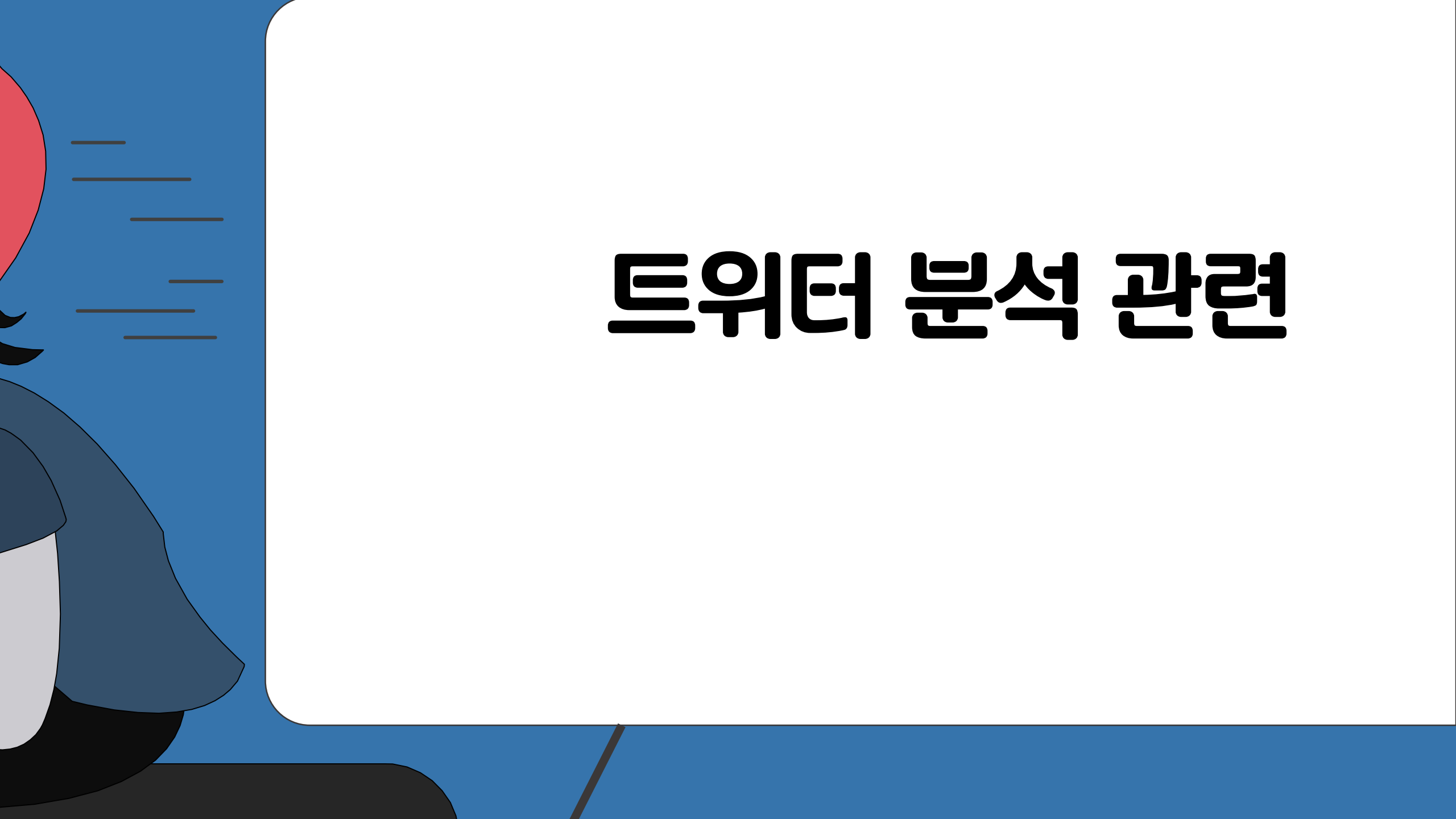


빅데이터 관련 세부설명

모르겠으면
물어봐!!

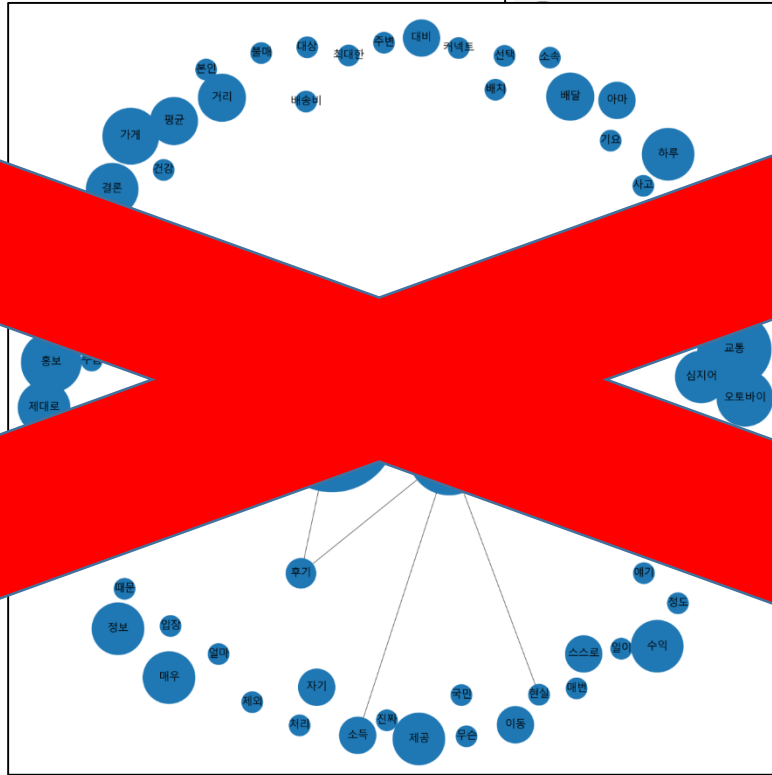


트위터분석
연관분석
데이터 전처리
데이터 시각화
회귀분석



트위터 분석 관련

The diagram shows a network of blue circular nodes connected by thin lines. The nodes are labeled with Korean text: '가', '김민', '하루', '김교', '교통', '삼지여', '오토바이', '우거', '배', '전도', '홍보', '제대로'. A large, thick red 'X' is superimposed over the entire network, indicating that this structure is incorrect or rejected.



0 RT @FF14_Orte: 근데 자만추 점점 클럽? 나이트? 분위기가 디폴트로 되어...

1 아ㅋㅋ 친구가 O딘 영상 보여줘서 보는데, 시네마틱 되게 잘뽑았다 디자인 좋다 이러...

