

**사내식당
식수 인원
예측 프로그램**

Forecast on Data Program

2022.01.20
Team 3

<https://github.com/oblige/Team3/>

Team 3

정재훈

데이터 전처리
R 통계 EDA
SQL DB 생성
머신러닝 모델링
코드 모듈화
Github 관리
PPT 제작

박지윤

데이터 확보
데이터 전처리
SQL EDA
SQL 쿼리문, 코드 정리
머신러닝 모델링
딥러닝 모델링

변주섭

데이터 확보
데이터 전처리
SQL EDA
머신러닝 모델링
발표

이행철

데이터 확보
데이터 전처리
SQL EDA
머신러닝 모델링
발표 준비

Contents

1.

Project Outline

2.

Analysis

3.

Module Structure

4.

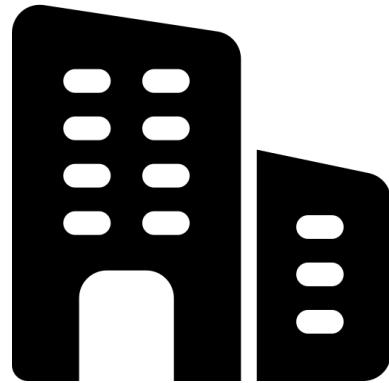
Mission

1. Project Outline

사내식당



급식업체

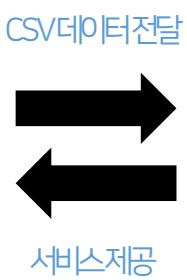


데이터전달



식수인원예측서비스제공

사내식당



급식업체



1. Python에서 전처리

2. SQLite로 DB에 저장

3. EDA 및 데이터 시각화

4. 통계보고서 작성

5. 예측모델 통한 인원수 예측

CSV

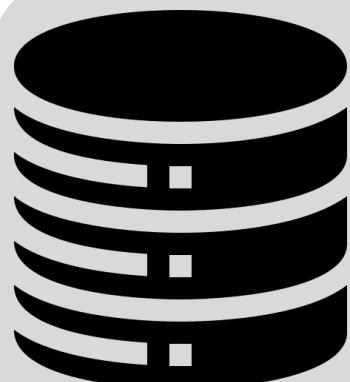


Python

Module

- 파생 변수 생성
식당 데이터 + 날씨 데이터
점심, 저녁 메뉴 칼럼 생성
SQLite3 연결

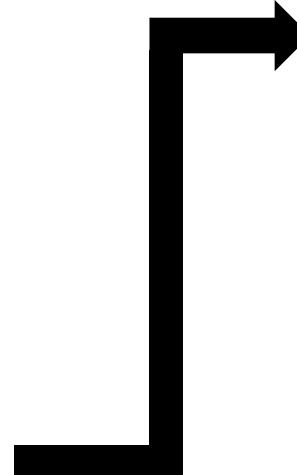
SQL



데이터 DB화

- HR Data table
Lunch Data table
Dinner Data table
Weather Data table
Calendar Data table

Python



R

1. 시각화

- Matplotlib
Seaborn

전처리

- One-Hot encoding
MinMax Scalar

2. 머신러닝

- Scikit-Learning(XGBRegressor)

3. 딥러닝

- Tensorflow – Keras(RNN)

통계분석

- 변수들이 통계적으로 유의미 여부 보고서 작성
요일, 계절, 신메뉴 여부 등

2. Analysis

Data

일자	요일	본사 정원 수	본사 휴가 자수	본사 출장 자수	본사시간외근 무명령서승인 건수	현본사소 속재택근 무자수	조식메뉴	중식메뉴	석식메뉴	증식계	석식계
0 2016-02-01	월	2601	50	150	238	0.0	모닝롤/찐빵 우유/두유/주스 계란후라이 호두죽/쌀밥 (쌀:국내산) 된장찌개 줘...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 오징어찌개 쇠불고기 (쇠고기:호주산) 계란찜 ...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 육개장 자반고등어구이 두부조림 건파래무침 ...	1039.0	331.0
1 2016-02-02	화	2601	50	173	319	0.0	모닝롤/단호박샌드 우유/두유/주스 계란후라이 팔죽/쌀밥 (쌀:국내산) 호박전국찌...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 김치찌개 가자미튀김 모둠소세지구이 마늘쫑무...	콩나물밥*양념장 (쌀,현미흑미:국내산) 어묵국 유산슬 (쇠고기:호주산) 아삭고추무...	867.0	560.0
2 2016-02-03	수	2601	56	180	111	0.0	모닝롤/베이글 우유/두유/주스 계란후라이 표고버섯죽/쌀밥 (쌀:국내산) 콩나물국...	카레덮밥 (쌀,현미흑미:국내산) 팽이장국 치킨핑거 (닭고기:국내산) 쫄면야채무침 ...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 청국장찌개 황태양념구이 (황태:러시아산) 고기...	1017.0	573.0
3 2016-02-04	목	2601	104	220	355	0.0	모닝롤/토마토샌드 우유/두유/주스 계란후라이 닭죽/쌀밥 (쌀,닭:국내산) 근대국...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 쇠고기무국 주꾸미볶음 부추전 시금치나물 ...	미니김밥*겨자장 (쌀,현미흑미:국내산) 우동 멕시칸샐러드 군고구마 무피클포...	978.0	525.0
4 2016-02-05	금	2601	278	181	34	0.0	모닝롤/와플 우유/두유/주스 계란후라이 쇠고기죽/쌀밥 (쌀:국내산) 재첩국방...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 떡국 돈육씨앗강정 (돼지고기:국내산) 우엉잡채...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 차돌박이찌개 (쇠고기:호주산) 닭갈비 (닭고기:....)	925.0	330.0

1. 메뉴 : 분석, 머신러닝에 필요한 데이터만 뽑아야 한다.
2. 정원 : 실제 사내에서 근무 중인 정원 수를 구한다.
3. 일자 : 계절, 날씨 등 요일 외 추가적인 정보를 확보해 추가해준다.

