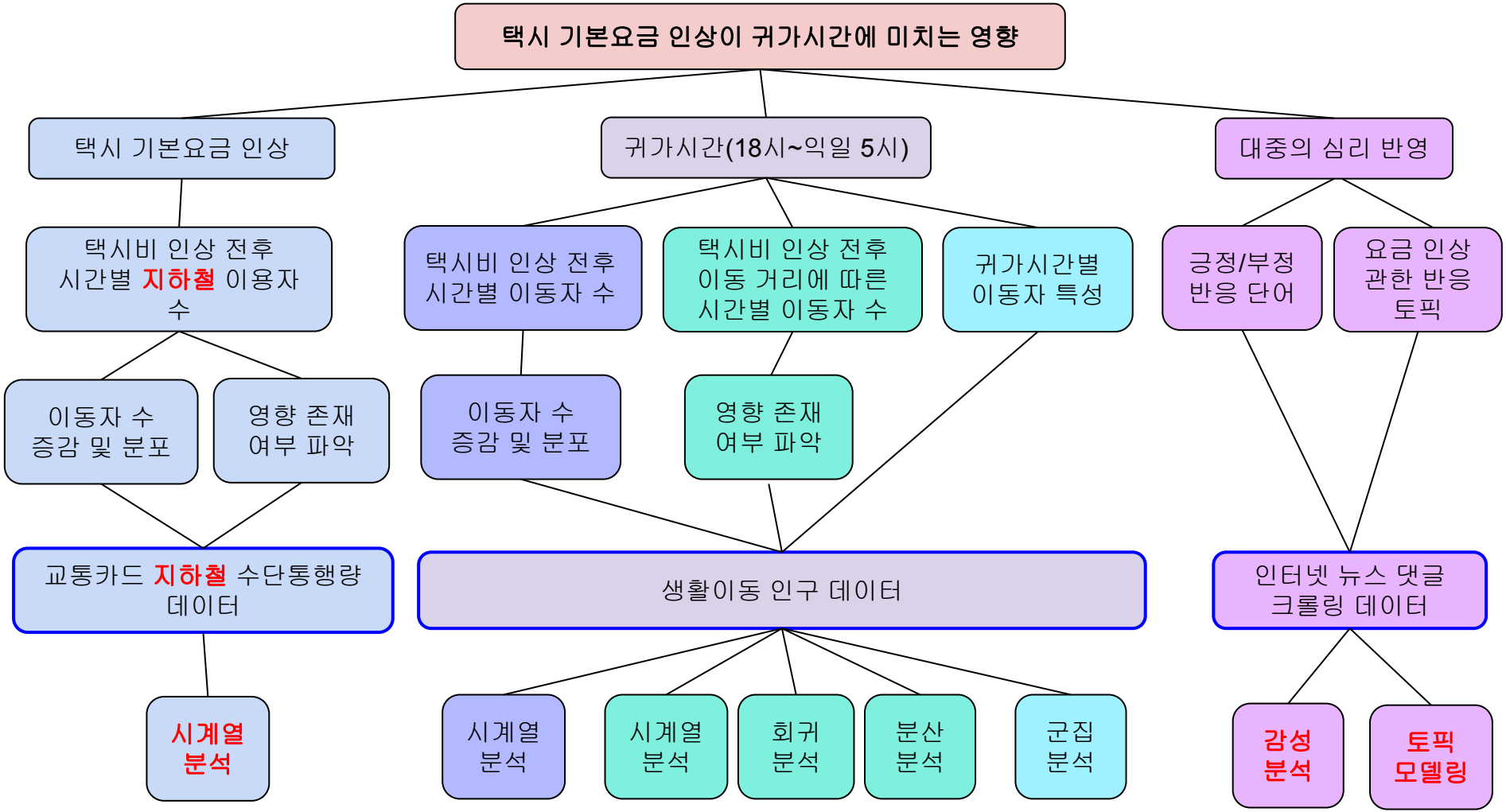
바다의 공주들

이세은 김윤아 정성희

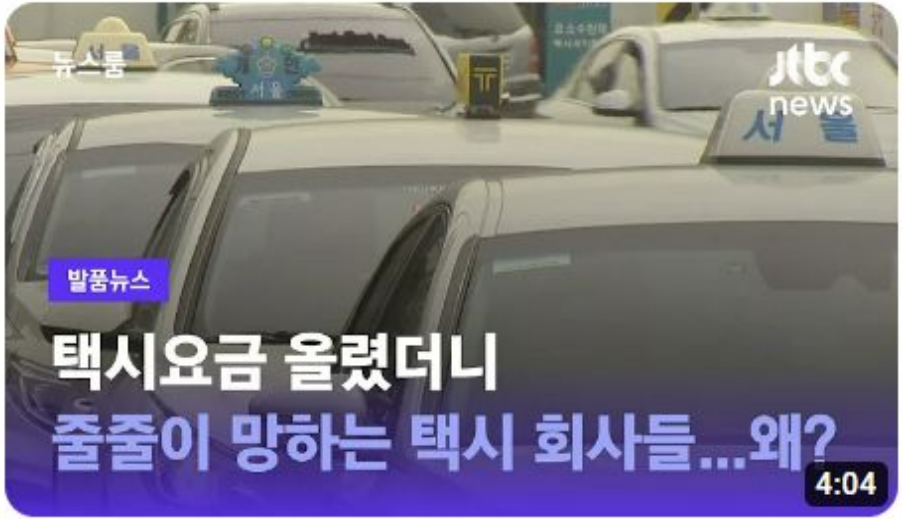
Contents

1. 분석계획 수정사항
2. 현재 진행상황
3. 향후 계획


택시 기본요금 인상이 귀가시간에 미치는 영향
택시 기본요금 인상
귀가시간(18시~익일 5시)
대중의 심리 반영
택시비 인상 전후
시간별 버스, 지하철
이용자 수
택시비 인상 전후
시간별 이동자 수
택시비 인상 전후
이동 거리에 따른
시간별 이동자 수
귀가시간별
이동자 특성
긍정/부정
반응 비율
귀가관련
부정단어
이동자 수
증감 및 분포
영향 존재
여부 파악
이동자 수
증감 및 분포
영향 존재
여부 파악
교통카드 버스, 지하철
수단통행량 데이터
생활이동 인구 데이터
인터넷 뉴스 댓글
크롤링 데이터
시계열
분석
회귀
분석
분산
분석
시계열
분석
시계열
분석
회귀
분석
분산
분석
군집
분석
감성
분석


