



코로나 시대의 음식 배달 문화 분석

멀티캠퍼스

빅데이터 기반 지능형 서비스 개발

2차 세미프로젝트

최종보고서



노청명, 이재웅 두 명이 힘을 합쳐 빅데이터 분석을 통해 세상을 이롭게 하는 인사이트를 발견한다는 의미입니다.

“Contents”

01 프로젝트 배경

- 프로젝트 배경
- 분석 목표
- 개발환경 및 도구

02 데이터 분석

- 데이터 수집
- 전처리 및 EDA
- 시계열분석
- 군집분석

03 웹 서비스

- 시각화
- 웹 서버
- 데이터베이스

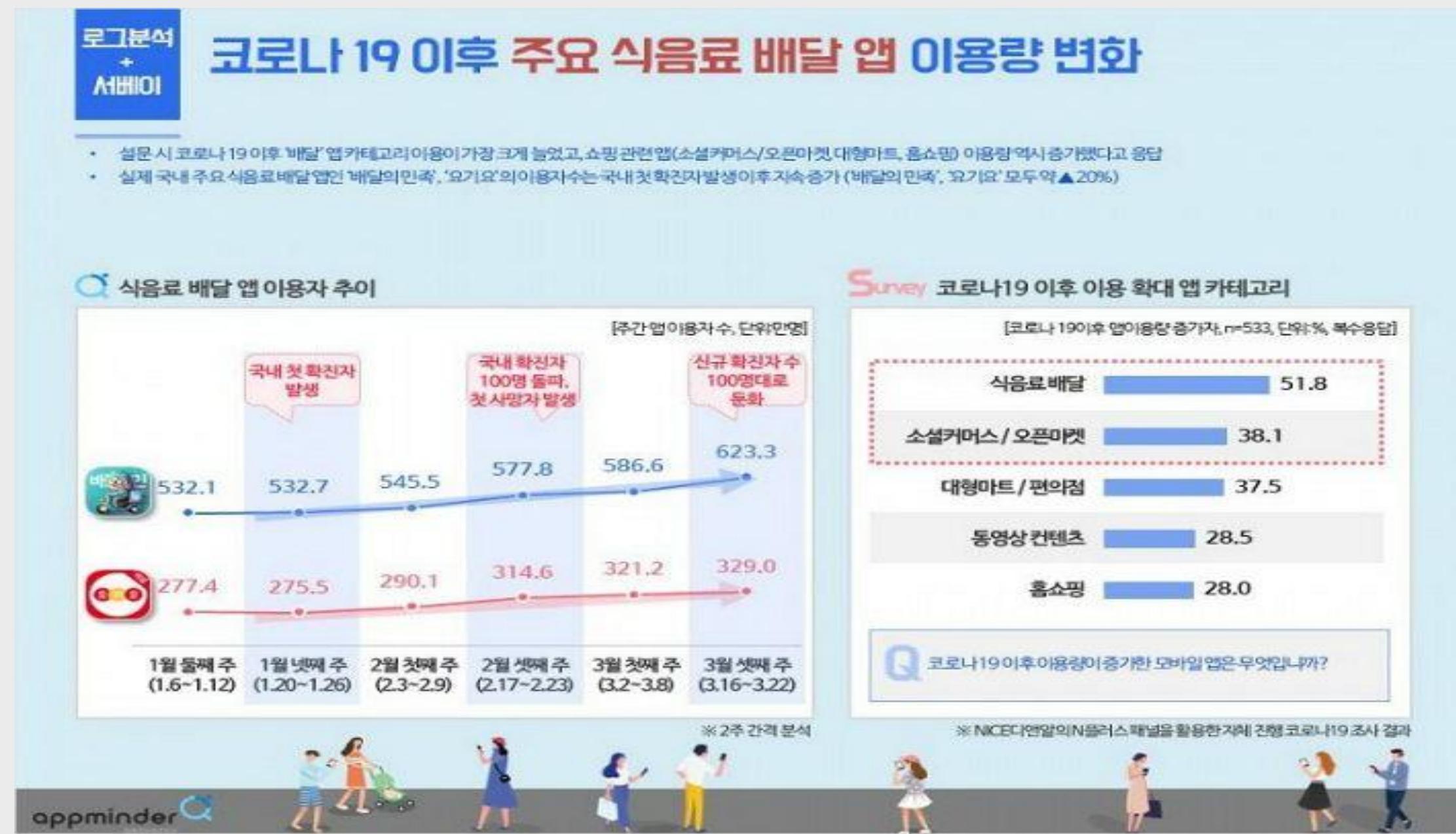
04 결론

- 수행 결과
- 기대효과
- 개발 후기

프로젝트 개요

01

코로나 시대에 급속히 성장하는 배달음식 시장



그렇다면, 자영업자들은?

자영업 배달 플랫폼 '양날의 검'

골목상권 '슈퍼갑' 떠오른 배달앱...수수료에 허덕이는 자영업

코로나19 생존 위해 택했던 '비대면 배달앱', 자영업자 영업이익 개선에 큰 효과 없어

고로나가 열어젖힌 배달·비대면 전성시대, 자영업 깊은 한숨

주문 늘수록 괴로운 자영업자들...배달 앱의 두 얼굴

“

코로나 확진자 추이와 음식 배달 주문 데이터를 분석하여 요식업 (예비)창업자에게 도움되는 정보를 제공

”