Data

	지점	지점명	일시	평균기온(°C)	일강수량(mm)	평균 풍속(m/s)	평균 상대습도(%)
0	192	진주	2016-02-01	-0.6	NaN	1.3	43.9
1	192	진주	2016-02-02	-2.3	NaN	0.8	47.5
2	192	진주	2016-02-03	-1.7	NaN	0.6	57.1
3	192	진주	2016-02-04	-0.2	NaN	0.7	53.4
4	192	진주	2016-02-05	1.3	NaN	1.0	44.4
...
2171	192	진주	2022-01-11	0.1	NaN	2.3	44.4
2172	192	진주	2022-01-12	-2.2	NaN	1.1	42.0
2173	192	진주	2022-01-13	-2.4	NaN	2.0	43.8
2174	192	진주	2022-01-14	-3.0	NaN	0.5	47.5
2175	192	진주	2022-01-15	-0.3	NaN	0.4	50.0

- 삭제: 지점, 지점명 등 불필요한 칼럼은 제거한다.
- 추가: 기온, 강수량, 풍속, 습도 등을 이용해 불쾌지수, 체감온도 칼럼을 추가한다.
- 변경: 비가 오지 않았을 때 결측값으로 처리했다. 이를 0으로 바꿔준다.

식당 CSV 데이터 변수

일자

요일

본사
정원수

출장자
수

야근자
수

재택
근무자

조식
메뉴

중식
메뉴

석식
메뉴

중식
인원

석식
인원

날씨 CSV 데이터 변수

지점

지점명

일시

평균
기온

일일
강수량

평균
풍속

평균
습도

식당 CSV 데이터 변수

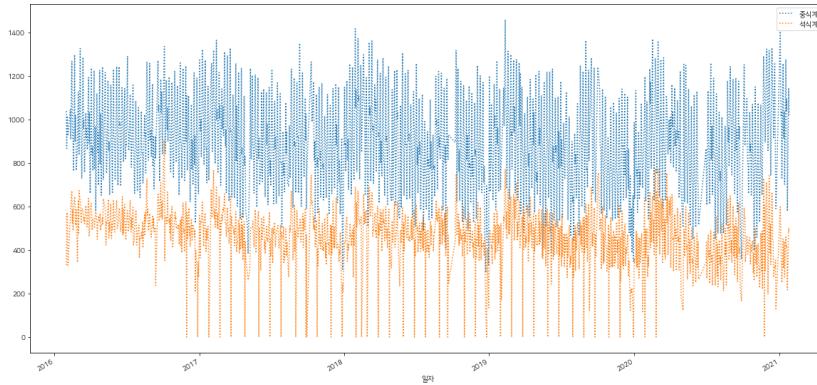
일자	요일	본사 정원수	출장자 수	야근자 수	재택 근무자	중식 메뉴	석식 메뉴	중식 인원	석식 인원
계절	연도	월	일	실근무 자수	밥	국	메인		

날씨 CSV 데이터 변수

일시	평균 기온	일일 강수량	평균 풍속	평균 습도	불쾌 지수	체감 온도
----	----------	-----------	----------	----------	----------	----------

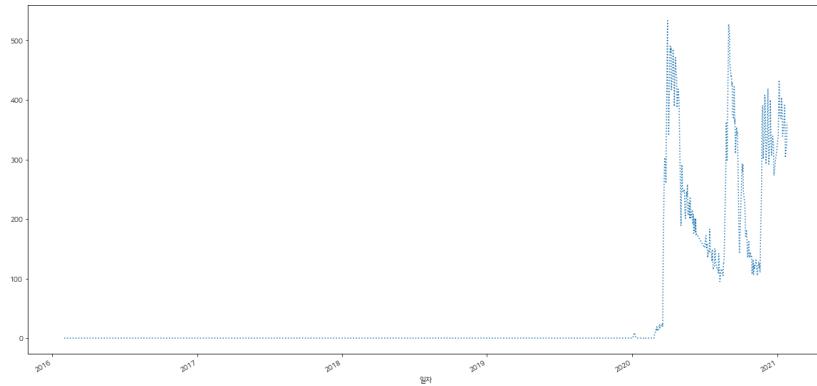
시계열 EDA

- 일별 중식, 석식 인원수 변화추이



1. 매년 비슷한 패턴의 증감 추이를 보임
2. 계절, 요일에 따른 증감 패턴이 있을 수 있음을 유추
3. 석식의 경우, 인원이 0인 값 파악. 석식 운영을 하지 않은 날 존재