택시 기본요금 인상이 귀가시간에 미치는 영향
택시 기본요금 인상
귀가시간(18시~익일 5시)
대중의 심리 반영
택시비 인상 전후 시간별 지하철 이용자 수
택시비 인상 전후 시간별 이동자 수
택시비 인상 전후 이동 거리에 따른 시간별 이동자 수
귀가시간별 이동자 특성
긍정/부정 반응 단어
요금 인상 관한 반응 토픽
이동자 수 증감 및 분포
영향 존재 여부 파악
이동자 수 증감 및 분포
영향 존재 여부 파악
교통카드 지하철 수단통행량 데이터
생활이동 인구 데이터
인터넷 뉴스 댓글 크롤링 데이터
시계열 분석
시계열 분석
시계열 분석
회귀 분석
분산 분석
군집 분석
감성 분석
토픽 모델링

대중의 심리 반영



조회수 **360**만회 / 댓글 **18224**개

조회수 **102**만회 / 댓글 **6071**개

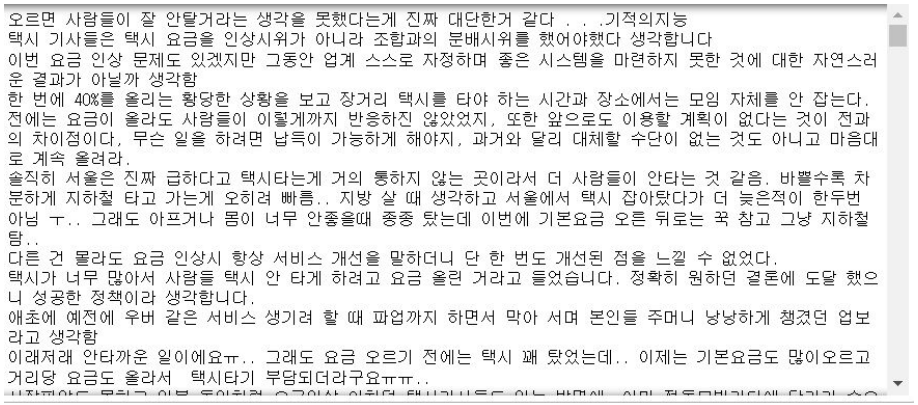
대중의 심리 반영

1. 유튜브 댓글 크롤링

```
# 유튜브 열기
Url = "https://www.youtube.com/watch?v=YiQvKeNvGXM"
driver.get(Url)
time.sleep(6)

# 유튜브 댓글 끝까지 로딩
last_page_height = driver.execute_script("return document.documentElement.scrollHeight")
while True:
    driver.execute_script("window.scrollTo(0, document.documentElement.scrollHeight);")
    time.sleep(3.0)
    new_page_height = driver.execute_script("return document.documentElement.scrollHeight")

    if new_page_height == last_page_height:
        break
    last_page_height = new_page_height

# 파싱
html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')

comments = soup.find_all("ytd-comment-thread-renderer", class_ = "style-scope ytd-item-section-renderer")
for comment in comments :
    comment_text = comment.find("yt-formatted-string", id="content-text").text
    try :
        print(comment_text)
    except :
        print("이모티콘 출력에 문제가 있음.")
```

대중의 심리 반영

1. 유튜브 댓글 크롤링


오르면 사람들이 잘 안탈거라는 생각을 못했다는게 진짜 대단한거 같다 . . .기적의지능
택시 기사들은 택시 요금을 인상시위가 아니라 조합과의 분배시위를 했어야했다 생각합니다
이번 요금 인상 문제도 있겠지만 그동안 업계 스스로 자정하며 좋은 시스템을 마련하지 못한 것에 대한 자연스러운 결과가 아닐까 생각함
한 번에 40%를 올리는 황당한 상황을 보고 장거리 택시를 타야 하는 시간과 장소에서는 모임 자체를 안 잡는다.
전에는 요금이 올라도 사람들이 이렇게까지 반응하진 않았었지, 또한 앞으로도 이용할 계획이 없다는 것이 전과의 차이점이다, 무슨 일을 하려면 납득이 가능하게 해야지, 과거와 달리 대체할 수단이 없는 것도 아니고 마음대로 계속 올려라.
솔직히 서울은 진짜 급하다고 택시타는게 거의 통하지 않는 곳이라서 더 사람들이 안타는 것 같음. 바쁠수록 차분하게 지하철 타고 가는게 오히려 빠름.. 지방 살 때 생각하고 서울에서 택시 잡아탔다가 더 늦은적이 한두번 아님 ㅜ.. 그래도 아프거나 몸이 너무 안좋을때 종종 탔는데 이번에 기본요금 오른 뒤로는 꾹 참고 그냥 지하철 탐..
다른 건 몰라도 요금 인상시 항상 서비스 개선을 말하더니 단 한 번도 개선된 점을 느낄 수 없었다.
택시가 너무 많아서 사람들 택시 안 타게 하려고 요금 올린 거라고 들었습니다. 정확히 원하던 결론에 도달 했으니 성공한 정책이라 생각합니다.
애초에 예전에 우버 같은 서비스 생기려 할 때 파업까지 하면서 막아 서며 본인들 주머니 낭낭하게 챙겼던 업보라고 생각함
이래저래 안타까운 일이에요ㅠ.. 그래도 요금 오르기 전에는 택시 꽤 탔었는데.. 이제는 기본요금도 많이오르고 거리당 요금도 올라서 택시타기 부담되더라구요ㅠㅠ..

	Comment
0	오르면 사람들이 잘 안탈거라는 생각을 못했다는게 진짜 대단한거 같다 . . .기적의지능
1	택시 기사들은 택시 요금을 인상시위가 아니라 조합과의 분배시위를 했어야했다 생각합니다
2	이번 요금 인상 문제도 있겠지만 그동안 업계 스스로 자정하며 좋은 시스템을 마련하지...
3	한 번에 40%를 올리는 황당한 상황을 보고 장거리 택시를 타야 하는 시간과 장소에...
4	솔직히 서울은 진짜 급하다고 택시타는게 거의 통하지 않는 곳이라서 더 사람들이 안타...
...	...
1335	이쪽 저쪽 눈치보지말고 우버택시 도입해라.
1336	3:06 와 주현영인줄..ㅎㅎ 발음이 진짜 ...
1337	어차피 자동운전 나오면 사장될 직업
1338	바본가?? ㅋㅋㅋㅋㅋ\n저걸 몰랐다고?? ㅋㅋㅋㅋㅋ
1339	이정부 하는일이 다그렇지 기대도 않는다