입지 선정



업종 선정



원가 절감



수익 극대화

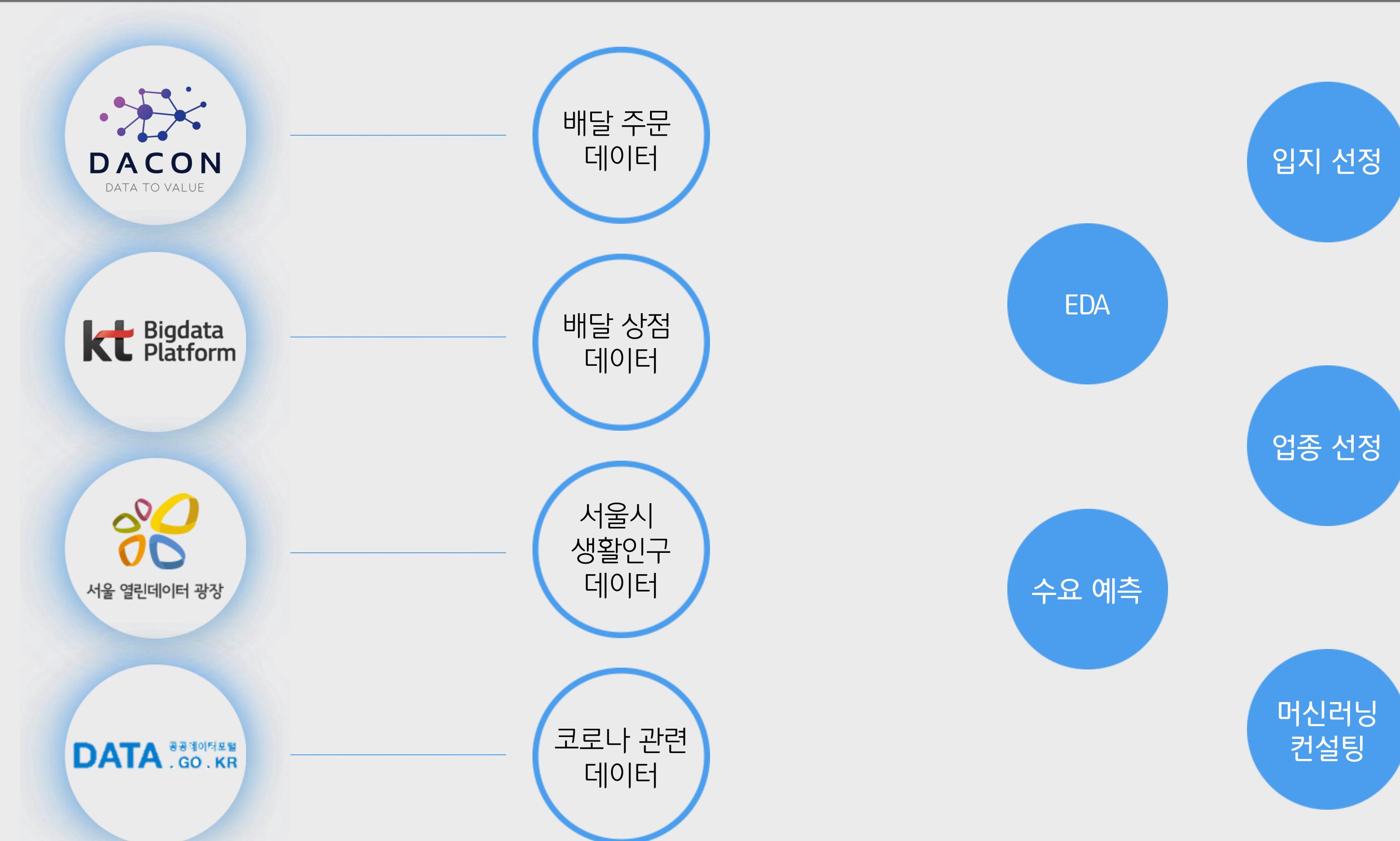
개발환경 및 도구

개발환경 및 도구	
언어	Python, SQL
통합 개발 환경	Pycharm, Jupyter Notebook, Colab
분석 라이브러리	Pandas, Numpy, Scikit-Learn, Keras, TensorFlow, FBprophet
시각화	Tableau, Matplotlib
웹 서버	Django
웹 클라이언트	HTML5, JS, Ajax, CSS
시스템 운영 환경	Python Anywhere
데이터베이스	MariaDB
협업 도구	Github, Zoom, Google Docs

데이터 분석

02

데이터 수집



코로나에 따른 주문현황 변화

1. 코로나 월별 누적 확진자수 데이터 추출
2. 월별 배달 주문 건수 추출
3. 두 데이터 병합 후 결측치 처리

시간대별 주문현황

1. 배달 주문 데이터 컬럼 추출
2. 배달 주문 데이터 시간대별 분류

시군구별 인구와 배달 주문

1. 월별 생활인구 데이터 병합
2. 성별, 연령별 생활인구 산출
3. 날짜와 행정동 코드 두 가지 조건을 기준으로 배달 주문 데이터와 병합

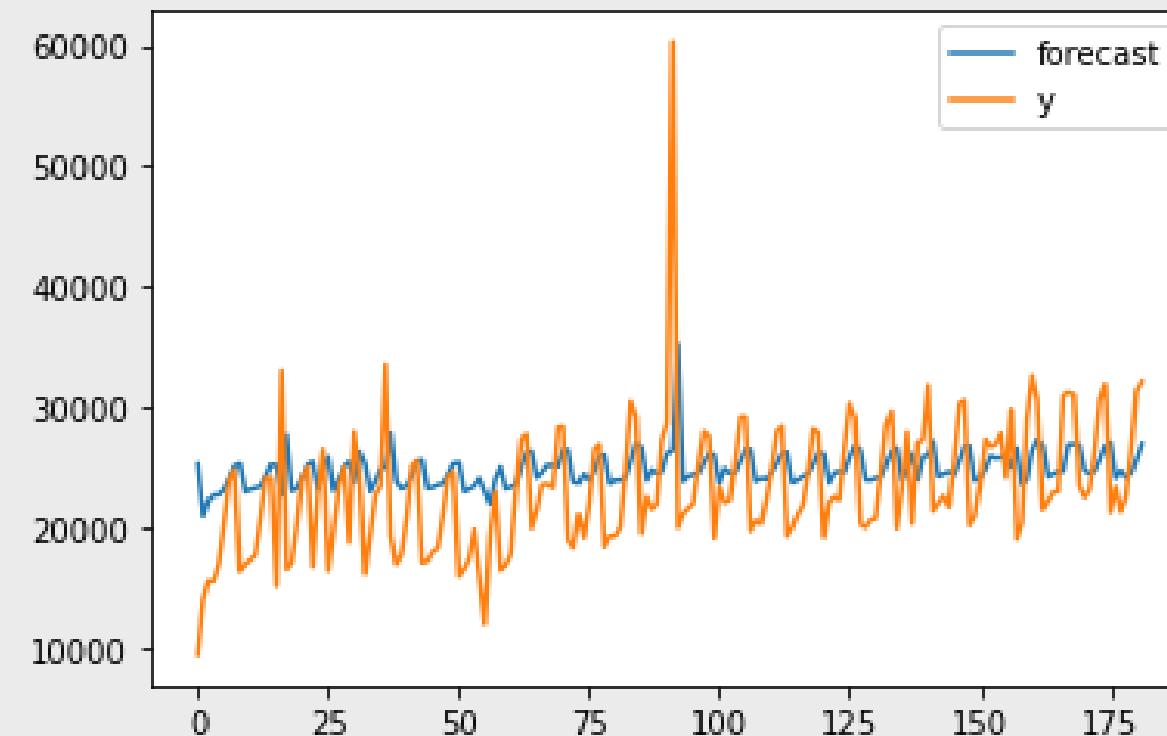
배달 주문과 배달 상점

1. 행정동 코드를 활용하여 배달 상점 데이터 선별
2. 주문 데이터와 상점 데이터 릴레이션 형성
3. 매출 금액 정규화
4. 함수를 활용하여 매출 금액 구간별 구분
5. 상점 수(공급) 대비 배달 건수(수요)를 점수로 표시