- 재택근무자수 변화추이



1. 코로나이후 재택근무자수가 늘어남
2. 재택근무자수 변화에 비해 식수인원 증감 변화폭이 크지 않음

SQL EDA

In [3]:

```
#일반 '밥' 아닌 메뉴 평균 중식계
conn = sqlite3.connect("data/team3.db")
cur= conn.cursor()
cur.execute("SELECT avg(lunch_number) FROM lunch WHERE lunch_rice != '밥'" )

for row in cur:
    print(row)
```

(851.009900990099,)

Menu :

밥&특식, 국&찌개, 면유무, 신메뉴 여부

중식은 영향 있는 편, 석식은 영향이 적은 편

특히 국 종류 선호 있는 것으로 유추

Weather :

기온에 따라 식당 이용 인원이 늘어나는 경향 X

비 오는 경우, 중식 이용 인원 늘어나는 편.

비 오는 경우에도 석식 이용 인원은 변화 없는 편.

Date :

주말에 가까울수록 이용 인원 감소

수요일 저녁은 자기계발의 날이 있어 0인 날 존재

연휴 전날 식수인원이 줄어드는 경향 보임

Machine Learning

- One-Hot Encoding

Data columns (total 24 columns):			
#	Column	Non-Null Count	Dtype
0	datetime	1205 non-null	datetime64[ns]
1	season	1205 non-null	object
2	year	1205 non-null	int64
3	month	1205 non-null	int64
4	date	1205 non-null	int64
5	weekdays	1205 non-null	object
6	vacation	1205 non-null	object
7	worker_number	1205 non-null	int64
8	real_number	1205 non-null	int64
9	vacation_number	1205 non-null	int64
10	biztrip_number	1205 non-null	int64
11	overtime_number	1205 non-null	int64
12	telecom_number	1205 non-null	int64
13	temperature	1204 non-null	float64
14	rain	1205 non-null	float64
15	wind	1204 non-null	float64
16	humidity	1205 non-null	float64
17	discomfort_index	1204 non-null	float64
18	perceived_temperature	1203 non-null	float64
19	lunch_rice	1205 non-null	object
20	lunch_soup	1205 non-null	object
21	lunch_main	1205 non-null	object
22	new_lunch	1205 non-null	object
23	lunch_number	1205 non-null	int64

1. 카테고리화해야 할 데이터 칼럼

season, year, month, date, weekdays, vacation,
lunch_rice, new_lunch

계절, 연도, 월, 일, 요일, 쌀밥 or 특식, 신메뉴 여부

2. Coding

`df['columns'] = df['columns'].astype(category)`

year, month, date 등 숫자로 된 값들은 카테고리로 바꿔주기

`df_coding = pd.get_dummies(df)`

year, month, date 등 숫자로 된 값들은 카테고리로 바꿔주기

Machine Learning

- GridSearchCV

```
param = {  
    'max_depth':[2,3,4],  
    'n_estimators':range(300,600,100),  
    'colsample_bytree':[0.5,0.7,1],  
    'colsample_bylevel':[0.5,0.7,1]  
}  
  
model = XGBRegressor()  
  
gs = GridSearchCV(estimator=model, param_grid=param, cv=10,  
scoring='neg_mean_squared_error',  
n_jobs=multiprocessing.cpu_count())  
  
gs.fit(lunch_X_train, lunch_y_train)  
  
print(gs.best_params_)  
print(gs.best_score_)
```

[최적의 파라미터 찾기]

- KFold & cross_val_score

```
kfold = KFold(n_splits=10, shuffle=True)  
  
xgbr_lunch = XGBRegressor(n_estimators=300, colsample_bylevel=1,  
learning_rate=0.1, colsample_bytree=1, max_depth=2)  
scores = cross_val_score(xgbr_lunch, lunch_train, lunch_target,  
cv=kfold, scoring="neg_mean_squared_error")  
print(scores)  
round(np.mean(scores)*100, 2)
```

[오차 적은 머신러닝 모델 만들기]

Machine Learning

- LinearRegression

```
lr = LinearRegression()
lr.fit(lunch_X_train, lunch_y_train)

pred = lr.predict(lunch_X_test)
print("훈련 결과 : {}".format(lr.score(lunch_X_train, lunch_y_train)))
print("점검 결과 : {}".format(lr.score(lunch_X_test, lunch_y_test)))
```

훈련결과:0.8060685580206686

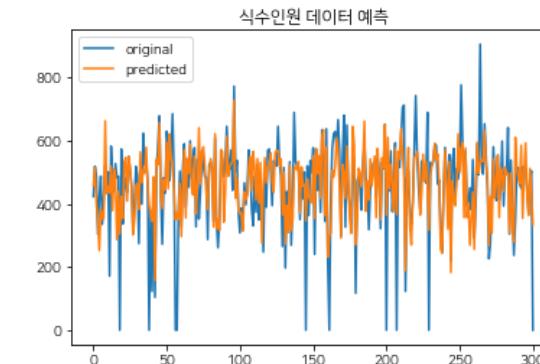
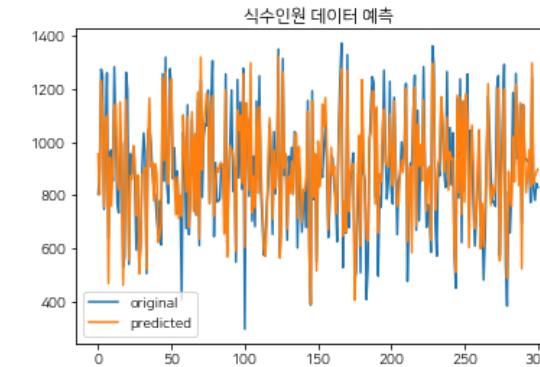
점검결과:0.81850928043697

```
lr = LinearRegression()
lr.fit(dinner_X_train, dinner_y_train)

pred = lr.predict(dinner_X_test)
print("훈련 결과 : {}".format(lr.score(dinner_X_train, dinner_y_train)))
print("점검 결과 : {}".format(lr.score(dinner_X_test, dinner_y_test)))
```