2 RT @cumo_0: 정우한테 돈세노 가사 따와서..잠깐 식었던 시간동안 그냥 ...

**그지같은 데이터 내용 + 얼마 없는 데이터
트위터 분석은 시각화 자료도 잘못된 것!!
쓰지 마세요~**



연관분석식 관련

연관분석용 키워드 수집

배달 관련 크롤링 ¶

```
6]: from selenium import webdriver
    from selenium.webdriver.common.keys import Keys # 엔터키 입력용
    import pandas as pd

    driver = webdriver.Chrome('C:/Temp/chromedriver') # 웹드라이버 객체 생성

    # 구글을 열어서 키워드 입력 후 이동
    driver.get('http://www.google.com') # 구글 홈페이지 내용 렌더링
    target=driver.find_element_by_css_selector("[name = 'q']") # 검색어 지정
    target.send_keys('라이더') # 타겟에 '라이더' 입력
    target.send_keys(Keys.ENTER) # 엔터처리
    |
    # 뉴스 페이지로 이동
    news = driver.find_element_by_css_selector('#hdtb-msb-vis > div:nth-child(4) > a')
    news.click()

    # 헤드라인 수집
    untact_title = []
    untact_time = []

    # 알고리즘을 짜보자
    # 다음 페이지 -> #pnnext > span:nth-child(2) // 이걸 19번해야 함 / 마지막에 누르자
    for i in range(18):
```

연관분석용 키워드 수집결과

뉴스기사제목

0	메쉬코리아, DB손해보험과 라이더 위한 보험 개발 나선다
1	전국배달라이더협회, 2020년 정책연구용역 착수보고회 개최
2	레다텍, 밀라그룹 자율주행 셔틀에 라이더 공급
3	이마트에 '카트라이더 러쉬플러스'가 떴다
4	라이더 수수료 올리고 배달주문은 반값세일 - 서울경제
...	...
172	코로나발 언택트 열풍에 올라탄 넥슨, 피파.카트라이더 고속질주
173	Vet accused of inventing expert to greenlight ...
174	'비싼 몸' 배달 라이더 공급 잡는다...유현철 스파이더크래프트 ...
175	배달라이더 '억'대 연봉, 실현 가능할까[영상] - 서울경제
176	Queensland rider Baylee Nothdurft to take an e...

177 rows × 1 columns

177개 데이터 (기사 타이틀)

연관분석용 키워드 전처리

	뉴스기사제목	특수문자제거
172	코로나발 언택트 열풍에 올라탄 넥슨, 피파.카트라이더 고속질주	코로나발 언택트 열풍에 올라탄 넥슨 피파카트라이더 고속질주
173	Vet accused of inventing expert to greenlight ...	
174	'비싼 몸' 배달 라이더 공급 잡는다...유현철 스파이더크래프트 ...	비싼 몸 배달 라이더 공급 잡는다유현철 스파이더크래프트
175	배달라이더 '억'대 연봉, 실현 가능할까[영상] - 서울경제	배달라이더 억대 연봉 실현 가능할까영상 서울경제
176	Queensland rider Baylee Nothdurft to take an e...	

영어 및 특수문자 제거

	뉴스기사제목	특수문자제거	명사
172	코로나발 언택트 열풍에 올라탄 넥슨, 피파.카트라이더 고속질주	코로나발 언택트 열풍에 올라탄 넥슨 피파카트라이더 고속질주	[코로, 나발, 언택트, 열풍, 넥슨, 피파, 카트라이더, 고속, 질주]
173	Vet accused of inventing expert to greenlight ...		[]
174	'비싼 몸' 배달 라이더 공급 잡는다...유현철 스파이더크래프트 ...	비싼 몸 배달 라이더 공급 잡는다유현철 스파이더크래프트	[배달, 라이더, 공급, 유현, 스파이더, 크래프트]
175	배달라이더 '억'대 연봉, 실현 가능할까[영상] - 서울경제	배달라이더 억대 연봉 실현 가능할까영상 서울경제	[배달, 라이더, 억대, 연봉, 실현, 영상, 서울, 경제]
176	Queensland rider Baylee Nothdurft to take an e...		[]

명사 추출

키워드 연관분석 코드

```
dataset = [['배달', '오토바이', '사고', '기관', '소식', '정책'], ['코로나', '집콕'],  
  
# 데이터셋을 원핫인코딩처럼 트랜스포밍해주기!  
te = TransactionEncoder()  
te_result = te.fit(dataset).transform(dataset)  
  
# 트랜스포밍된 결과를 데이터프레임으로 형성  
# 컬럼명이 키워드 / 각 row마다 포함된 키워드에 True를 반환  
df = pd.DataFrame(te_result, columns=te.columns_)  
  
# apriori 알고리즘 사용 / 옵션을 주지 않으면 자동으로 지지도 0.5  
keywordset = apriori(df, use_colnames=True)  
display(keywordset)  
  
# 지지도를 0.1로 낮춰서 apriori 실행  
keywordset = apriori(df, min_support=0.1, use_colnames=True)  
display(keywordset.head())  
  
# 신뢰도 추가 / 기본값은 0.8 / 낮게 설정해야 목록이 나옴 / 0.1로 설정  
association_rules(keywordset, metric="confidence", min_threshold=0.1).head(10)
```


키워드 연관분석

미리 알아두면 좋을 용어

support = A와 B가 동시에 일어난 확률 / 전체 횟수

confidence = A와 B가 동시에 일어난 확률 / A가 일어난 횟수

lift = A와 B가 동시에 일어난 확률 / A와 B가 독립일 확률

키워드 연관분석 결과

라이더 키워드와 연관분석

	support	itemsets
0	0.872727	(배달)
1	0.554545	(안전)

지지도가 0.5인 키워드
안전이 나왔다는 것을 강조하자!!

키워드 연관분석 결과

라이더 키워드와 연관분석

	support	itemsets
0	0.127273	(뉴스)
1	0.172727	(라이더)
2	0.109091	(민족)
3	0.872727	(배달)
4	0.136364	(사고)

지지도가 0.1인 키워드
안전, 사고가 나왔다는 것을
강조하자!!
head() 를 줘서 앞의 다섯개
만 출력된 거 / **사고** 밑에 **안**
전 키워드 있음

키워드 연관분석 결과

라이더 키워드와 연관분석

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(배달)	(라이더)	0.872727	0.172727	0.172727	0.197917	1.145833	0.021983	1.031405
1	(라이더)	(배달)	0.172727	0.872727	0.172727	1.000000	1.145833	0.021983	inf
2	(안전)	(라이더)	0.554545	0.172727	0.109091	0.196721	1.138913	0.013306	1.029870
3	(라이더)	(안전)	0.172727	0.554545	0.109091	0.631579	1.138913	0.013306	1.209091
4	(민족)	(배달)	0.109091	0.872727	0.109091	1.000000	1.145833	0.013884	inf
5	(배달)	(민족)	0.872727	0.109091	0.109091	0.125000	1.145833	0.013884	1.018182
6	(사고)	(배달)	0.136364	0.872727	0.136364	1.000000	1.145833	0.017355	inf
7	(배달)	(사고)	0.872727	0.136364	0.136364	0.156250	1.145833	0.017355	1.023569
8	(배달)	(안전)	0.872727	0.554545	0.472727	0.541667	0.976776	-0.011240	0.971901
9	(안전)	(배달)	0.554545	0.872727	0.472727	0.852459	0.976776	-0.011240	0.862626

향상도(lift)를 0.1로 준 값
서로 관련있는 키워드를 추천해준 것
안전 관련해서 강조하자!!