전처리

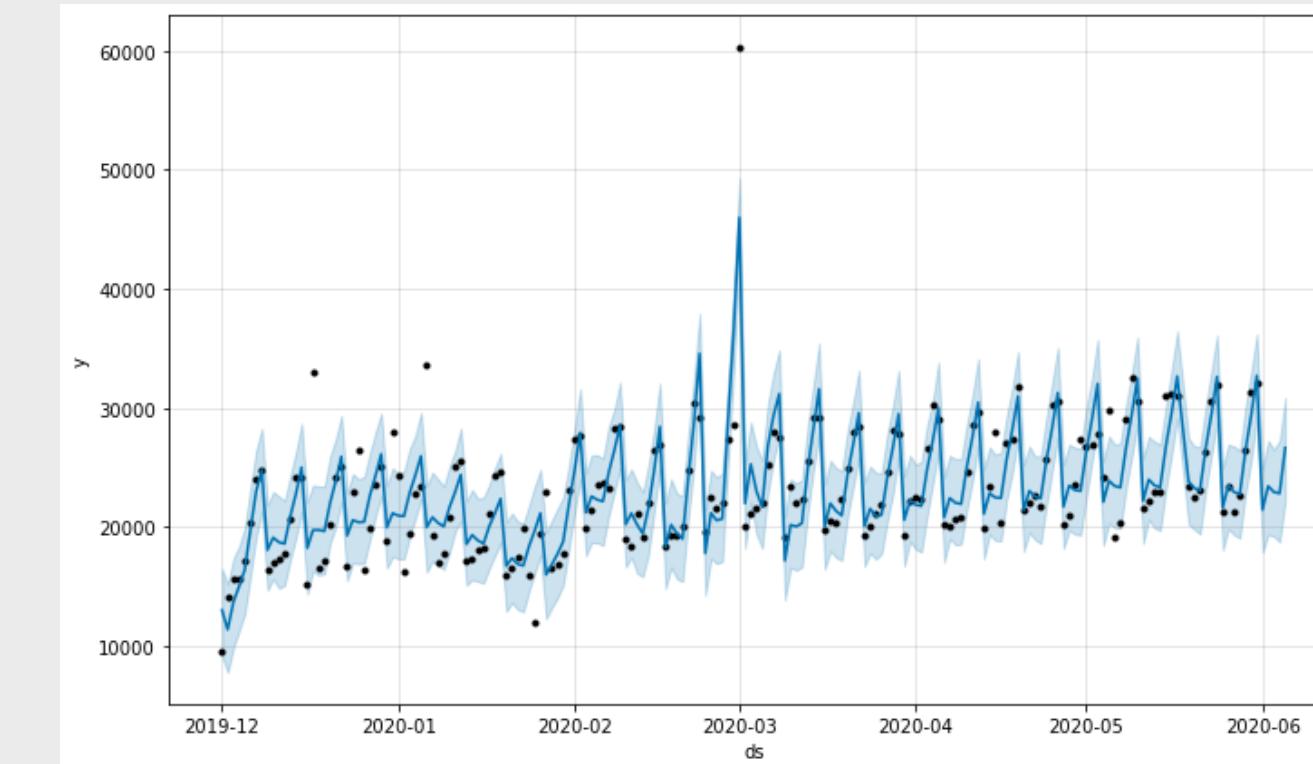
1. 배달 주문 시간, 배달 주문 상태 컬럼 선택
2. 배달 주문 상태가 완료인 행 추출
3. 주문 시간 결측치 행 삭제
4. 주문 시간 시계열 데이터로 변환