1340 rows × 1 columns

대중의 심리 반영

2. 한국어 텍스트 데이터 전처리

```
!pip install konlpy
```

```
Requirement already satisfied: konlpy in c:₩users₩82107₩anaconda3₩lib₩site-packages (0.6.0)
Requirement already satisfied: JPype1>=0.7.0 in c:₩users₩82107₩anaconda3₩lib₩site-packages (from konlpy) (1.
4.1)
Requirement already satisfied: numpy>=1.6 in c:₩users₩82107₩anaconda3₩lib₩site-packages (from konlpy) (1.23.
5)
Requirement already satisfied: lxml>=4.1.0 in c:₩users₩82107₩anaconda3₩lib₩site-packages (from konlpy) (4.9.
1)
Requirement already satisfied: packaging in c:₩users₩82107₩appdata₩roaming₩python₩python310₩site-packages (fr
om JPype1>=0.7.0->konlpy) (23.0)
```

```
# 정규 표현식 함수 정의

import re

def apply_regular_expression(Comment):
    hangul = re.compile('[^ ㄱ-ㅣ 가-힣]')  # 한글 추출 규칙: 띄어 쓰기(1 개)를 포함한 한글
    result = hangul.sub('', Comment)  # 위에 설정한 "hangul"규칙을 "text"에 적용(.sub)시킴
    return result
```

```
df['Comment'][1]
```

```
'택시 기사들은 택시 요금을 인상시위가 아니라 조합과의 분배시위를 했어야했다 생각합니다'
```

```
apply_regular_expression(df['Comment'][1])
```

```
'택시 기사들은 택시 요금을 인상시위가 아니라 조합과의 분배시위를 했어야했다 생각합니다'
```

대중의 심리 반영

2. 한국어 텍스트 데이터 전처리_ 명사단위

1) 명사 형태소 추출

```
from konlpy.tag import Okt
from collections import Counter
```

```
okt = Okt()  # 명사 형태소 추출 함수
nouns = okt.nouns(apply_regular_expression(df['Comment'][1]))
nouns
```

```
['택시', '기사', '택시', '요금', '인상', '시위', '조합', '분배', '시위', '생각']
```

```
df = pd.read_excel("taxinews2_comments.xlsx")
pd.set_option('display.max_columns', None)
df_dropna = df.dropna()
doc_origin = df_dropna['Comment'].to_list()
doc_origin
```

대중의 심리 반영

2. 한국어 텍스트 데이터 전처리_ 명사단위

2) 말뭉치 생성 -> 명사 형태소 추출

```
# 말뭉치 생성: 전체 말뭉치에 적용해서 명사 형태소 추출
corpus = "".join(map(str,df['Comment'].to_list()))
corpus
# 정규 표현식 적용
apply_regular_expression(corpus)
# 전체 말뭉치(corpus)에서 명사 형태소 추출
nouns = okt.nouns(apply_regular_expression(corpus))
print(nouns)
```

```
['사람', '생각', '진짜', '기적', '지능', '택시', '기사', '택시', '요금', '인상', '시위', '조합', '분배',
'시위', '생각', '이번', '요금', '인상', '문제', '그동안', '업계', '스스로', '자정', '시스템', '마련',
'못', '것', '대한', '결과', '생각', '함', '번', '를', '상황', '보고', '장거리', '택시', '시간', '장소',
'모임', '자체', '안', '전', '요금', '사람', '반응', '또한', '앞', '이용', '계획', '것', '전과', '차이점',
'무슨', '일', '납득', '과거', '달리', '대체', '수단', '것', '마음대로', '계속', '서울', '진짜', '택시',
'거의', '통', '곳', '더', '사람', '안타', '것', '지하철', '타고', '오히려', '지방', '살', '때', '생각',
'서울', '택시', '더', '적', '번', '몸', '때', '종종', '이번', '기본', '요금', '오른', '뒤', '꼭', '참고',
'그냥', '지하철', '다른', '건', '요금', '인상', '시', '항상', '서비스', '개선', '말', '단', '번', '개선',
'점', '수', '택시', '사람', '택시', '안', '요금', '거', '결론', '도달', '정책', '생각', '애초', '예전',
'우버', '서비스', '때', '파업', '본인', '주머니', '낭낭', '업보', '생각', '함', '일이', '요금', '전', '택
시', '꽤', '이제', '요금', '거리', '요금', '택시', '부담', '시장', '파악', '못', '일본', '독일', '요금인
상', '택시', '기사', '방면', '모빌리티', '단거리', '수요', '상황', '택시', '기사', '요금인상', '일', '한
번', '휴일', '무제', '해결', '손', '이야기', '평소', '택시', '일', '안', '이번', '요금', '오른', '이후',
'더욱더', '안', '거', '옆', '나라', '일본', '비교', '택시', '요금', '하나', '서비스', '차원', '넘사벽',
'수도권', '야간', '택시', '승차', '거부', '비일', '비재', '우버', '랩', '승차', '공유', '플랫폼', '규제',
'경쟁', '해외', '랩', '보', '말', '자기', '스스로', '구조조정', '모습', '간만', '가슴', '해지', '뉴스',
'그동안', '택시', '회사', '한국', '선진국', '중', '택시', '요금', '가장', '틈', '맹비난', '퍼', '정작',
'택시', '요금', '시민', '택시', '타지', '오히려', '택시', '회사', '도산', '요금', '수익', '것', '단세포',
'발상', '때문', '택시', '회사', '화', '자초', '것', '수요', '가격탄력성', '상품', '가격', '변화', '요량',
```

대중의 심리 반영

2. 한국어 텍스트 데이터 전처리

3) 불용어 제거

In [95]:
```
stopwords = pd.read_csv("https://raw.githubusercontent.com/yoonkt200/FastCampusDataset/master/korean_stopwords
stopwords[:500]
```

Out[95]:
```
[['휴'],
 ['아이구'],
 ['아이쿠'],
 ['아이고'],
 ['어'],
 ['나'],
 ['우리'],
 ['저희'],
 ['따라'],
 ['의해'],
 ['을'],
 ['를'],
 ['에'],
 ['의'],
 ['가'],
 ['으로'],
 ['로'],
 ['에게'],
 ['뿐이다'],
```

In [96]:
```
taxinews_stopwords = ['택시','요금','인상','택시요금']
for word in taxinews_stopwords:
    stopwords.append(word)
```