훈련결과:0.5918834531954169

점검결과:0.49013363353201655



Machine Learning

- Ridge

```
lr = model = Ridge(alpha=10)
lr.fit(lunch_X_train, lunch_y_train)

pred = lr.predict(lunch_X_test)
print("훈련 결과 : {}".format(lr.score(lunch_X_train, lunch_y_train)))
print("점검 결과 : {}".format(lr.score(lunch_X_test, lunch_y_test)))
```

훈련결과:0.8051371079674186

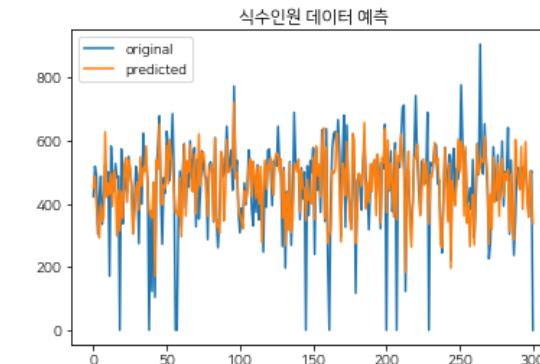
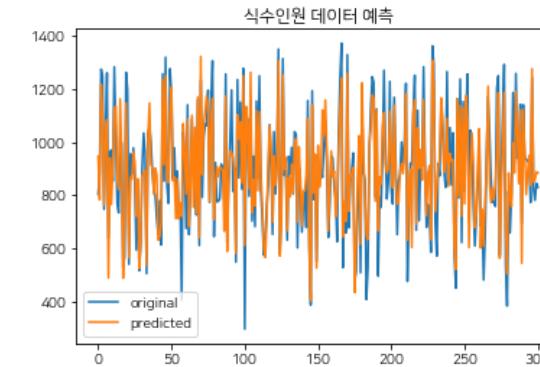
점검결과:0.8216060418923412

```
lr = model = Ridge(alpha=10)
lr.fit(dinner_X_train, dinner_y_train)

pred = lr.predict(dinner_X_test)
print("훈련 결과 : {}".format(lr.score(dinner_X_train, dinner_y_train)))
print("점검 결과 : {}".format(lr.score(dinner_X_test, dinner_y_test)))
```

훈련결과:0.5846964863308441

점검결과:0.5114079584877648



Machine Learning

- Lasso

```
lr = Lasso(alpha=1)
lr.fit(lunch_X_train, lunch_y_train)

pred = lr.predict(lunch_X_test)
print("훈련 결과 : {}".format(lr.score(lunch_X_train, lunch_y_train)))
print("점검 결과 : {}".format(lr.score(lunch_X_test, lunch_y_test)))
```

훈련결과:0.79289893058166

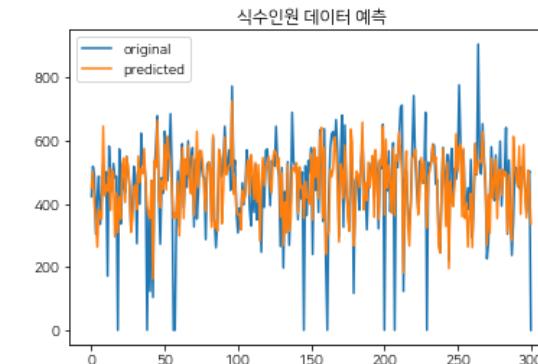
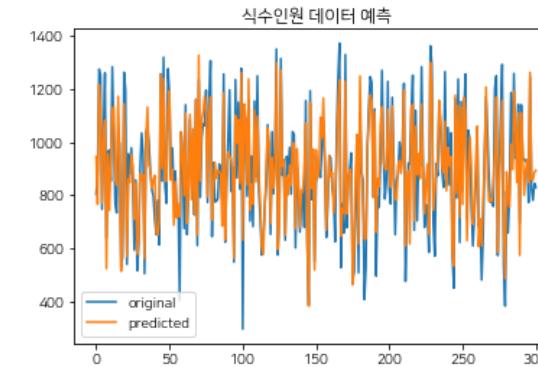
점검결과:0.825226121116565

```
lr = Lasso(alpha=0.1)
lr.fit(dinner_X_train, dinner_y_train)

pred = lr.predict(dinner_X_test)
print("훈련 결과 : {}".format(lr.score(dinner_X_train, dinner_y_train)))
print("점검 결과 : {}".format(lr.score(dinner_X_test, dinner_y_test)))
```