5. 날짜별 주문 건수 합계 산출

모델링



Statsmodels ARIMA

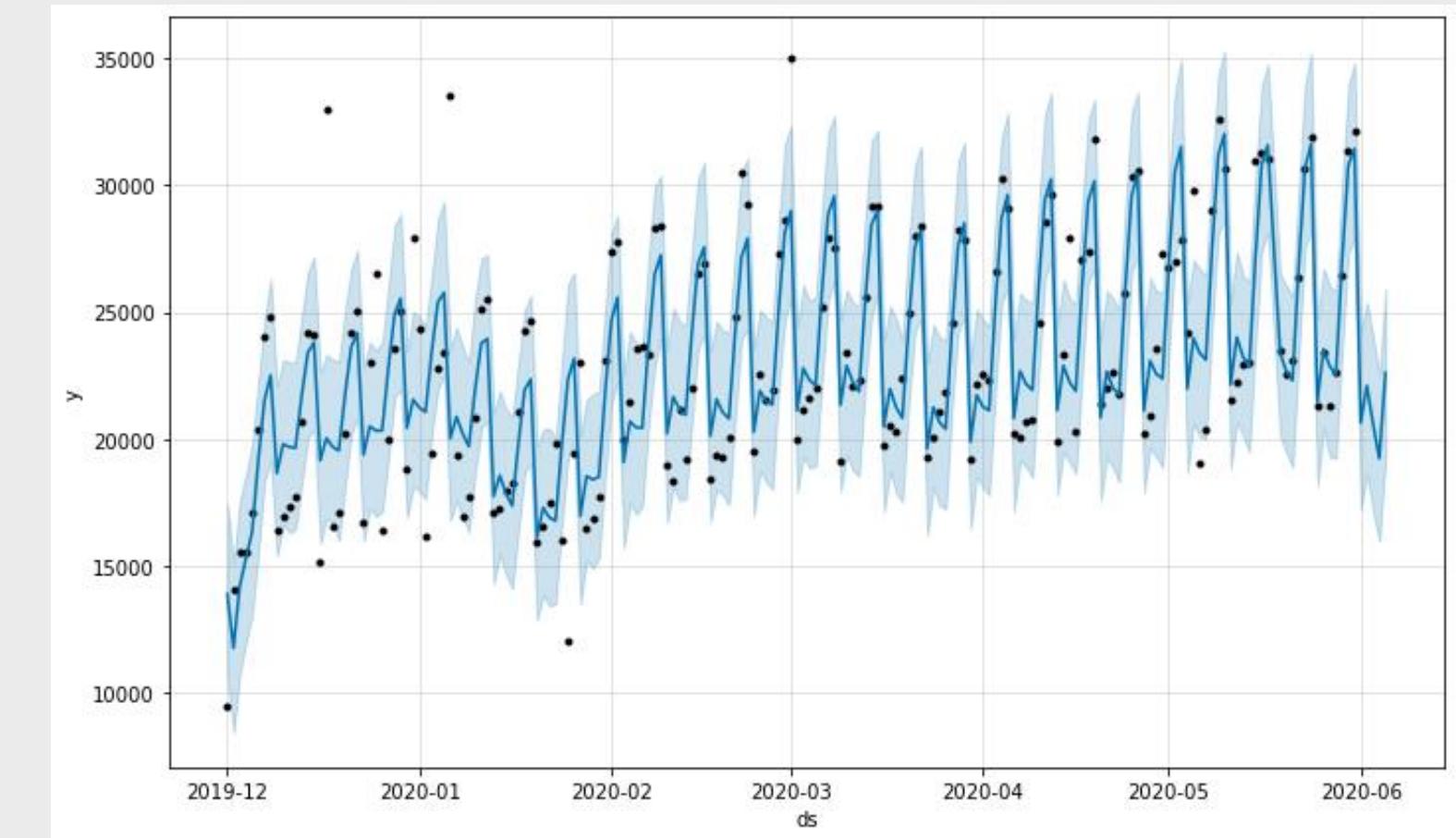
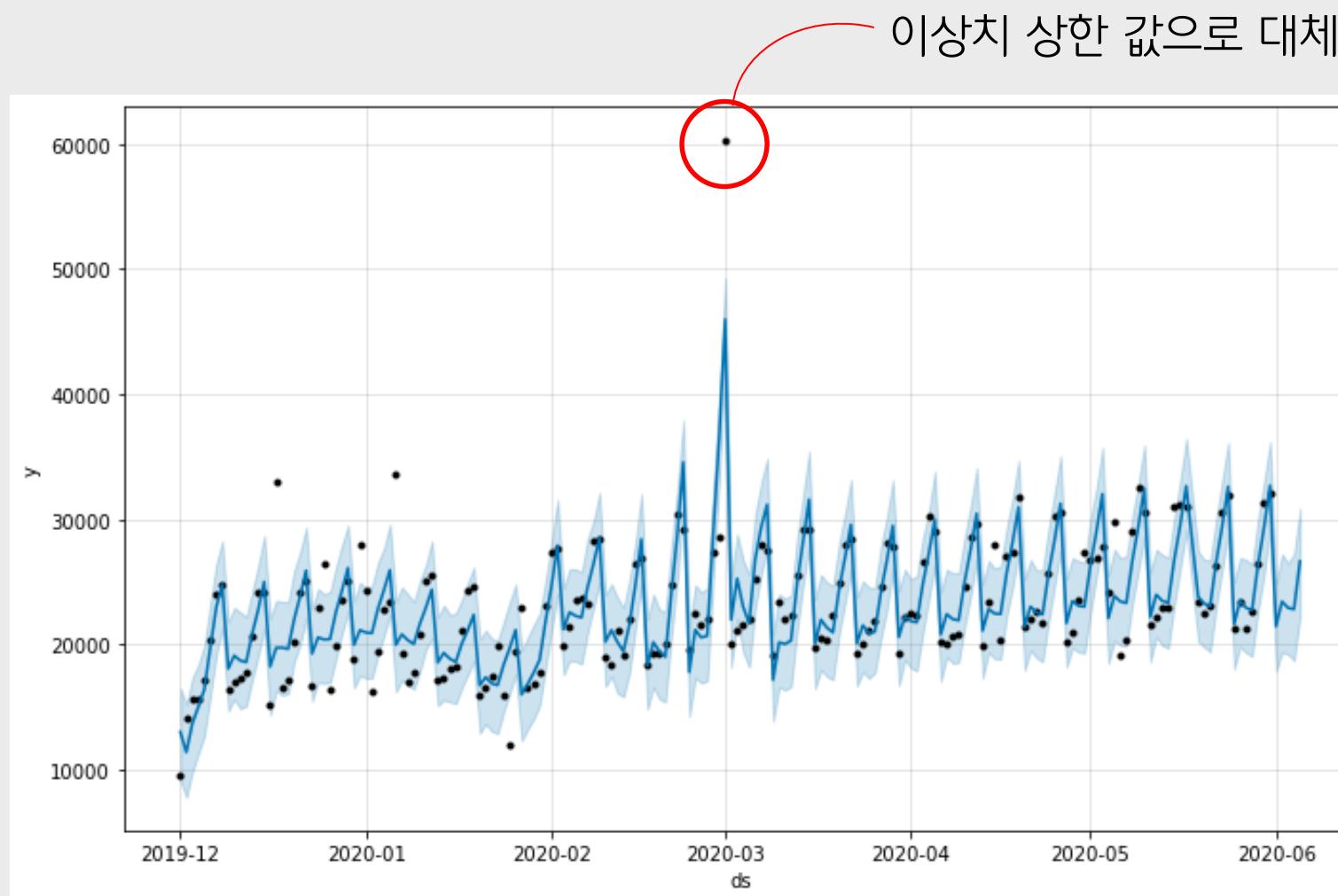
VS