대중의 심리 반영

3. Word Count

```
df_cleaned = df.dropna(subset=['Comment'])
df['Comment'].fillna('', inplace=True)

import numpy as np
from sklearn.feature_extraction.text import CountVectorizer

def text_cleaning(text):
    hangul = re.compile('[^ ㄱ-ㅣ 가-힣]')  # 정규 표현식 처리
    result = hangul.sub('', text)
    okt = Okt()  # 형태소 추출
    nouns = okt.nouns(result)
    nouns = [x for x in nouns if len(x) > 1]  # 한글자 키워드 제거
    nouns = [x for x in nouns if x not in stopwords]  # 불용어 제거
    return nouns

vect = CountVectorizer(tokenizer = lambda x: text_cleaning(x))
bow_vect = vect.fit_transform(df['Comment'].tolist())
word_list = vect.get_feature_names_out()
count_list = np.sum(bow_vect.toarray(),axis=0)
```

```
C:₩Users₩82107₩anaconda3₩lib₩site-packages₩sklearn₩feature_extraction₩text.py:528: UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn(
```

```
# 단어 리스트
word_list
```

```
array(['가게', '가격', '가격탄력성', ..., '흐름', '흠왜', '흥정'], dtype=object)
```

대중의 심리 반영

3. Word Count

```
# 각 단어가 전체 리뷰중에 등장한 총 횟수
count_list
```

```
array([ 11, 101,   2, ...,   5,   1,   1], dtype=int64)
```

```
# 각 단어의 리뷰별 등장 횟수
bow_vect.toarray()
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```
bow_vect.shape
```

```
(1340, 2796)
```

```
# "단어" - "총 등장 횟수" Matching

word_count_dict = dict(zip(word_list, count_list))
word_count_dict
```

```
'교육': 7,
'교차로': 1,
'교체': 2,
'교촌': 1,
'교통': 29,
'교통난': 1,
'교통비': 4,
'교통상황': 2,
'교통체증': 4,
'굉윤설': 1,
'구간': 1,
'구나': 1,
'구로': 1,
'구리': 1,
'구박': 2,
'구분': 1,
'구석': 1,
```

대중의 심리 반영

4. TF-IDF 변환

```
from sklearn.feature_extraction.text import TfidfTransformer

tfidf_vectorizer = TfidfTransformer()
tf_idf_vect = tfidf_vectorizer.fit_transform(bow_vect)
```

```
print(tf_idf_vect.shape)
```

```
(1340, 2796)
```

```
# 첫 번째 리뷰에서의 단어 중요도(TF-IDF 값) -- 0이 아닌 것만 출력
print(tf_idf_vect[0])
```

```
  (0, 2300)     0.3422634775092302
  (0, 2253)     0.5629068230446256
  (0, 1252)     0.29345494687670975
  (0, 1186)     0.3012825978395228
  (0, 384)      0.6237825029012524
```

```
# 첫 번째 리뷰에서 모든 단어의 중요도 -- 0인 값까지 포함
print(tf_idf_vect[0].toarray().shape)
print(tf_idf_vect[0].toarray())
```

```
(1, 2796)
[[0. 0. 0. ... 0. 0. 0.]]
```

대중의 심리 반영

4. TF-IDF 변환

```
# "벡터" - "단어" mapping
vect.vocabulary_
```

```
{'사람': 1186,
 '생각': 1252,
 '진짜': 2300,
 '기적': 384,
 '지능': 2253,
 '기사': 372,
 '시위': 1413,
 '조합': 2173,
 '분배': 1115,
 '이번': 1839,
 '문제': 911,
 '그동안': 322,
 '업계': 1600,
 '스스로': 1374,
 '자정': 1974,
 '시스템': 1411,
 '마련': 778,
 '대한': 617,
 '결과': 144,
```

```
invert_index_vectorizer = {v: k for k, v in vect.vocabulary_.items()}
print(str(invert_index_vectorizer)[:100]+'...')
```

```
{1186: '사람', 1252: '생각', 2300: '진짜', 384: '기적', 2253: '지능', 372: '기사', 1413: '시위', 2173: '조
합', 1115:...
```

대중의 심리 반영

5. 감성 분류-Logistic Regression

1) 데이터셋 생성

```
df.sample(1340)
```

	Unnamed: 0	Comment
737	737	곧 10년 이내 완전 자율 주행 차가 돌아다닐태고 일자리는 줄어드는데 시민의 발...
624	624	2~3달전에 택시탔는데 \n택시 기사님들이 일부러 일 안하고있다고했음. \n택시수가...
457	457	정책을 펼치면 원하는대로 가는게아니라 반대로의 결과만 나오는게 걱정이다...
394	394	아주 훈훈한 뉴스네요
29	29	업계 관계자들에게는 생계가 걸렸을 수도 있는 안타까운 일이지만 충격 이후에는 택시 ...
...	...	...
751	751	우리나라는 지하철 버스 마을버스가너무잘되있음그래서 택시기본요금오르면경쟁력없어짐
1036	1036	뭐 우버들어오면 망한다, 카택도 문제다. 카풀도 문제다 등등 지들한테 조금만 불리하...
253	253	이제 개인택시들만 거의 남게되고 외국처럼 길거리에 쓸데없이 길만막히게 하는 택시들이...
748	748	진짜 누칼협밖에 할말이 없다
41	41	걸어서 십분거리가 기본요금올라서 오천원인데 예전에는 피곤하면 택시탔는데 지금은 운동...

1340 rows × 2 columns

대중의 심리 반영

5. 감성 분류-Logistic Regression

1) 데이터셋 생성

```
# 'rating' 열 추가
df['rating'] = [1,2,3,4,5] * (len(df) // 5) + [1,2,3,4,5][:len(df) % 5]
```

```
def rating_to_label(rating):
    if rating > 3:
        return 1
    else:
        return 0

df['y'] = df['rating'].apply(lambda x: rating_to_label(x))
```

```
df.head()
```

	Unnamed: 0	Comment	rating	y
0	0	오르면 사람들이 잘 안탈거라는 생각을 못했다는게 진짜 대단한거 같다 . . .기적의지능	1	0
1	1	택시 기사들은 택시 요금을 인상시위가 아니라 조합과의 분배시위를 했어야했다 생각합니다	2	0
2	2	이번 요금 인상 문제도 있겠지만 그동안 업계 스스로 자정하며 좋은 시스템을 마련하지...	3	0
3	3	한 번에 40%를 올리는 황당한 상황을 보고 장거리 택시를 타야 하는 시간과 장소에...	4	1
4	4	솔직히 서울은 진짜 급하다고 택시타는게 거의 통하지 않는 곳이라서 더 사람들이 안타...	5	1

대중의 심리 반영

5. 감성 분류-Logistic Regression

2) Training set/Test set 나누기

```
from sklearn.model_selection import train_test_split

x = tf_idf_vect
y = df['y']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state=1)
```

```
x_train.shape, y_train.shape
```

```
((938, 2796), (938,))
```

```
x_test.shape, y_test.shape
```

```
((402, 2796), (402,))
```

대중의 심리 반영

5. 감성 분류-Logistic Regression

3) 모델 학습

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# fit in training set
lr = LogisticRegression(random_state = 0)
lr.fit(x_train, y_train)