**여기서부터는
사고 데이터 관련**

데이터 확인

```
# ./fusion/data/accident
# 전년도도 같이 가져와야 함

accident_kind1 = pd.read_csv('./data/accident/서울시 교통사고 현황 (자동차종류별) 통계.txt', sep="\t")
accident_safe_count = pd.read_csv('./data/accident/서울시 교통안전지수 통계.txt', sep="\t")
accident_safetyproduct = pd.read_csv('./data/accident/서울시 보호장구 착용여부별 교통사고 사상자수 통계.txt', sep="\t")
accident_bike = pd.read_csv('./data/accident/서울시 자전거 교통사고 통계.txt', sep="\t")
accident_kind2 = pd.read_csv('./data/accident/서울시 차량용도별 교통사고 현황 통계.txt', sep="\t")
```

사고 데이터 확인

데이터 전처리

```
# 서울시 교통사고 현황 통계
import pandas as pd
```

```
# 데이터 불러오기
accident_kind1 = pd.read_csv('./data/accident/서울시 차량용도별 교통사고 현황 통계_2021년.csv')
```

```
# 데이터 확인
accident_kind1
```

```
# 서울시 보호장구 착용여부 통계
import pandas as pd
```

```
# 데이터 불러오기 // 구분
accident_safetyproduct = pd.read_csv('./data/accident/서울시 보호장구 착용여부 통계_2021년.csv')
```

```
# 필요없는 컬럼 제거
accident_safetyproduct.drop('구분', axis=1, inplace=True)
display(accident_safetyproduct)
```

```
## 딱봐도 이건 아니다..
accident_safetyproduct.to_excel('./data/accident/서울시 보호장구 착용여부 통계_2021년.xlsx')
```

```
# 서울시 교통안전 통계
import pandas as pd
```

```
# 데이터 불러오기
accident_safe_count = pd.read_csv('./data/accident/서울시 교통안전 통계_2021년.csv')
```

```
# 합계 지우기
accident_safe_count.drop('합계', axis=1, inplace=True)
```

```
# 서울시 자전거 교통사고 통계
import pandas as pd
```

```
# 데이터 불러오기
accident_bike = pd.read_csv('./data/accident/서울시 자전거 교통사고 통계_2021년.csv')
```

```
# 합계 지우기
accident_bike.drop(0, axis=1, inplace=True)
```

```
# 데이터 확인
display(accident_bike)
```

```
# 데이터 저장
accident_bike.to_excel('./data/accident/서울시 자전거 교통사고 통계_2021년.xlsx')
```

```
# 서울시 차량용도별 교통사고 현황 통계
import pandas as pd
```

```
# 데이터 불러오기
```

```
accident_kind2 = pd.read_csv('./data/accident/서울시 차량용도별 교통사고 현황 통계_2021년.csv')
```

```
# 데이터 전처리
```

```
# 1. 필요없는 컬럼 제거
```

```
accident_kind2.drop(accident_kind2.columns[[3,4,5,6,7,8]], axis=1, inplace=True)
```

```
# 2. 필요없는 행 제거
```

```
accident_kind2.drop(accident_kind2.index[[0,1,2,3,4]], axis=0, inplace=True)
```

```
# 구분별로 구분하여 발생건수만 출력
```

```
accident_kind2 = accident_kind2.groupby('구분')
```

```
accident_kind2_EDA = accident_kind2.get_group('발생건수')
```

```
# 확인
```

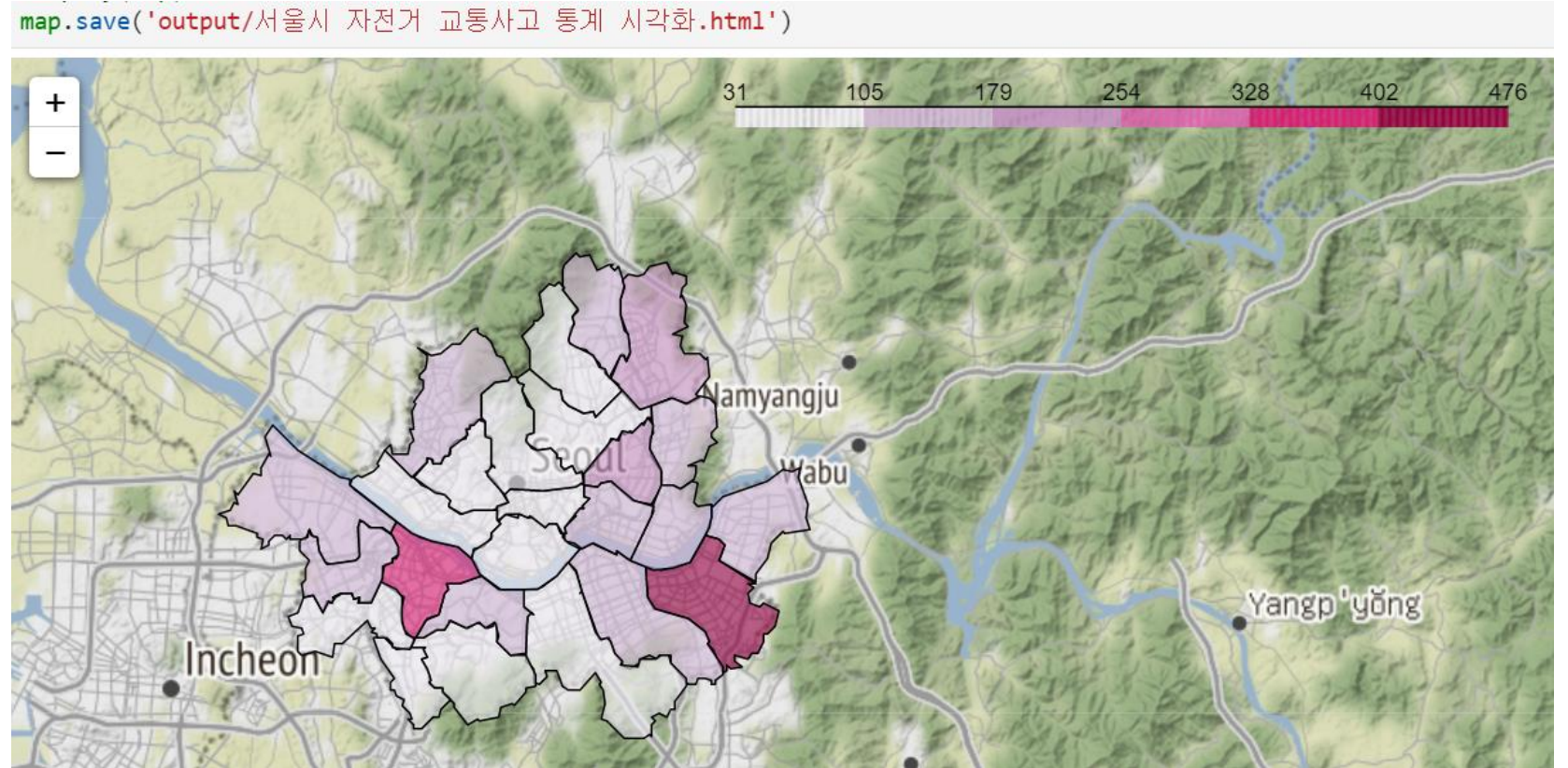
```
display(accident_kind2_EDA)
```

```
# 데이터 저장
```

```
accident_kind2_EDA.to_excel('./data/accident/서울시 차량용도별 교통사고 현황 통계_2021년_EDA.xlsx')
```