훈련결과:0.5915282738733196

점검결과:0.49785499714720094



Machine Learning

- ElasticNet

```
lr = ElasticNet(alpha=0.01)
lr.fit(lunch_X_train, lunch_y_train)

pred = lr.predict(lunch_X_test)
print("훈련 결과 : {}".format(lr.score(lunch_X_train, lunch_y_train)))
print("점검 결과 : {}".format(lr.score(lunch_X_test, lunch_y_test)))
```

훈련결과:0.8068362251560386

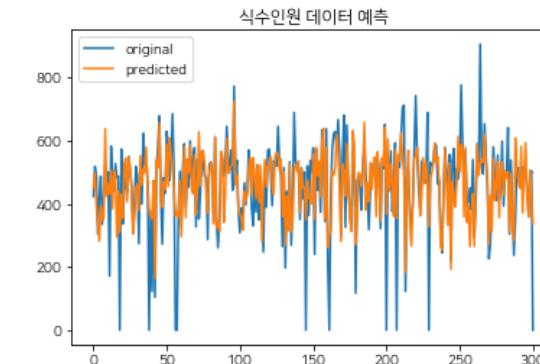
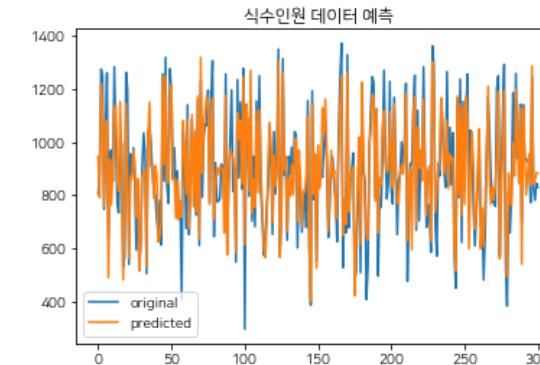
점검결과:0.8215253825745953

```
lr = ElasticNet(alpha=0.01)
lr.fit(dinner_X_train, dinner_y_train)

pred = lr.predict(dinner_X_test)
print("훈련 설명력 : {}".format(lr.score(dinner_X_train, dinner_y_train)))
print("점검 결과 : {}".format(lr.score(dinner_X_test, dinner_y_test)))
```

훈련결과:0.5895755419338433

점검결과:0.5038979785074525



Machine Learning

- XGBRegressor

```
model = XGBRegressor(n_estimators=300, colsample_bytree=1,  
learning_rate=0.1, colsample_bytree=1, max_depth=2)  
  
model.fit(lunch_X_train, lunch_y_train)  
  
pred = model.predict(lunch_X_test)  
print("훈련 결과 : {}".format(lr.score(lunch_X_train, lunch_y_train)))  
print("점검 결과 : {}".format(lr.score(lunch_X_test, lunch_y_test)))
```

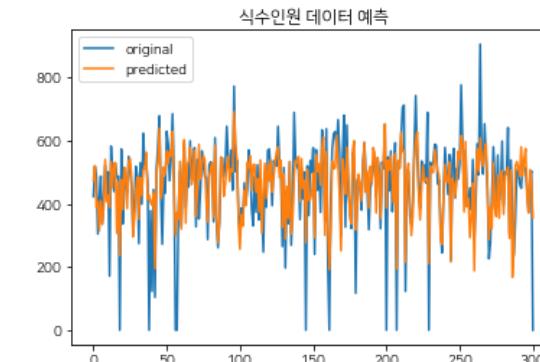
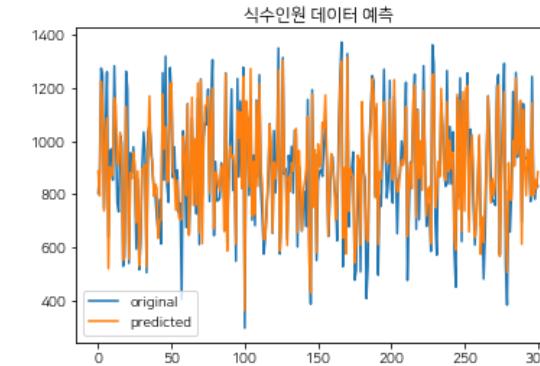
훈련결과:0.9023025200953646

점검결과:0.860508319733749

```
model = XGBRegressor(n_estimators=300, colsample_bytree=0.7,  
learning_rate=0.1, colsample_bytree=1, max_depth=2)  
  
model.fit(dinner_X_train, dinner_y_train)  
  
pred = model.predict(dinner_X_test)  
print("훈련 결과 : {}".format(lr.score(dinner_X_train, dinner_y_train)))  
print("점검 결과 : {}".format(lr.score(dinner_X_test, dinner_y_test)))
```

훈련결과:0.7960968384558765

점검결과:0.5983145743541849



Machine Learning

- LightGBMRegressor

```
model = LGBMRegressor(learning_rate=0.1, max_depth=-1, n_estimators=500,  
num_leaves=3)  
  
model.fit(lunch_X_train, lunch_y_train)  
pred = model.predict(lunch_X_test)  
print("훈련 결과 : {}".format(model.score(lunch_X_train, lunch_y_train)))  
print("점검 결과 : {}".format(model.score(lunch_X_test, lunch_y_test)))
```