# predict in test set
y_pred = lr.predict(x_test)
```

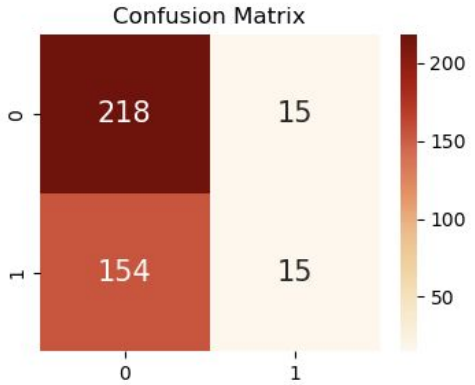
```
# classification result for test set

print('accuracy: %.2f' % accuracy_score(y_test, y_pred))
print('precision: %.2f' % precision_score(y_test, y_pred))
print('recall: %.2f' % recall_score(y_test, y_pred))
print('F1: %.2f' % f1_score(y_test, y_pred))
```

```
accuracy: 0.58
precision: 0.50
recall: 0.09
F1: 0.15
```

대중의 심리 반영

5. 감성 분류-Logistic Regression

```
# confusion matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

confu = confusion_matrix(y_true = y_test, y_pred = y_pred)

plt.figure(figsize=(4, 3))
sns.heatmap(confu, annot=True, annot_kws={'size':15}, cmap='OrRd', fmt='.10g')
plt.title('Confusion Matrix')
plt.show()
```



모델 평가결과를 살펴보면, 모델이 지나치게 부정(“0”)으로만 예측하는 경향이 있음.

즉, 부정 리뷰를 잘 예측하지만 긍정 리뷰에 대한 예측 정확도가 매우 낮음.

이는 샘플데이터의 클래스 불균형으로 인한 문제로 보이기에 클래스 불균형 조정이 필요해보임.

대중의 심리 반영

5. 감성 분류-Logistic Regression

4) 샘플링 재조정

```
df['y'].value_counts()
```

```
0    804
1    536
Name: y, dtype: int64
```

```
positive_random_idx = df[df['y']==1].sample(275, random_state=12).index.tolist()
negative_random_idx = df[df['y']==0].sample(275, random_state=12).index.tolist()
```

```
random_idx = positive_random_idx + negative_random_idx
x = tf_idf_vect[random_idx]
y = df['y'][random_idx]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=1)
```

```
x_train.shape, y_train.shape
```

```
((412, 2796), (412,))
```

```
x_test.shape, y_test.shape
```

```
((138, 2796), (138,))
```

대중의 심리 반영

5. 감성 분류-Logistic Regression

5) 모델 재학습

```
lr2 = LogisticRegression(random_state = 0)
lr2.fit(x_train, y_train)
y_pred = lr2.predict(x_test)
```

```
# classification result for test set

print('accuracy: %.2f' % accuracy_score(y_test, y_pred))
print('precision: %.2f' % precision_score(y_test, y_pred))
print('recall: %.2f' % recall_score(y_test, y_pred))
print('F1: %.2f' % f1_score(y_test, y_pred))
```

```
accuracy: 0.50
precision: 0.49
recall: 0.50
F1: 0.50
```

결과적으로, 모델 재학습 결과 정확도와 정밀도는 약간 감소했지만, 재현율과 F1 스코어는 크게 증가함.

이는 모델이 더 많은 실제 양성을 잘 예측하면서 일부 정확도를 포기했으며, 모델의 성능이 전반적으로 개선되었다는 것을 의미함.

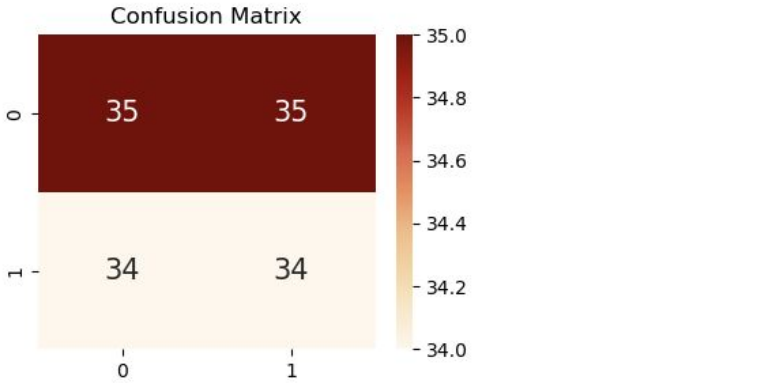
cf) F1: 정밀도와 재현율의 조화 평균으로 계산되는 지표

대중의 심리 반영

5. 감성 분류-Logistic Regression

5) 모델 재학습

```
# confusion matrix

from sklearn.metrics import confusion_matrix

confu = confusion_matrix(y_true = y_test, y_pred = y_pred)

plt.figure(figsize=(4, 3))
sns.heatmap(confu, annot=True, annot_kws={'size':15}, cmap='OrRd', fmt='.10g')
plt.title('Confusion Matrix')
plt.show()
```



대중의 심리 반영

6. 긍정/부정 키워드 분석

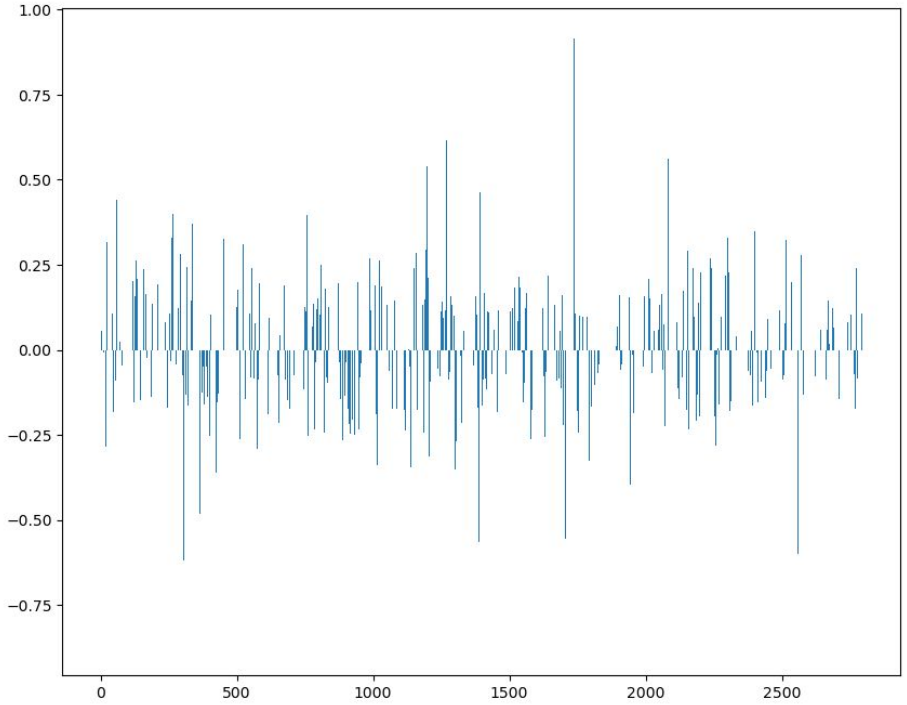
1) 각 단어의 **coefficient** 시각화