Facebook fbprophet

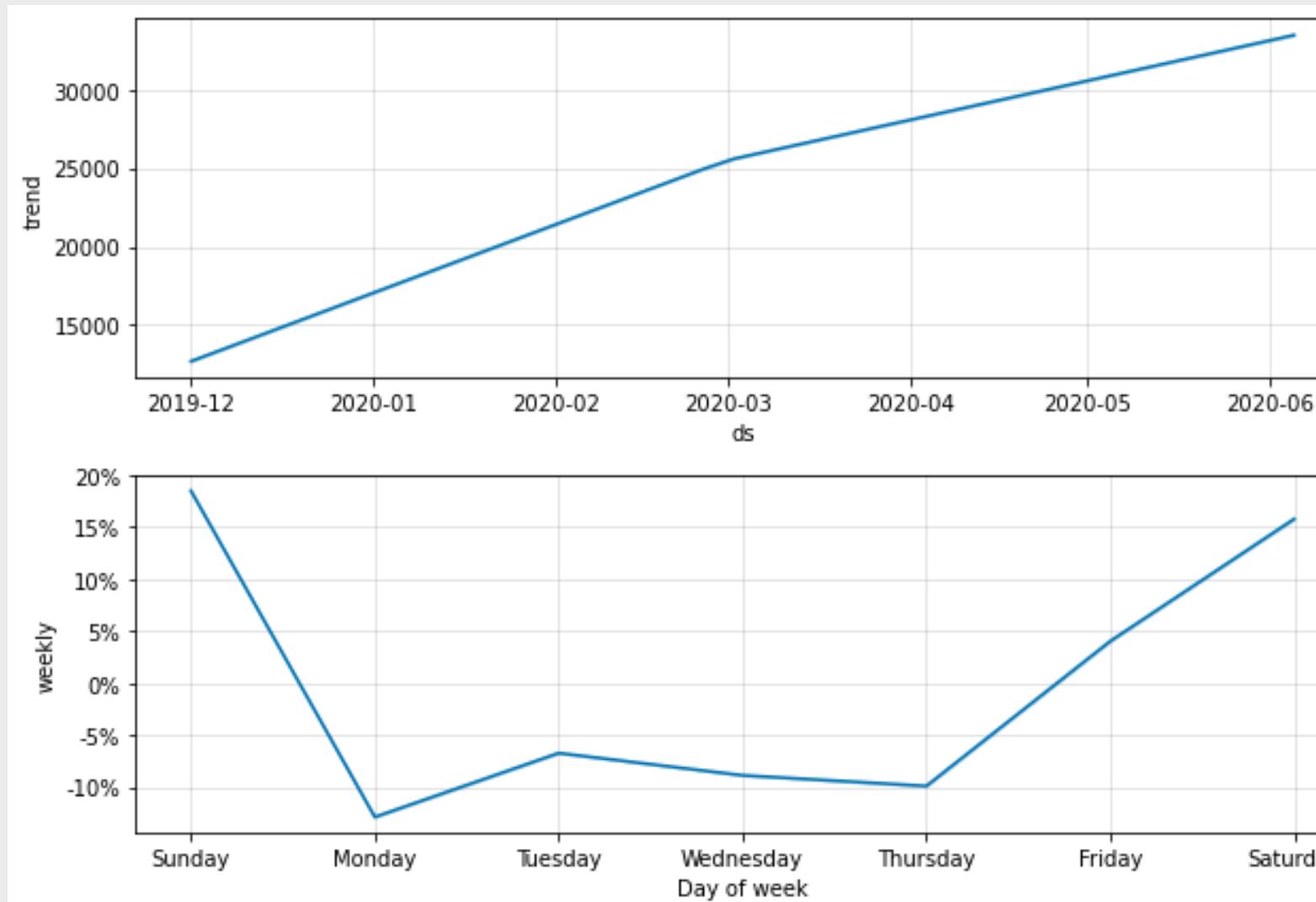
시계열분석

모델링

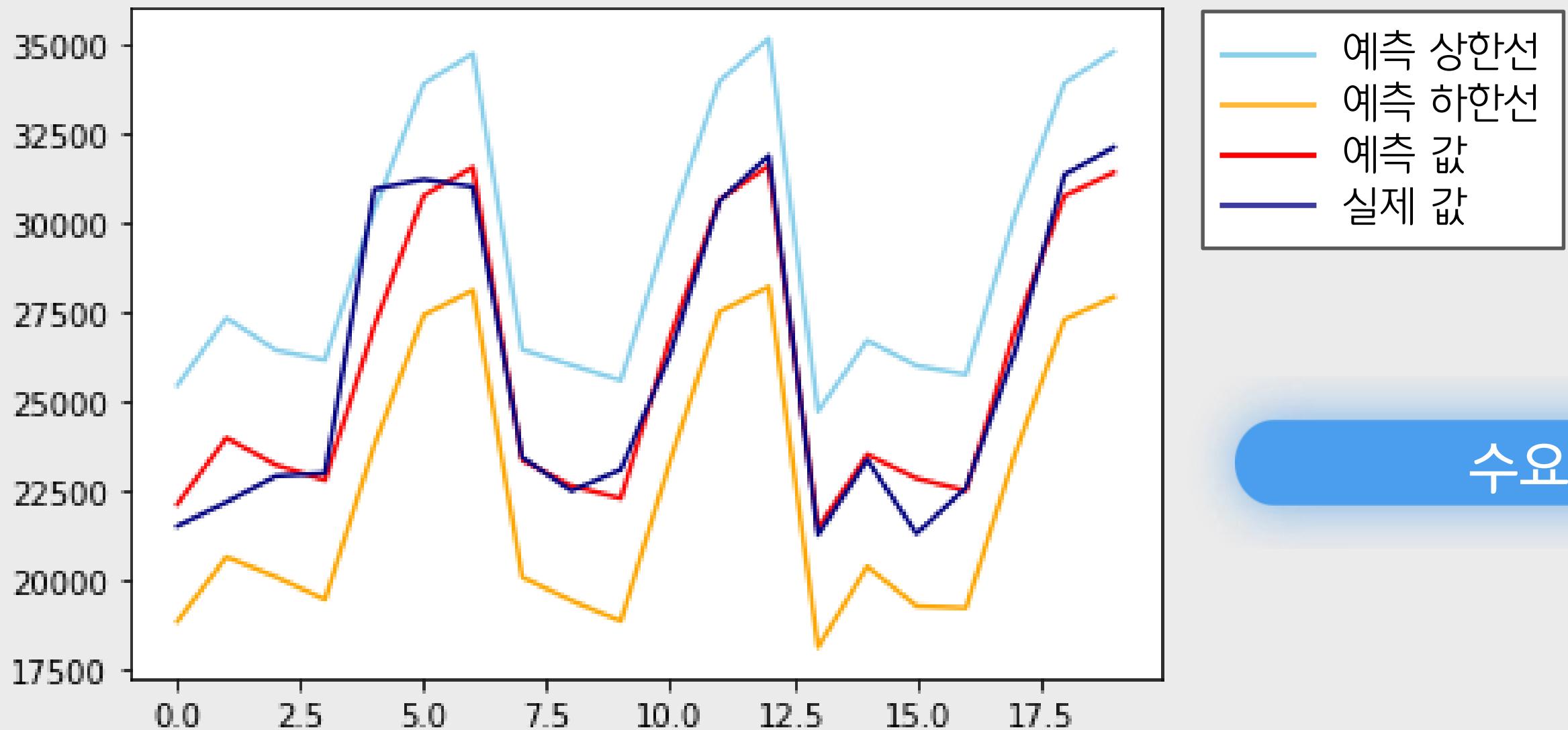


시계열분석

모델 확인



예측



수요 예측 서비스에 활용

전처리

1. 컬럼 선택 및 추출
2. 배달접수시간, 배달수령시간 결측치 행 삭제
3. 주문 처리 시간 산출(배달수령시간 – 배달접수시간)
4. 주문 처리 시간을 연속형 변수로 활용하기 위해 초단위로 변경
5. 일반적 최소주문금액(5000원) 이하의 주문상품금액 값을 가진 행 제거
6. 각 컬럼의 상점별 합계 또는 평균 산출
7. 모든 연속형 변수 정수형으로 변환
8. 범주형 데이터(행정동 코드, 업종명, 읍면동)와 병합