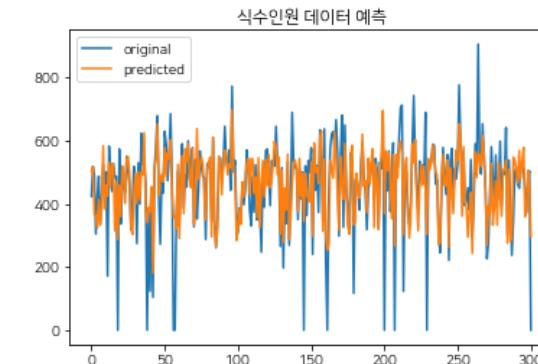
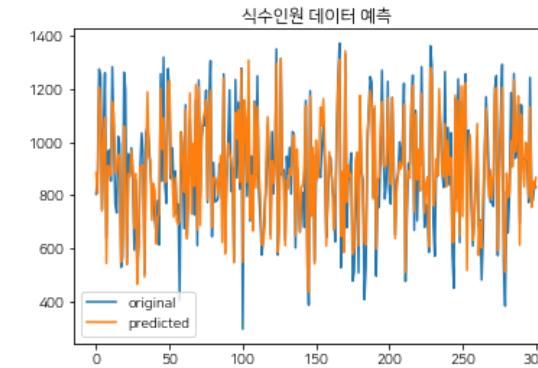
훈련결과:0.8945925450346035

점검결과:0.8516641136589393

```
model = LGBMRegressor(learning_rate=0.03, max_depth=3, n_estimators=500,  
num_leaves=14)  
model.fit(dinner_X_train, dinner_y_train)  
pred = model.predict(dinner_X_test)  
print("훈련결과 : {}".format(model.score(dinner_X_train, dinner_y_train)))  
print("점검결과 : {}".format(model.score(dinner_X_test, dinner_y_test)))
```

훈련결과:0.7474081798821995

점검결과:0.5423141823861122



Result

XGB Regressor

Accuracy
Neg mean squared error

Tree > Linear

Model

LightGBM과 비슷한 정확성 보이나
XGB가 훈련성적, 점검성적 둘 다 더 좋은 결과

Reason

정확성 그리고 회귀분석 평가지표인 MSE가
XGB가 LightGBM보다 좋은 결과

Opinion

XGB, LightGBM은 트리구조의
회귀분석방법이다. 이를 근거로 딥러닝에서도
좋은 결과를 낼 수 있지 않을까 추측

Deep Learning(시계열 분석)

- 정규화/입력 데이터 배열(데이터수, 시계열 기간, 칼럼 수)

```
standard_scaler = StandardScaler()

#점심
lunch_fitted = standard_scaler.fit(lunch)

lunch_output = standard_scaler.transform(lunch)
lunch_output = pd.DataFrame(lunch_output, columns=lunch.columns,
index=list(lunch.index.values))
print(lunch_output.head())

#저녁
dinner_fitted = standard_scaler.fit(dinner)

dinner_output = standard_scaler.transform(dinner)
dinner_output = pd.DataFrame(dinner_output, columns=dinner.columns,
index=list(dinner.index.values))
print(dinner_output.head())
```

1. 정규화

MinMax, Standard 둘 다 활용 가능

MinMax, Standard 둘 다 성능에 큰 차이 없어
다만, 정규화하지 않으면 성능 크게 떨어져

```
# reshape into (size(개수), time step, 입력 feature)
train_x = train_x.reshape(train_x.shape[0],time_step,input_size)
validation_x = validation_x.reshape(validation_x.shape[0], time_step,
input_size)
test_x = test_x.reshape(test_x.shape[0], time_step, input_size)
train_x.shape, test_x.shape
```

2. 입력 데이터 배열 - 시계열 분석에 맞게

np.array(data size, time step, feature)

Deep Learning

- LSTM

```
# 모델 구조
model = Sequential()
model.add(LSTM(1024, input_shape=(1, input_size)))
model.add(Dropout(0.2))
model.add(Dense(1))

#오차 및 최적화기 설정
model.compile(loss='mse',
optimizer=tf.keras.optimizers.Adam(lr=learning_rate),
metrics=['mae', 'mape'])
model.summary()

#학습
history = model.fit(train_x, train_y, epochs=training_cnt,
batch_size=batch_size, verbose=1)
val_mse, val_mae, val_mape = model.evaluate(test_x, test_y, verbose=0)
```

[최적의파라미터찾기]

- LSTM Parameter

```
learning_rate = 0.01
training_cnt = 100
batch_size = 200
input_size = train_x.shape[1]
time_step = 1
```

[오차적은머신러닝 모델만들기]

Deep Learning

- LSTM

Layer (type) Output Shape Param #

lstm_4 (LSTM) (None, 1024) 4308992
dropout_4 (Dropout) (None, 1024) 0
dense_4 (Dense) (None, 1) 1025