자세한 설명은 생략한다

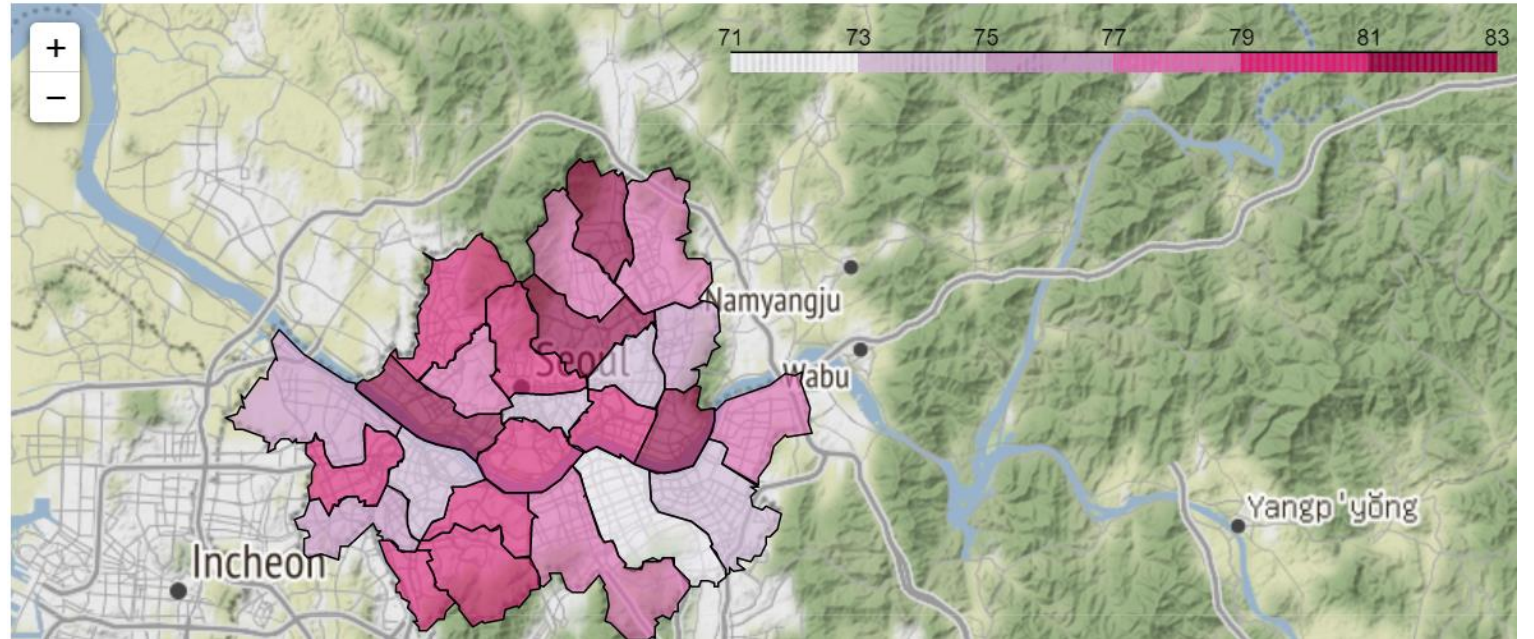
데이터 시각화



자전거 사고는 송파구와 영등포구에 많다

데이터 시각화

```
# 강남구가 제일 위험!!  
map.save('output/서울시 교통안전지수 통계 시각화.html')
```

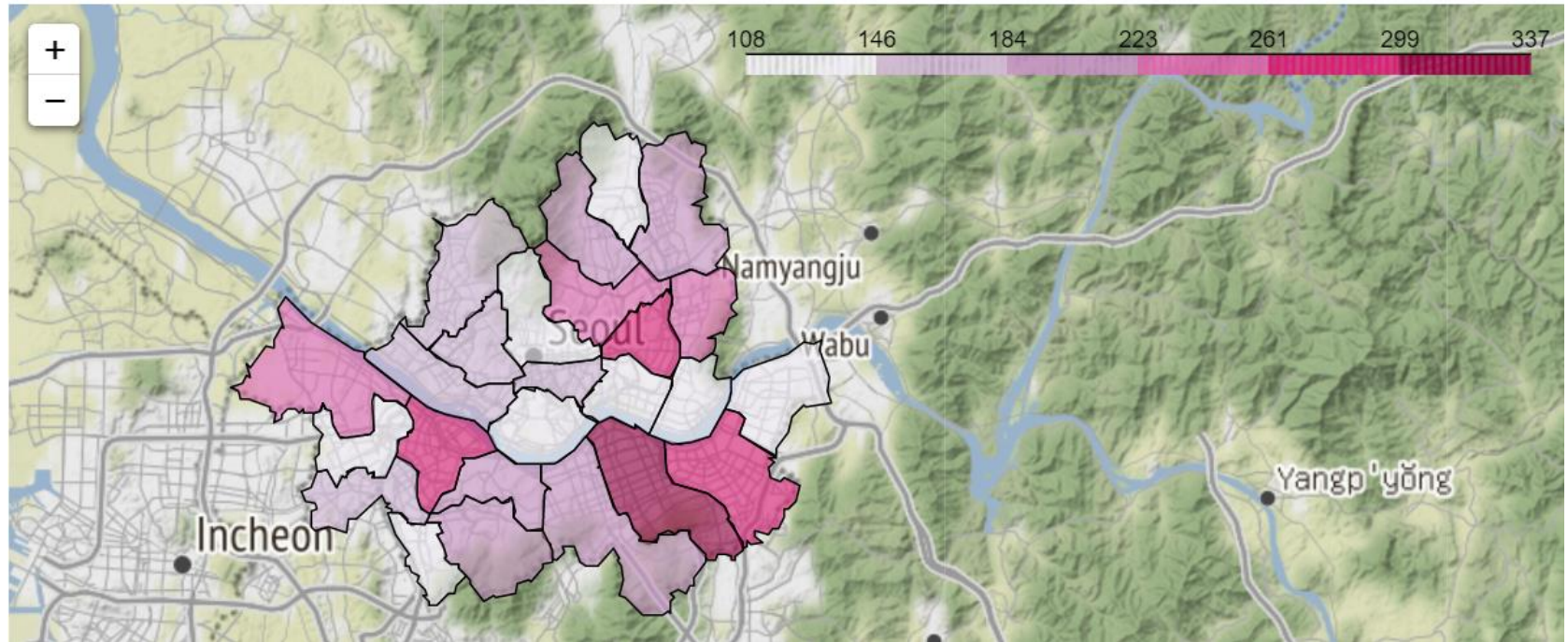


교통 안전지수로는 강남구가 제일 위험!