모델링

1. 원핫인코딩을 통해 더미변수 생성
2. 정규화
3. 모델 생성
4. 모델 적합
5. 예측 모델 피클 생성(웹 서비스에 활용)

```
# 원핫인코딩(더미 변수)
label_encoder = preprocessing.LabelEncoder()
onehot_encoder = preprocessing.OneHotEncoder()

onehot_hjdCode = label_encoder.fit_transform(dlvr_fin_kp['DLVR_STORE_ADSTRD_CODE'])
onehot_inIndustry = label_encoder.fit_transform(dlvr_fin_kp['DLVR_STORE_INDUTY_NM'])
onehot_bjdCode = label_encoder.fit_transform(dlvr_fin_kp['DLVR_STORE_LEGALDONG_NM'])

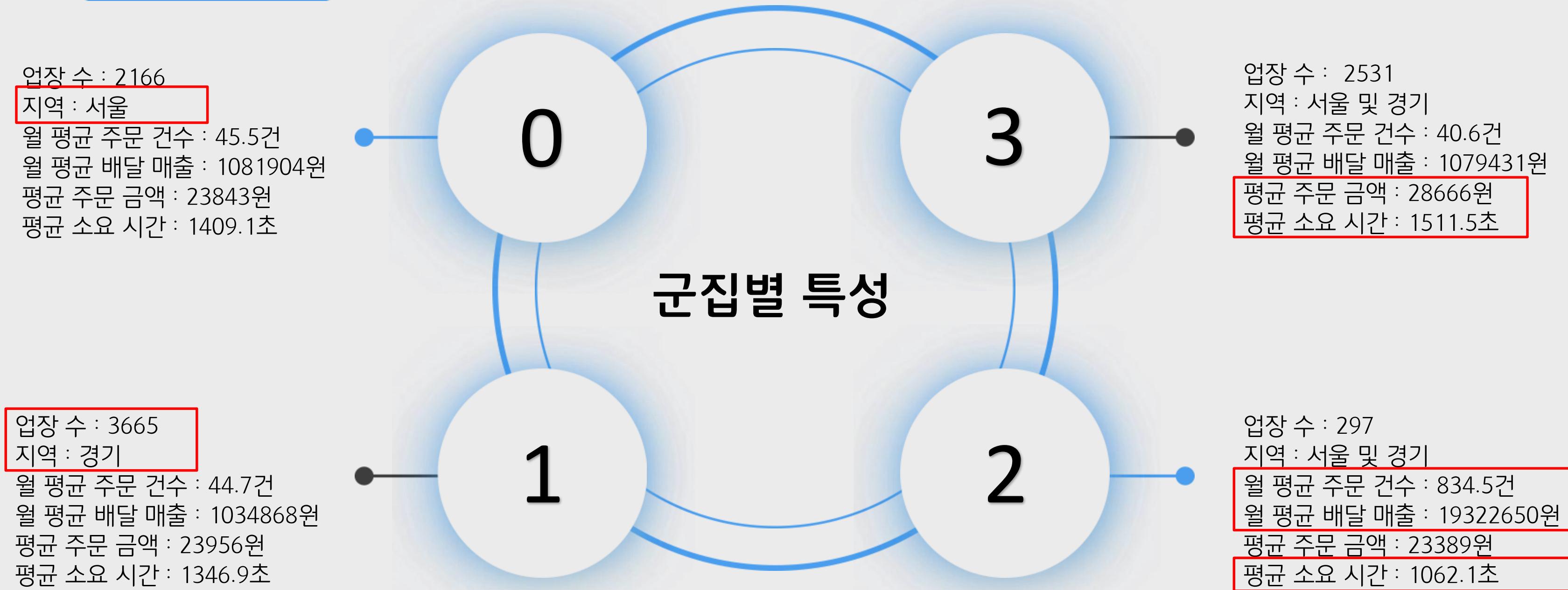
dlvr_fin_kp['DLVR_STORE_ADSTRD_CODE'] = onehot_hjdCode
dlvr_fin_kp['DLVR_STORE_INDUTY_NM'] = onehot_inIndustry
dlvr_fin_kp['DLVR_STORE_LEGALDONG_NM'] = onehot_bjdCode
```

```
# 정규화
dlvr_fin_kp = preprocessing.StandardScaler().fit(dlvr_fin_kp).transform(dlvr_fin_kp)

kp = KPrototypes(n_clusters=4, init='Huang', n_init=10, max_iter=30, verbose=True)
```

```
kp_dlvr = kp.fit_predict(dlvr_fin_kp, categorical=[3])
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run: 1, iteration: 1/30, moves: 2359, ncost: 45615.680857584404
Run: 1, iteration: 2/30, moves: 1387, ncost: 44849.57524406394
Run: 1, iteration: 3/30, moves: 643, ncost: 44570.591191517575
Run: 1, iteration: 4/30, moves: 459, ncost: 44426.48261500759
Run: 1, iteration: 5/30, moves: 280, ncost: 44368.11522918309
Run: 1, iteration: 6/30, moves: 173, ncost: 44343.5734216796
Run: 1, iteration: 7/30, moves: 56, ncost: 44340.23543478895
Run: 1, iteration: 8/30, moves: 23, ncost: 44336.82072364307
Run: 1, iteration: 9/30, moves: 5, ncost: 44336.79950878276
Run: 1, iteration: 10/30, moves: 1, ncost: 44336.796901475805
Run: 1, iteration: 11/30, moves: 0, ncost: 44336.796901475805
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run: 2, iteration: 1/30, moves: 1574, ncost: 46321.04632550257
Run: 2, iteration: 2/30, moves: 742, ncost: 45005.57520400040
```