Total params: 4,310,017 Trainable params: 4,310,017 Non-trainable params: 0

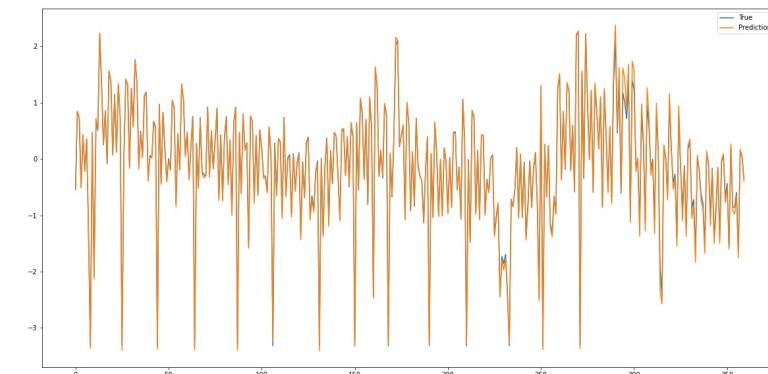
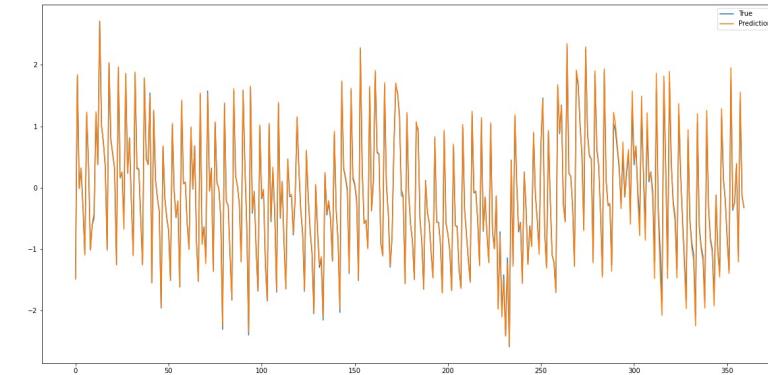
loss: 0.0012 - mae: 0.0249 - mape: 11.6815

Layer (type) Output Shape Param #

lstm_4 (LSTM) (None, 1024) 4308992
dropout_4 (Dropout) (None, 1024) 0
dense_4 (Dense) (None, 1) 1025

Total params: 4,310,017 Trainable params: 4,310,017 Non-trainable params: 0

loss: 0.0013 - mae: 0.0250 - mape: 11.4684



Result

LSTM
시계열분석

Accuracy
Neg mean squared error

Deep Learning

Model

Machine Learning 모델보다
우수한 성적을 보임

Reason

설명력, 오차범위 등 모든 지표에서
더 좋은 예측값을 보임

Opinion

식수 예측 서비스를 한다면
LSTM으로 시계열 분석하는 것이
가장 정확할 것으로 판단

3. Module

Module Outline

Target

데이터 2개만 입력 후 클래스, 함수 호출하면
자동으로 DB 생성, 데이터 분석이 이뤄지도록
클래스 소스 생성. <프로그램 개발 용이 목적>
분기별로 데이터 확보 후 머신러닝 모델 <자동 업데이트>

Process

sql.py : DB 생성, DB에 데이터 저장, DB에서 데이터 출력
train_encoding.py : 훈련에 적합한 데이터테이블 변경
forecast_encoding.py : 예측해야할 데이터 구조 생성 및 변경
learning.py : 훈련 / 데이터 예측까지

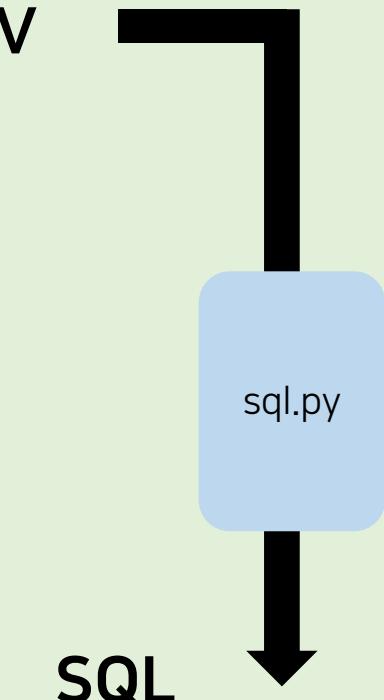
Library

pandas, numpy, matplotlib.pyplot
xgboost,
xgboost.XGBRegressor

Language

Python 3.9.7

CSV



Python

Module

-
- 파생 변수 생성
식당 데이터 + 날씨 데이터
점심, 저녁 메뉴 칼럼 생성
SQLite3 연결

Python

train_encoding.py
forecast_encoding.py
learning.py

데이터 DB화

-
- HR Data table
Lunch Data table
Dinner Data table
Weather Data table
Calendar Data table

1. 시각화

-
- Matplotlib
Seaborn

전처리

-
- One-Hot encoding
MinMax Scalar

2. 머신러닝

-
- Scikit-Learning(XGBRegressor)

3. 딥러닝

-
- Tensorflow - Keras(RNN)

통계분석

-
- 변수들이 통계적으로 유의미 여부 보고서 작성
요일, 계절, 신메뉴 여부 등

R

Path :

<https://github.com/obilige/Team3/tree/master/module>

kinds :

sql.py : 프로그램 내 DB 관련 자동화 클래스 모음

train_encoding.py : 훈련용 데이터 인코딩 관련 클래스 모음

Forecast_encoding.py : 예측용 데이터 인코딩 관련 클래스 모음

Learning.py : XGBRegressor, RNN 분석 클래스 모음

Code Structure :