```
lr2.coef_
```

```
array([[ 0.29186529, -0.23766842, -0.09075965, ...,  0.1057736 ,
         0.        ,  0.14807588]])
```

```
# print logistic regression's coef

plt.figure(figsize=(10, 8))
plt.bar(range(len(lr2.coef_[0])), lr2.coef_[0])
```

```
<BarContainer object of 2796 artists>
```



대중의 심리 반영

6. 긍정/부정 키워드 분석

2) 긍정/부정 키워드 리스트 정의 & 출력

```
coef_pos_index = sorted(((value, index) for index, value in enumerate(lr2.coef_[0])), reverse = True)
coef_neg_index = sorted(((value, index) for index, value in enumerate(lr2.coef_[0])), reverse = False)
coef_pos_index
```

```
[(0.9139148697097403, 1737),
 (0.9105536662303744, 2282),
 (0.67766968181376, 641),
 (0.6142161329779281, 1269),
 (0.5944201041937243, 1606),
 (0.574251160274271, 152),
 (0.5602008565475167, 2083),
 (0.5415520948950141, 322),
 (0.539991835610359, 1198),
 (0.5307550787041556, 913),
 (0.5176635700137586, 524),
 (0.5092608552355544, 1760),
 (0.5064474439811009, 2218),
 (0.49698409664979676, 2758),
 (0.4879513599778237, 46),
 (0.4820074418537661, 1373),
 (0.47395596475650376, 2636),
 (0.47332049162219064, 1846),
 (0.46505475030374854, 783),
```

대중의 심리 반영

6. 긍정/부정 키워드 분석

3) index -> 단어 변환

```
invert_index_vectorizer = {v: k for k, v in vect.vocabulary_.items()}
invert_index_vectorizer
```

```
{1186: '사람',
 1252: '생각',
 2300: '진짜',
 384: '기적',
 2253: '지능',
 372: '기사',
 1413: '시위',
 2173: '조합',
 1115: '분배',
 1839: '이번',
 911: '문제',
 322: '그동안',
 1600: '업계',
 1374: '스스로',
 1974: '자정',
 1411: '시스템',
 778: '마련',
 617: '대한',
 144: '결과',
```

대중의 심리 반영

6. 긍정/부정 키워드 분석

[Top 20 긍정 키워드 리스트]

```
for coef in coef_pos_index[:20]:
    print(invert_index_vectorizer[coef[1]], coef[0])
```

운전 0.9139148697097403
지하철 0.9105536662303744
도로 0.67766968181376
서울 0.6142161329779281
업체 0.5944201041937243
경기도 0.574251160274271
절반 0.5602008565475167
그동안 0.5415520948950141
사업자 0.539991835610359
물가 0.5307550787041556
다른 0.5176635700137586
월급 0.5092608552355544
중간 0.5064474439811009
화이팅 0.49698409664979676
간다 0.4879513599778237
스미스 0.4820074418537661
하니 0.47395596475650376
이야기 0.47332049162219064
마음 0.4650547503037485 4
시간 0.464393144687646

[Top 20 부정 키워드 리스트]