데이터 시각화

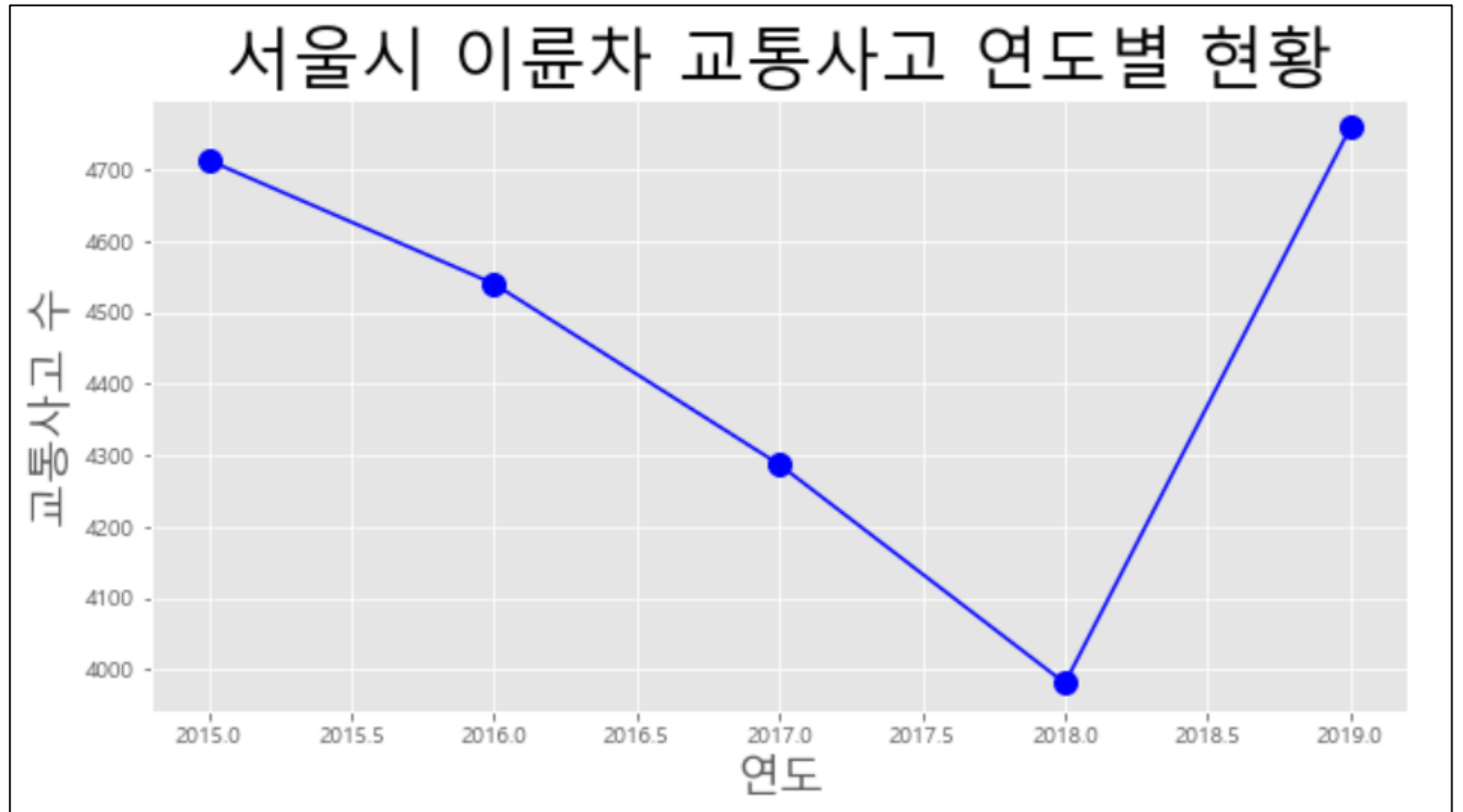
강남구가 제일 위험!!

```
map.save('output/서울시 차량용도별 교통사고 현황 통계_전처리 시각화.html')
```



이런차 사고건수도 강남구가 제일 많음!

데이터 시각화



**2019년부터 이륜차사고가 급증
앞의 배달-안전 과 역어서 설명!!**



**여기서부터는
배달 데이터 관련**

데이터 확인

```
# 배달 관련 데이터 확인 // # ./fusion/data/delivery

delivery_shop = pd.read_csv('./data/delivery/배달_상점_데이터.csv', header=None)
delivery_shop.columns = ['상점ID', '업종명', '상점코드', '시도명', '구군명', '읍면동명', '법정동리명', '행정동교']

# delivery_loc = pd.read_csv('./data/delivery/배달자_위치정보_데이터.csv')

delivery_time_mean = pd.read_csv('./data/delivery/시간-지역별_배달_소요시간평균.csv', header=None)
delivery_time_mean.columns = ['날짜', '시간', '도', '시구', '평균시간']

delivery_time_order_count = pd.read_csv('./data/delivery/시간-지역별_배달_주문건수.csv', header=None)
delivery_time_order_count.columns = ['날짜', '시간', '도', '시구', '주문건수']

delivery_time_pay_mean = pd.read_csv('./data/delivery/시간-지역별_배달_평균주문금액.csv', header=None)
delivery_time_pay_mean.columns = ['날짜', '시간', '도', '시구', '평균주문금액']

delivery_job_order_count = pd.read_csv('./data/delivery/업종-지역별_배달_주문건수.csv', header=None)
delivery_job_order_count.columns = ['날짜', '업종', '도', '시', '주문건수']

delivery_job_time_mean = pd.read_csv('./data/delivery/업종-지역별_평균배달소요시간.csv', header=None)
delivery_job_time_mean.columns = ['날짜', '업종', '도', '시', '평균시간']

delivery_job_pay_mean = pd.read_csv('./data/delivery/업종-지역별_평균주문금액.csv', header=None)
delivery_job_pay_mean.columns = ['날짜', '업종', '도', '시', '평균주문금액']

delivery_hum = pd.read_csv('./data/delivery/주문지역_인구_특성.csv', header=None)
delivery_hum.columns = ['기준연월', '시군구코드', '시도명', '시군구명', '5세대위구분', '총인구수', '남성인구수',
```