군집 해석

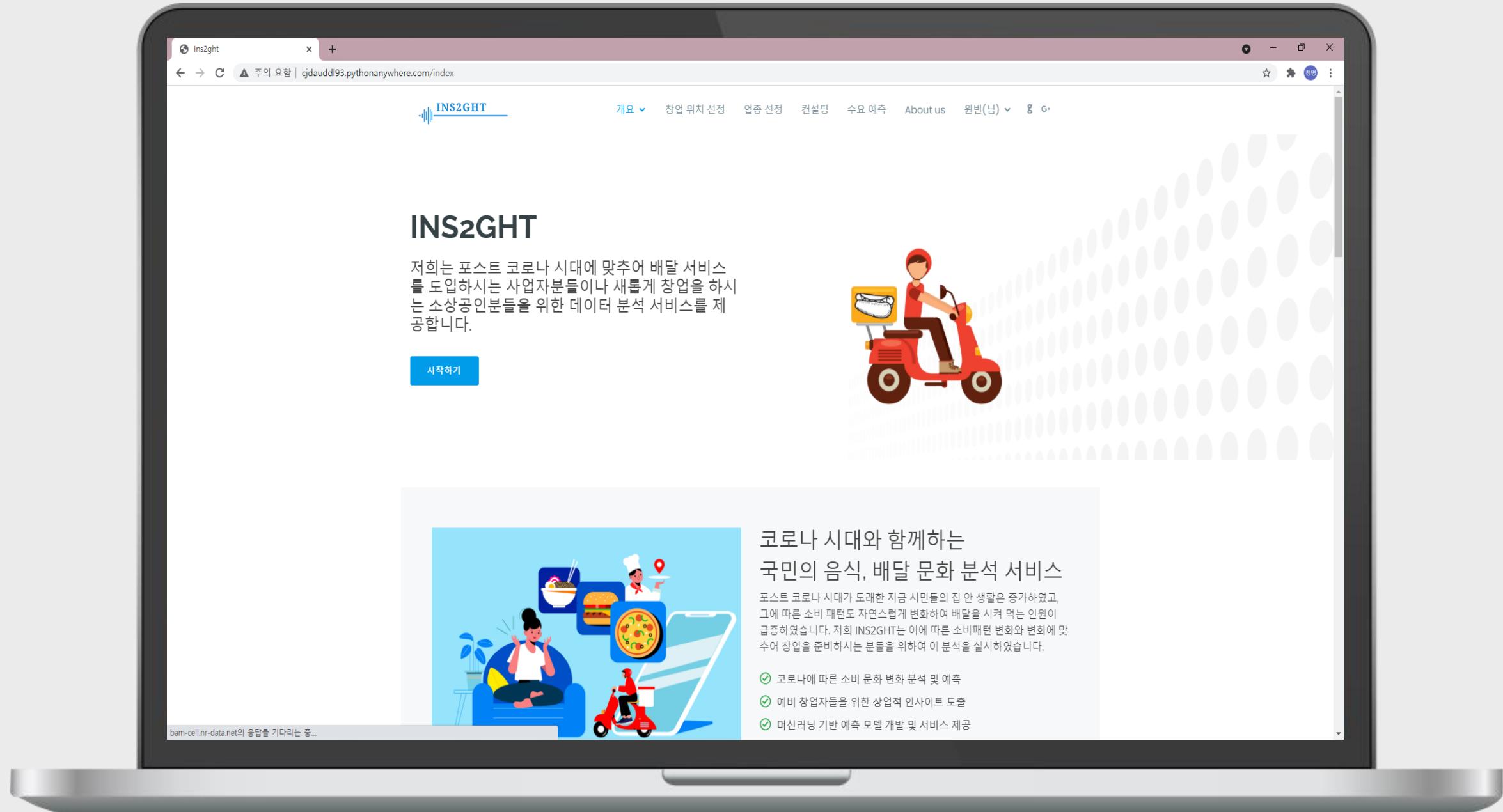


웹 서비스

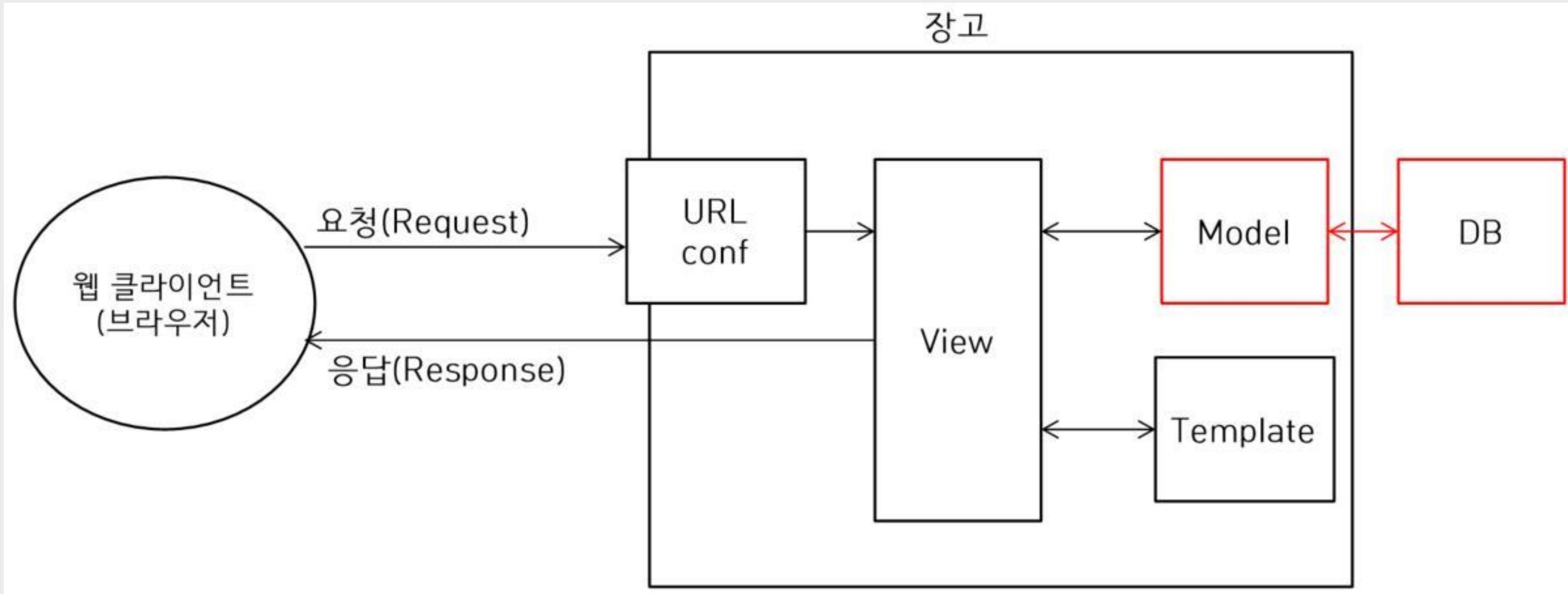
03



시각화

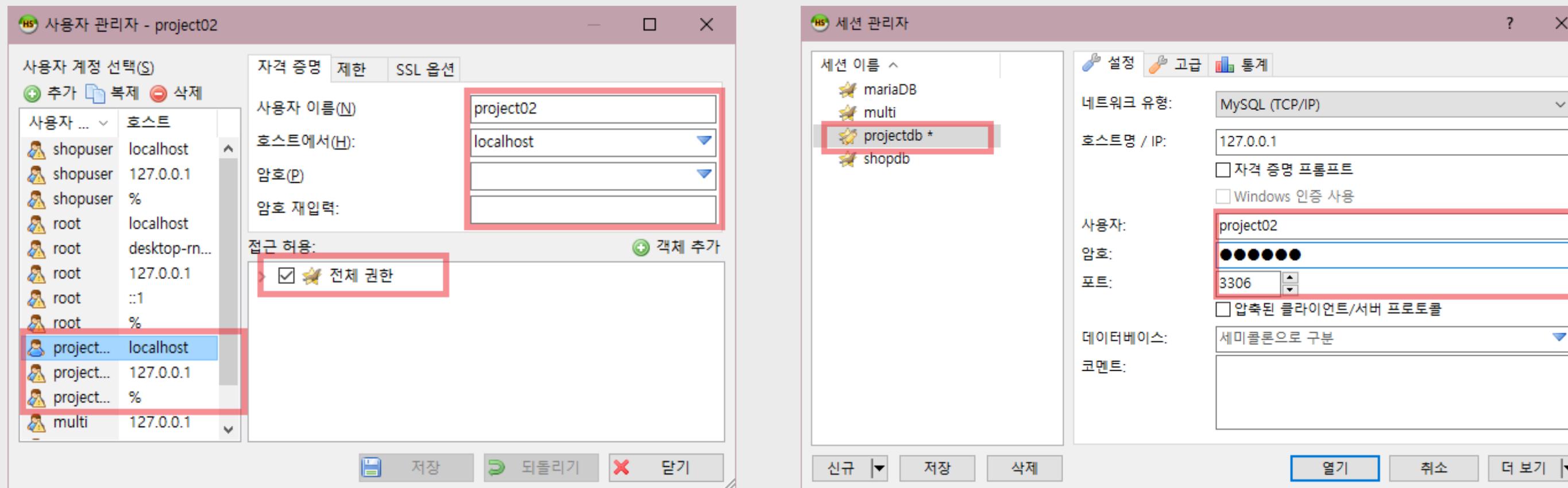


MVT 패턴



데이터베이스

MariaDB 세팅



DB 모델 구축

The screenshot shows the HeidiSQL interface with the following details:

- WindowTitle:** projectdb@userdb - HeidiSQL 9.4.0.5125
- Toolbar:** Includes standard database management icons.
- Host:** 127.0.0.1
- Database:** userdb
- Query Tab:** query7.sql
- Schemas:** A tree view shows the schema structure under projectdb, including databases like employees, information_schema, mysql, performance_schema, shop, shopdb, test, and userdb. The userdb schema is selected.
- Table List:** Under userdb, there is one table named user.
- Code Editor:** The main pane displays the following SQL code:

```
1 DROP TABLE user;
2
3 CREATE TABLE user(
4     id VARCHAR(10),
5     pwd VARCHAR(10) NOT NULL,
6     name VARCHAR(10) NOT NULL,
7     imgname VARCHAR(20),
8     email VARCHAR(30) NOT NULL,
9     regdate DATE NOT NULL
10 );
11 ALTER TABLE user ADD PRIMARY KEY (id);
12 INSERT INTO user VALUES('test01','pwd01','노 청 명 ',null,'test01@gmail.com',CURRENT_DATE());
13 INSERT INTO user VALUES('test02','pwd02','원 빙 ',null,'test02@gmail.com',CURRENT_DATE());
14 INSERT INTO user VALUES('test03','pwd03','장 동 건 ',null,'test03@gmail.com',CURRENT_DATE());
15
```
- Right Panel:** A sidebar with various navigation options such as 열 (Tables), SQL 함수 (SQL Functions), SQL 키워드 (SQL Keywords), and 쿼리 내역 (Query History).