```
for coef in coef_neg_index[:20]:
    print(invert_index_vectorizer[coef[1]], coef[0])
```

덕분 -0.8677261148796194
일찍 -0.7894892715496942
문제 -0.6840995652355156
법인 -0.6285778808722096
귀가 -0.618188147910983
한번 -0.6082523184144087
파업 -0.6006574282204555
정도 -0.5998105426371465
승차 -0.5657614658159098
요금인상 -0.5558596829821437
카카오 -0.5380754051420519
불만 -0.5321804136061533
정책 -0.5276870950096644
정말 -0.5234909875935251
일부러 -0.5005116103443894
평소 -0.4998660528660913 7
거기 -0.4900348293382298 4
기도 -0.480332089929977 6
개택들 -0.4768622900333695
소비자 -0.4714063338694657

추후 계획

1. 10개의 영상 웹 크롤링-> Logistic Regression 감성분류 모델 결과 해석

2. 토픽 모델링-LDA

귀가시간

1. 지역구 5개 선정

-사용 데이터: 행정구역(시군구)별 주민등록세대수
https://kosis.kr/statHtml/statHtml.do?orgId=101&tblId=DT_1B040A3

-선정기준: 서울특별시 내 주민등록세대수 1위 ~ 5위

-선정지역: 1. 송파구 2. 강서구 3. 강남구 4. 노원구 5. 관악구

택시 기본요금 인상

1. 지역구별 지하철역 분류

-**송파구**: '잠실나루', '잠실(송파구청)', '잠실새내', '종합운동장', '가락시장', '경찰병원', '오금', '올림픽공원(한국체대)', '방이', '개롱', '거여', '마천', '몽촌토성(평화의문)', '석촌', '송파', '문정', '장지', '복정', '삼전', '석촌고분', '송파나루', '한성백제'

-**강서구**: '방화', '개화산', '김포공항', '송정', '마곡', '발산', '우장산', '화곡', '까치산', '개화', '공항시장', '신방화', '마곡나루(서울식물원)', '양천향교', '가양', '증미', '등촌'

-**강남구**: '삼성(무역센터)', '선릉', '역삼', '강남', '압구정', '신사', '매봉', '도곡', '대치', '학여울', '대청', '일원', '수서', '청담', '강남구청', '학동', '논현', '언주', '선정릉', '삼성중앙', '봉은사', '신논현', '압구정로데오', '한티', '구룡', '개포동', '대모산입구'

-**노원구**: '당고개', '상계', '노원', '석계', '태릉입구', '화랑대(서울여대입구)', '수락산', '마들', '중계', '하계', '공릉(서울과학기술대)', '월계', '광운대'

-**관악구**: '낙성대(강감찬)', '서울대입구(관악구청)', '봉천', '신림', '당곡', '서원', '서울대벤처타운', '관악산(서울대)'

택시 기본요금 인상

2. 분석 기간 선정

-분석 기간: **2022년 12월 1일 ~ 2023년 3월 31일**

-특이사항: 2022년 12월 1일에 심야택시 요금 할증시간 및 할증률이 높아져, 2022년 12월 1일 이전 데이터는 보지 않는 것으로 결정정

택시 기본요금 인상

3. 사용 데이터 선정

1. 서울시 지하철호선별 역별 승하차 인원 정보

사용일자	호선명	역명	승차총승객수	하차총승객수	등록일자
20230724	우이신설선	4.19민주묘지	3,087	2,986	20230727
20230724	경원선	가능	6,711	6,436	20230727
20230724	8호선	가락시장	7,908	8,683	20230727
20230724	3호선	가락시장	9,599	9,301	20230727
20230724	경부선	가산디지털단지	18,596	21,388	20230727

2. 서울시 지하철 호선별 역별 시간대별 승하차 인원 정보

사용월	호선명	지하철역	04시-05시 승...	04시-05시 하...	05시-06시 승...	05시-06시 하...	06시-07시 승..
202306	1호선	동대문	854	33	11,997	2,157	
202306	1호선	동묘앞	228	2	3,088	1,143	
202306	1호선	서울역	677	33	8,470	9,569	1
202306	1호선	시청	61	1	2,351	4,528	
202306	1호선	신설동	494	24	9,258	2,481	

택시 기본요금 인상

4. 데이터 전처리

-기존 데이터에 각 지하철역에 맞는 지역구 열 추가해 각 월별 데이터 프레임 생성
-> 하나의 데이터 프레임으로 통합

		하차총승객수
사용일자	area	
20221201	강남구	690730
	강서구	235186
	관악구	173855
	노원구	205567
	송파구	329212
...	...	...
20221231	강남구	340027
	강서구	140370
	관악구	116160
	노원구	131549
	송파구	236602

155 rows × 1 columns

택시 기본요금 인상

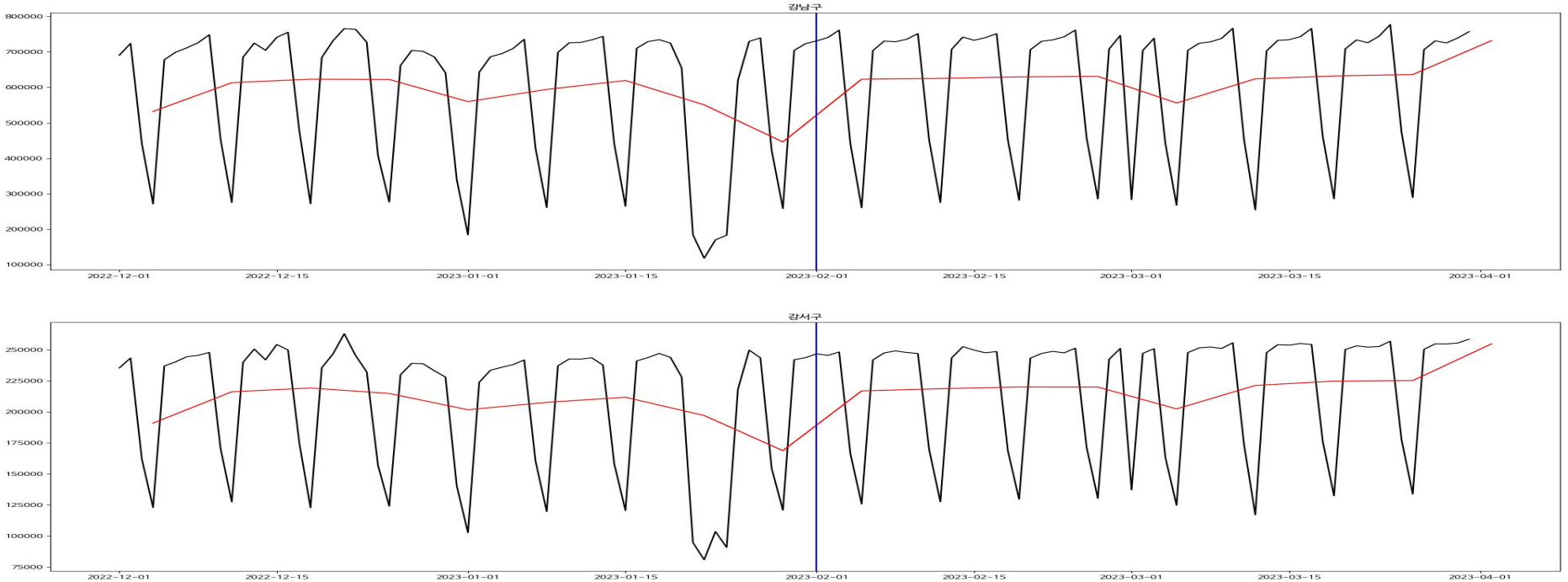
5. 정상성 검정

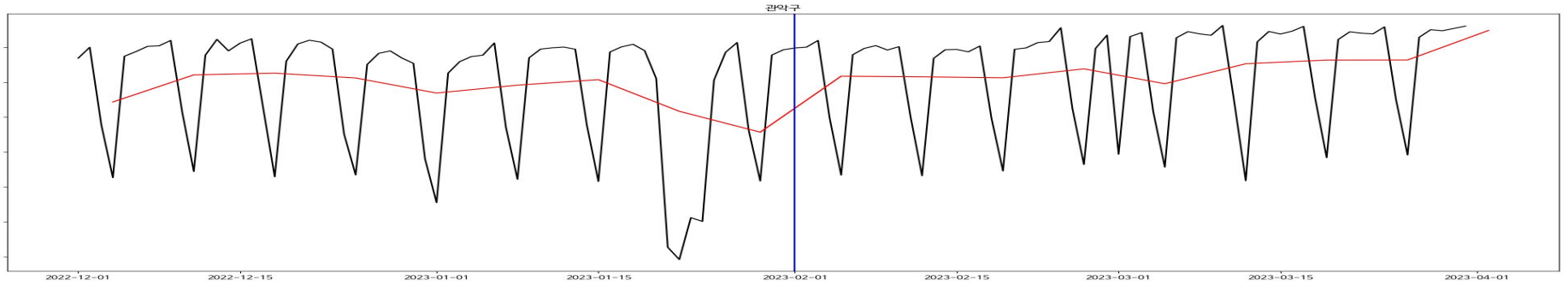
```
def adfuller_test(df):
        result=adfuller(df)

        labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations Used']

        for value,label in zip(result,labels):
            print(label+' : '+str(value) )