배달 데이터 확인

데이터 전처리

```
# 배달 상점 데이터 전처리
import pandas as pd
```

```
# 시간-지역별 배달 소요시간
```

```
# 시간-지역별 배달 주문건수
```

```
import pandas as pd
```

```
# 시간-지역별 배달 주문건수
```

```
# 업종-지역별 배달 주문건수
```

```
import pandas as pd
```

```
# 데이터 불러오기
```

```
delivery_job_pay_mean = pd.read_csv('./data/delivery/업종별_지역별_배달_주문건수.csv')
```

```
delivery_job_pay_mean.columns = ['날짜', '업종', '시', '구', '평균주문금액']
```

```
# 서울특별시만 걸러내기
```

```
delivery_job_pay_mean = delivery_job_pay_mean[delivery_job_pay_mean['시'] == '서울특별시']
```

```
df5 = delivery_job_pay_mean
```

```
# 업종별 구분
```

```
df5.drop(df5[df5['업종'] == '기타'], axis=1, inplace=True)
```

```
display(df5)
```

```
# 몇 개인지 확인
```

```
df5 = df5.groupby('구')
```

```
print(len(df5))
```

```
# 업종-지역별 평균배달소요시간
```

```
import pandas as pd
```

```
# 데이터 불러오기
```

```
delivery_job_pay_mean = pd.read_csv('./data/delivery/업종별_지역별_배달_주문건수.csv')
```

```
delivery_job_pay_mean.columns = ['날짜', '업종', '시', '구', '평균주문금액']
```

```
# 서울특별시만 걸러내기
```

```
delivery_job_pay_mean = delivery_job_pay_mean[delivery_job_pay_mean['시'] == '서울특별시']
```

```
df6 = delivery_job_pay_mean
```

```
# 업종별 구분
```

```
df6.drop(df6[df6['업종'] == '기타'], axis=1, inplace=True)
```

```
display(df6)
```

```
# 몇 개인지 확인
```

```
df6 = df6.groupby('구')
```

```
print(len(df6))
```

```
# 업종-지역별 평균주문금액
```

```
import pandas as pd
```

```
# 데이터 불러오기
```

```
delivery_job_pay_mean = pd.read_csv('./data/delivery/업종별_지역별_배달_주문건수.csv')
```

```
delivery_job_pay_mean.columns = ['날짜', '업종', '시', '구', '평균주문금액']
```

```
# 서울특별시만 걸러내기
```

```
# 주문지역 주거 특성
```

```
import pandas as pd
```

```
# 데이터 불러오기
```

```
delivery_home = pd.read_csv('./data/delivery/주문지역_주거특성.csv')
```

```
delivery_home.columns = ['날짜', '시군구코드', '시', '구', '주거특성']
```

```
# 서울특별시만 걸러내기
```

```
delivery_home = delivery_home.groupby('시')
```

```
df9 = delivery_home.get_group('서울특별시')
```

```
# 필요한 컬럼만 남기자 -> 날짜, 시군구코드 삭제
```

```
df9.drop(df9.columns[1], axis=1, inplace=True)
```

```
display(df9)
```

```
# 몇 개인지 확인하기
```

```
# df8 = df8.groupby('구')
```

```
# print(len(df8)) # 전체 있는 것 확인 = 25
```

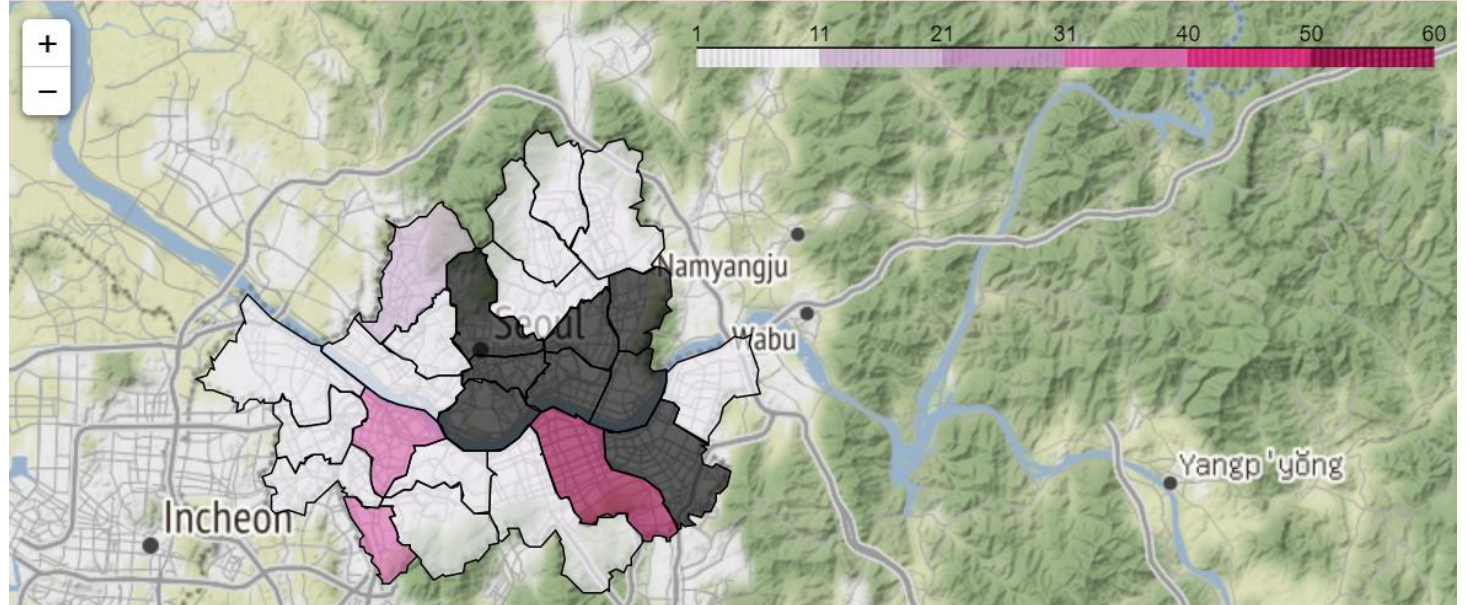
자세한 설명은 생략한다

데이터 시각화

```
map.save('output/시간-지역별 배달 주문건수 0시_누락데이터 확인.html')
```

C:\Users\parkgun\anaconda3\envs\pydataenv\lib\site-packages\pandas\core\frame.py:4170: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#reindexing-a-view-versus-a-copy
errors=errors,



누락 데이터가 너무 많아..
대체로 강남에 배달건수가 많음!!
강남을 타겟팅하여 분석

데이터기반 정보제공 To 회사

```
# 구 별로 묶기
gangnam = time_zero.groupby('구')
gangnam = gangnam.get_group('강남구')
# display(gangnam)

# 예상 주문건수 --> 지금까지 있었던 모든 날짜의 평균시간을 더한 뒤 평균 값을 전달
print("강남구", i, "시 평균주문금액 :", gangnam['평균주문금액'].mean().round(-2), "원")
```