DB 서버 연결

```
# 로컬용
config = {
    'database': 'userdb',
    'user': 'project02',
    'password': [REDACTED],
    'host': '127.0.0.1',
    'port': 3306,
    'charset': 'utf8',
    'use_unicode': True
}
class Db:
    def getConnection(self):
        conn = pymysql.connect(**config)
        return conn

    def close(self, conn, cursor):
        if cursor != None:
            cursor.close()
        if conn != None:
            conn.close()
```

```
# 배포용
config = {
    'database': 'cjdauddl93$user',
    'user': 'cjdauddl93',
    'password': [REDACTED],
    'host': 'cjdauddl93.mysql.pythonanywhere-services.com',
    'port': 3306,
    'charset': 'utf8',
    'use_unicode': True
}
```

Python 내부 db.py를 통해 서버와 연결

데이터베이스

Class 생성

```
class Sql:  
    userlist = "SELECT * FROM user";  
    userlistone = "SELECT * FROM user WHERE id= '%s' ";  
    userinsert = "INSERT INTO user VALUES ('%s', '%s', '%s', '%s', '%s', CURRENT_DATE())";  
    userdelete = "DELETE FROM user WHERE id= '%s' ";  
    userupdate = "UPDATE user SET pwd='%s', name='%s', imgname='%s', email='%s' WHERE id= '%s' ";
```

Sql 클래스

```
class ErrorCode:  
    e0001 = '아이디가 중복 되었습니다. 다시 입력 하세요.';  
    e0002 = '삭제 중 오류.';  
    e0003 = 'ID 또는 Password가 틀렸습니다.';  
    e0004 = '공란을 모두 채워 주세요.';
```

ErrorCode 클래스

```
class User:  
    def __init__(self, id, pwd, name, imgname, email, regdate):  
        self.id = id;  
        self.pwd = pwd;  
        self.name = name;  
        self.imgname = imgname;  
        self.email = email;  
        self.regdate = regdate;  
    def __str__(self):  
        return self.id+' '+self.pwd+' '+self.name+' '+str(self.imgname)+ ' '+self.email+' '+ str(self.regdate);
```

User 클래스

함수 정의

```
class UserDB(Db):
    def update(self, id, pwd, name, imgname, email):
        try:
            conn = super().getConnection();
            cursor = conn.cursor();
            cursor.execute(Sql.userupdate % (pwd, name, imgname, email, id));
            conn.commit();
        except:
            conn.rollback();
            raise Exception;
            print(ErrorCode.e0002)
        finally:
            super().close(conn, cursor);
    def delete(self, id):
        try:
            conn = super().getConnection();
            cursor = conn.cursor();
            cursor.execute(Sql.userdelete % id);
            conn.commit();
        except:
            conn.rollback();
            raise Exception;
            print(ErrorCode.e0002)
        finally:
            super().close(cursor, conn)
```

회원가입, 탈퇴, 로그인, 프로필 수정 함수를 정의

DB 웹 구현

```
<form action="loginimpl" method="post">
  {% csrf_token %}
  <div class="form-group first">
    <label for="username">Username</label>
    <input type="text" class="form-control" name="id" id="username">
  </div>
  <div class="form-group last mb-4">
    <label for="password">Password</label>
    <input type="password" class="form-control" name="pwd" id="password">
  </div>

  <div class="d-flex mb-5 align-items-center">
    <label class="control control--checkbox mb-0"><span class="caption">Remember me</span>
      <input type="checkbox" checked="checked"/>
      <div class="control__indicator"></div>
    </label>
    <span class="ml-auto"><a href="#" class="forgot-pass">Forgot Password</a></span>
  </div>
  <div>
    {{msg}}
  </div>
  <input type="submit" value="Log In" class="btn btn-block btn-dark">
  <div id="box1"><span class="d-block text-left my-4 text-muted"> 아직 회원이 아니신가요? </span>
  </div>
</form>
```

```
def loginimpl(request):
    id = request.POST['id'];
    pwd = request.POST['pwd'];
    next = 'index.html'
    try:
        user = UserDB().selectone(id);
        if pwd == user.pwd:
            request.session['suser'] = {'id': user.id, 'name': user.name};
            context = {
                'loginuser': user
            };
        else:
            raise Exception();
    except:
        next = 'login.html'
        context = {
            'msg': ErrorCode.e0003
        };
    return render(request, next, context);
```

결론

04



EDA

시계열분석

군집분석

- 코로나 감염자 수와 배달 주문의 양의 상관관계
- 시간대별, 지역별 상이한 배달 문화 변화

- 요일, 계절에 따른 추세
- ARIMA보다 간편하고 높은 적합도를 보이는 FBProphet

- 소수의 업체가 시장의 대부분을 차지
- 더 많은 특성 데이터의 필요성

01

포스트 코로나 시대에 예비 창업자들을 위한 창업 가이드 제시

02

기창업자 소상공인들을 위한 방향 제시

03

머신러닝 및 데이터 분석, 데이터 시각화에 참고자료로 활용

개발후기



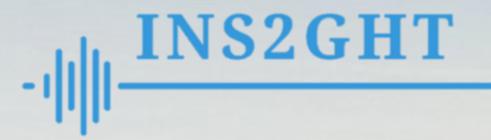
노청명

처음에는 다른 조들과 다르게 4명이 아닌 2명으로 프로젝트를 시작한다는 게 부담스럽기도 했습니다. 하지만 저희 둘 다 팀원이 적은 만큼 더 노력해야 한다는 마음가짐을 가지고 열심히 노력하고 공부했습니다. 그 결과 실력도 확실히 늘었고 저희 스스로 만족할 만한 결과물이 된 것 같아 기분이 좋습니다. 또한 “두 명에서도 충분히 할 수 있다.”라는 자신감도 얻을 수 있었습니다. 함께 열심히 노력해준 팀원 이재웅 님께 다시 한번 감사하다고 말하고 싶고, 저희의 프로젝트가 코로나로 고생하시는 소상공인분들에게 조금이나마 도움이 되었으면 좋겠습니다.



이재웅

이번 프로젝트는 저에게 많은 것을 배우고, 성장할 수 있는 기회였습니다. 다양한 분석 라이브러리를 활용해보고 새로운 시각화 툴을 접하게 되었습니다. 두 명이 프로젝트의 전체 과정을 진행해야 하는 상황이 조금은 부담스러웠지만, 좋은 실력을 가진 노청명님 덕분에 수월하게 진행할 수 있었고 만족스러운 결과를 얻었습니다. 또한 프로젝트를 진행하며 요식업 소상공인 대다수가 상당한 어려움을 겪고 있다는 사실을 체감할 수 있었습니다. 저희의 프로젝트가 소상공인분들에게 조금이나마 도움이 되었으면 합니다. 앞으로도 데이터 분석을 통해 더 많은 사람에게 유익한 가치를 제공하겠습니다.



발표를 들어주셔서 감사합니다

Thank you