        if result[1] <= 0.05:
            print("stationary")
        else:
            print("non-stationary")
```

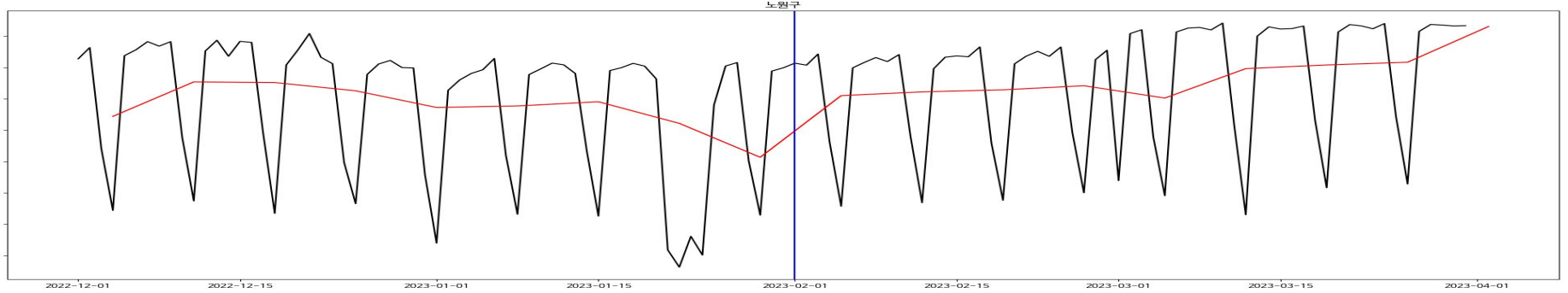
-관악구, 노원구 **=>** 비정상성

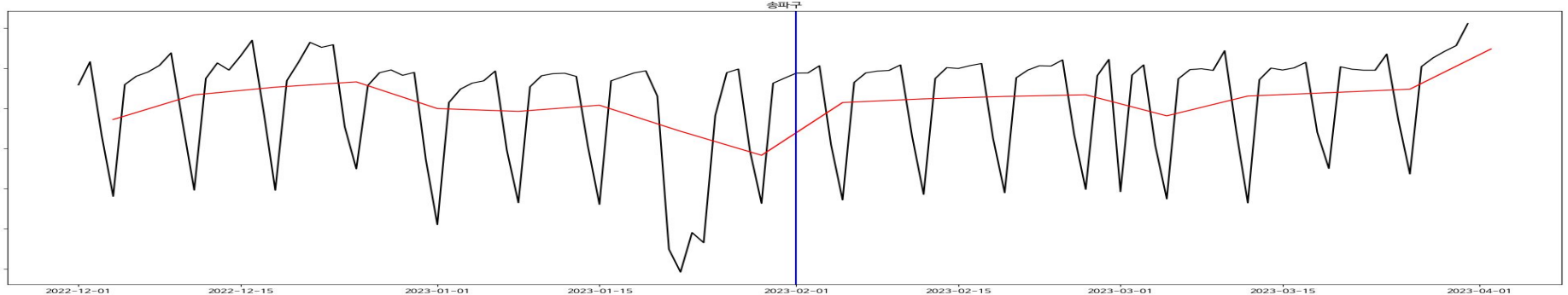
택시 기본요금 인상

6. 전체적인 주별 추세 확인



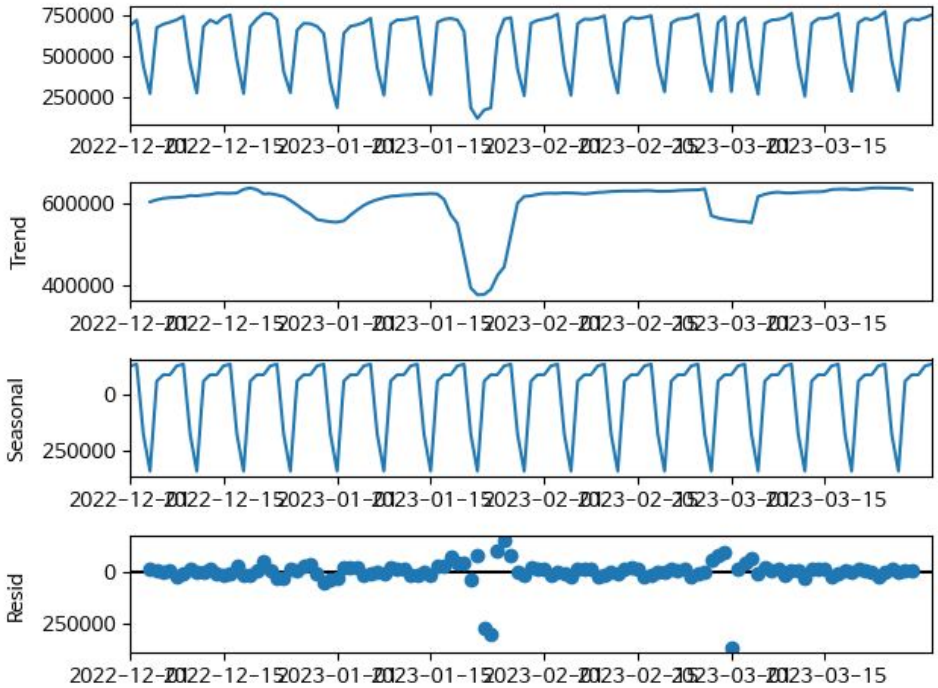

관악구
180000
160000
140000
120000
100000
80000
60000
2022-12-01
2022-12-15
2023-01-01
2023-01-15
2023-02-01
2023-02-15
2023-03-01
2023-03-15
2023-04-01


노원구
220000
200000
180000
160000
140000
120000
100000
80000
2022-12-01
2022-12-15
2023-01-01
2023-01-15
2023-02-01
2023-02-15
2023-03-01
2023-03-15
2023-04-01


송파구
400000
350000
300000
250000
200000
150000
100000
2022-12-01
2022-12-15
2023-01-01
2023-01-15
2023-02-01
2023-02-15
2023-03-01
2023-03-15
2023-04-01

택시 기본요금 인상

6. seasonal decompose



택시 기본요금 인상

7. 추후 계획

-앞선 결과들을 봤을 때, 2월 1일 기준으로 살짝 올라가는 것 같지만, 유의미하게 증가하는 것 같지는 않음

-(S)ARIMA 모델을 적합한 후, 표준화 잔차를 구해 이상치를 탐지하는 방법을 사용하고자 함

만약 2월 이후에 이상치가 높은 일자의 개수가 2월 이전에 비해 많다면, 영향이 존재한다고 판단

만약 2월 이후에 이상치가 높은 일자의 개수가 2월 이전과 비슷하거나 적다면, 영향이 없다고 판단

-한계점: 데이터의 부족으로 인해 귀가시간대로만 분석하는 것이 불가. 시간대별 데이터의 단위가 월 단위이기 때문.