강남구 16 시 평균주문금액 : 24500.0 원

```
# 구 별로 묶기
gangnam = time_zero.groupby('구')
gangnam = gangnam.get_group('강남구')
# display(gangnam)

# 예상 주문건수 --> 지금까지 있었던 모든 날짜의 평균시간을 더한 뒤 평균 값을 전달
print("강남구", i, "시 평균배달시간 :", gangnam['평균시간'].mean().round(1), "분")
```

강남구 14 시 평균배달시간 : 23.1 분

```
# 여기에서 여성인구수나 남성 인구수 추출 // 가중치 주기 위한 --> 그때는 총인구
display(delivery_hum_EDA_20)
print("강남구 여성비중 :", delivery_hum_EDA_20['여성인구수'].values / deliv
```

C:\Users\parkgun\anaconda3\envs\pydataenv\lib\site-packages\pandas\core\fa
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs-ng-a-view-versus-a-copy>
errors=errors,

	기준연월	시	구	5세단위구분	총인구수	남성인구수	여성인구수
128675	202006	서울특별시	강남구	20-24세	33634	16342	17292

강남구 여성비중 : [0.51412261]

성별 배달주문 건수
여성 > 남성

데이터기반 정보제공 To 라이더

```
# 구 별로 묶기
gangnam = time_zero.groupby('구')
gangnam = gangnam.get_group('강남구')
# display(gangnam)

# 예상 주문건수 --> 지금까지 있었던 모든 날짜의 주문건수를 더한 뒤 평균 값을 전달
print("강남구",i, "시 예상주문건수 :",gangnam['주문건수'].mean().round(1), "건")

# 문제점 : 배달하는 라이더 수를 알아야 개인별 예상건수 예측 가능!! --> 앱을 통해 활
```

강남구 17 시 예상주문건수 : 40.3 건



**회귀분석식으로
들어갑시다**

데이터 회귀분석용 전처리

시간-업종 데이터 통합 전처리 (회귀분석용)

```
import pandas as pd

# 서울 소요시간 평균
delivery_time_mean = pd.read_csv('./data/delivery/시간-지역별 배달 소요시간평균.csv', header=None)
delivery_time_mean.columns = ['날짜', '시간', '시', '구', '평균시간']
delivery_time_mean = delivery_time_mean.groupby('시')
delivery_time_mean = delivery_time_mean.get_group('서울특별시')

# 서울 주문건수 평균
delivery_time_order_count = pd.read_csv('./data/delivery/시간-지역별 배달 주문건수.csv', header=None)
delivery_time_order_count.columns = ['날짜', '시간', '시', '구', '주문건수']
delivery_time_order_count = delivery_time_order_count.groupby('시')
delivery_time_order_count = delivery_time_order_count.get_group('서울특별시')

# 서울 주문금액 평균
delivery_time_pay_mean = pd.read_csv('./data/delivery/시간-지역별 배달 평균주문금액.csv', header=None)
delivery_time_pay_mean.columns = ['날짜', '시간', '시', '구', '평균주문금액']
delivery_time_pay_mean = delivery_time_pay_mean.groupby('시')
delivery_time_pay_mean = delivery_time_pay_mean.get_group('서울특별시')
```

```
# 일단 합쳐보자 --> 마지막에 33178 개보다 작아야 함
# 각 데이터프레임의 날짜+업종+시+구 를 unique 코드로 만든 다음 컬럼명 지정
delivery_job_order_count['식별인자'] = delivery_job_order_count['날짜_업종_시_구']
delivery_job_time_mean['식별인자'] = delivery_job_time_mean['날짜_업종_시_구']
delivery_job_pay_mean['식별인자'] = delivery_job_pay_mean['날짜_업종_시_구']
# 이것을 기준으로 merge how='inner'
df = pd.merge(delivery_job_order_count, delivery_job_time_mean, on='식별인자')
df = pd.merge(df, delivery_job_pay_mean, how='inner', on='식별인자')
# 32917 개

# 열 삭제 및 열 이름 변경
df.drop(df.columns[[5,6,7,8,9,11,12,13,14]],axis=1,inplace=True)
df.rename(columns = {'날짜_x':'날짜', '업종_x':'업종', '시_x':'시', '구_x':'구'},inplace=True)
display(df)

df.to_excel('./data/delivery/서울업종별통합파일.xlsx')
```

데이터 회귀분석용 전처리

	날짜		시간	시	구	평균시간	주문건수	평균주문금액	
0	2019-07-18	0		서울특별시	구로구	22.65	16	23679	
1	2019-07-18	0		서울특별시	동작구	22.17	1	15400	
2	2019-07-18	0		서울특별시	영등포구	20.41	14	20864	
3	2019-07-18	1		서울특별시	구로구	28.09	6	19667	
4	2019-07-18	1		서울특별시	영등포구	22.19	8	18625	
	기준연월			시	구	5세단위구분	총인구수	남성인구수	여성인구수
0	201907			서울특별시	종로구	20-24세	10947	5251	5696
1	201908			서울특별시	종로구	20-24세	10975	5272	5703
2	201909			서울특별시	종로구	20-24세	11032	5261	5771
3	201910			서울특별시	종로구	20-24세	10990	5238	5752
4	201911			서울특별시	종로구	20-24세	10950	5201	5749

서로 데이터 유형이 다름
일부 데이터를 추출해 컬럼을 만들어서
데이터를 통합하자

데이터 회귀분석용 전처리

	날짜	시간	시	구	평균시간	주문건수	평균주문금액	5세단위구분	총인구수	남성인구수	여성인구수
0	2019-07-18	0	서울특별시	구로구	22.65	16	23679	20-24세	24384	12039	12345
1	2019-07-18	1	서울특별시	구로구	28.09	6	19667	20-24세	24384	12039	12345
2	2019-07-18	9	서울특별시	구로구	20.25	3	15167	20-24세	24384	12039	12345
3	2019-07-18	10	서울특별시	구로구	21.37	21	14721	20-24세	24384	12039	12345
4	2019-07-18	11	서울특별시	구로구	22.12	73	24563	20-24세	24384	12039	12345
...
55610	2020-06-29	15	서울특별시	서대문구	123.20	1	29000	20-24세	23011	10555	12456
55611	2020-06-29	16	서울특별시	서대문구	17.39	3	18900	20-24세	23011	10555	12456
55612	2020-06-30	21	서울특별시	서대문구	91.18	1	14000	20-24세	23011	10555	12456
55613	2020-06-30	22	서울특별시	서대문구	32.67	1	13700	20-24세	23011	10555	12456
55614	2020-06-20	21	서울특별시	강서구	41.22	1	27500	20-24세	35114	16643	18471

55615 rows × 11 columns

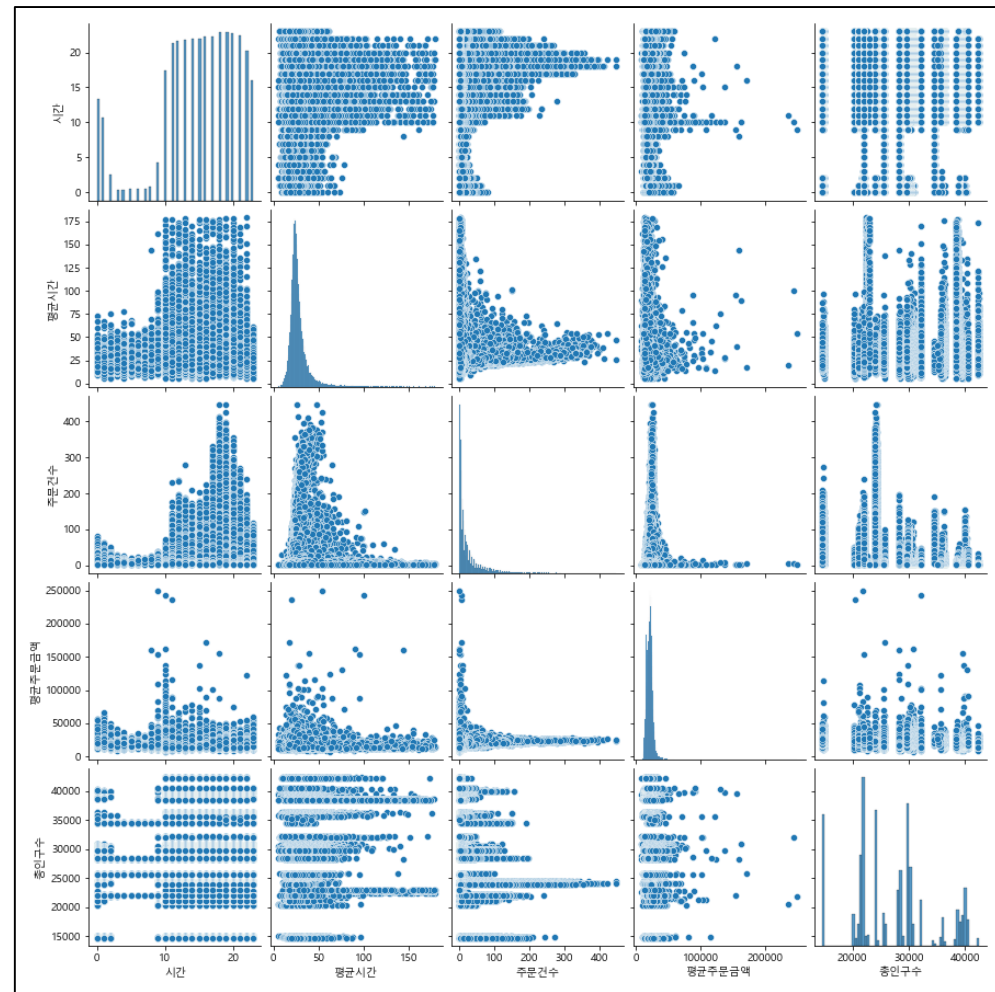
통합한 데이터 : 55615개

데이터 회귀분석용 전처리

	시간	평균시간	주문건수	평균주문금액	총인구수
0	0	22.65	16	23679	24384
1	1	28.09	6	19667	24384
2	9	20.25	3	15167	24384
3	10	21.37	21	14721	24384
4	11	22.12	73	24563	24384
(55615, 5)					

회귀분석에 필요한 데이터만 추출
55615개 데이터, 5개의 컬럼

데이터 시각화



데이터 분포 파악
여기서부터 망한 거 같다

데이터 회귀분석

	시간	평균시간	주문건수	평균주문금액	총인구수
0	0	22.65	16	23679	24384
1	1	28.09	6	19667	24384
2	9	20.25	3	15167	24384
3	10	21.37	21	14721	24384
4	11	22.12	73	24563	24384
(55615, 5)					

OLS Regression Results			
Dep. Variable:	주문건수	R-squared:	0.169
Model:	OLS	Adj. R-squared:	0.169
Method:	Least Squares	F-statistic:	1978.
Date:	Wed, 25 Nov 2020	Prob (F-statistic):	0.00
Time:	13:42:16	Log-Likelihood:	-2.0060e+05
No. Observations:	38930	AIC:	4.012e+05
Df Residuals:	38925	BIC:	4.012e+05
Df Model:	4		
Covariance Type:	nonrobust		

Raw data로 회귀분석 돌리기
R-squared 값이 0.17로 매우 구린 모형이다.
(R-squared는 신뢰도를 의미,
1에 가까울수록 신뢰도가 높은 것)

데이터 회귀분석

```
# 피쳐 각각에 대한 scaling을 수행하는 함수를 정의  
# 표준편차와 평균을 이용해 스케일링 (Z score)
```

```
def standard_scaling(df, scale_columns):  
    for col in scale_columns:  
        series_mean = df[col].mean()  
        series_std = df[col].std()  
        df[col] = df[col].apply(lambda x: (x-series_mean)/series_std)  
    return df
```

```
# 시간 피쳐를 one-hot encoding으로 변환 --> 범주형이기 때문  
time_encoding = pd.get_dummies(re_df['시간'])  
re_df = re_df.drop(re_df.columns[0], axis=1)  
re_df = re_df.join(time_encoding) # 합치기!
```

	평균시간	y	평균주문금액	총인구수	0	1	2	3	4	5	...	14	15	16	17	18	19	20	21	22	23
0	-0.399011	16	0.399378	-0.37888	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	-0.027347	6	-0.334507	-0.37888	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	-0.562980	3	-1.157660	-0.37888	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	-0.486461	21	-1.239243	-0.37888	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	-0.435221	73	0.561082	-0.37888	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

데이터 표준화를 해봅시다!!
시간은 카테고리로 one-hot-encoding

데이터 회귀분석 결과

```
# 회귀 분석 모델을 평가
print(model.score(X_train, y_train)) # train R2 score를 출력합니다.
print(model.score(X_test, y_test)) # test R2 score를 출력합니다.
```

```
0.2583872015758766
0.24870539112728418
```

```
# 회귀 분석 모델을 평가
y_predictions = lr.predict(X_train)
print(sqrt(mean_squared_error(y_train, y_predictions))) # train RMSE score를 출력합니다.
y_predictions = lr.predict(X_test)
print(sqrt(mean_squared_error(y_test, y_predictions))) # test RMSE score를 출력합니다.
```

```
# 차이의 제곱 값이 RMSE
```

```
39.3374843399196
39.9097209526107
```

하지만 여전히 구림
R-squared 값이 0.25로 나온다
(보통 0.5가 넘어야 씀)
변수 설정을 잘못된 것 같음
처음부터 다시!!



**2020.11.27 기준
빅데이터 담당업무 수행현황임**

궁금한 거 있으면 물어봐!!