우측 이미지 참고

```
class Train_Encoding():
    def __init__(self, lunch, dinner):
        self.lunch = lunch
        self.dinner = dinner

    def Na(self):
        self.lunch = self.lunch.dropna()
        self.dinner = self.dinner.dropna()

    return self.lunch, self.dinner

    def split_lunch(self):
        lunch_data = self.lunch.drop("lunch_number", axis = "columns")
        lunch_target = self.lunch['lunch_number']
        self.lunch_X_train, self.lunch_X_test, self.lunch_y_train, ...
        = train_test_split(lunch_data, lunch_target)

    return self.lunch_X_train, self.lunch_X_test, self.lunch_y_train, ...

    def split_dinner(self):
        dinner_data = self.dinner.drop("dinner_number", axis = "columns")
        dinner_target = self.dinner['dinner_number']
        self.dinner_X_train, self.dinner_X_test, self.dinner_y_train, ...
        = train_test_split(dinner_data, dinner_target)

    return self.dinner_X_train, self.dinner_X_test, self.dinner_y_train, ...

    def format(self): ...
```

4. Mission

```

class Train_Encoding():
    def __init__(self, lunch, dinner):
        self.lunch = lunch
        self.dinner = dinner

    def Na(self):
        self.lunch = self.lunch.dropna()
        self.dinner = self.dinner.dropna()

    return self.lunch, self.dinner

    def split_lunch(self):
        lunch_data = self.lunch.drop("lunch_number", axis = "columns")
        lunch_target = self.lunch['lunch_number']
        self.lunch_X_train, self.lunch_X_test, self.lunch_y_train, ...
        = train_test_split(lunch_data, lunch_target)

    return self.lunch_X_train, self.lunch_X_test, self.lunch_y_train, ...

    def split_dinner(self):
        dinner_data = self.dinner.drop("dinner_number", axis = "columns")
        dinner_target = self.dinner['dinner_number']
        self.dinner_X_train, self.dinner_X_test, self.dinner_y_train, ...
        = train_test_split(dinner_data, dinner_target)

    return self.dinner_X_train, self.dinner_X_test, self.dinner_y_train, ...

    def format(self): ...

```

Mission 1.

모듈 완성

RNN 이용한 딥러닝 분석

분석 모델 pickle 저장

저장된 pickle로 분석 결과 도출 모듈 생성

Suggestion.

2022.01.22 Test 예정

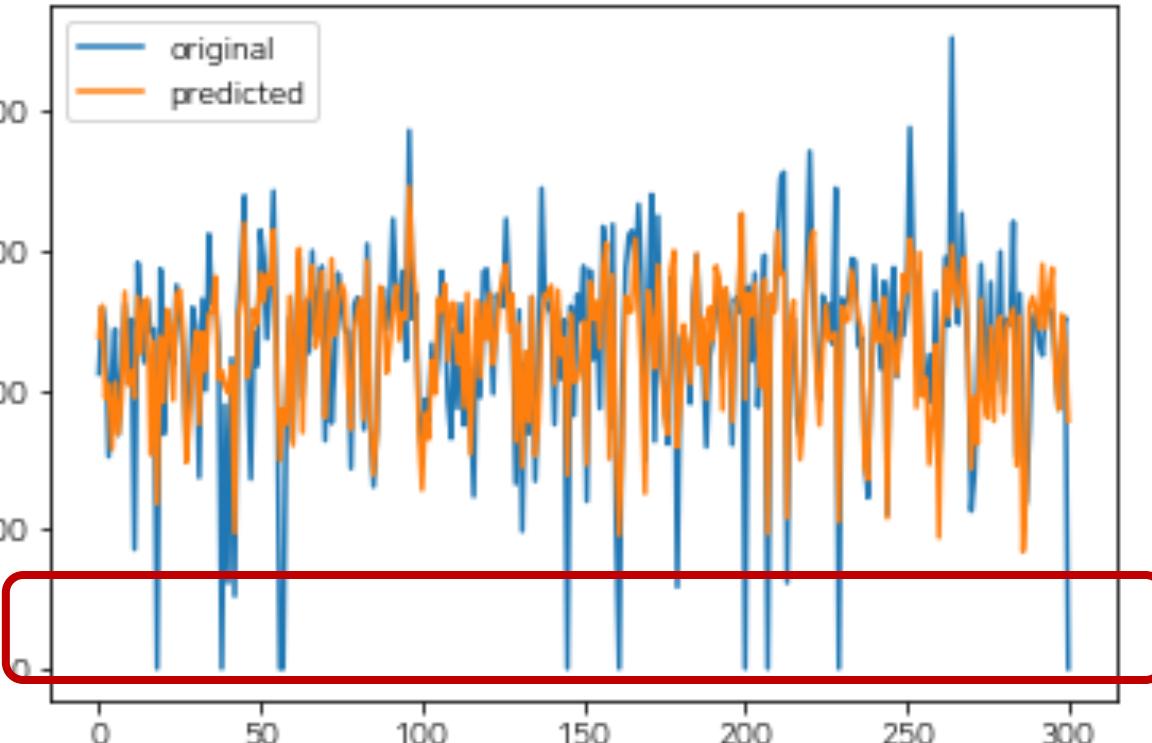
2022.01.24 Test에서 발견한 문제 수정

2022.02.20 분석 모델 pickle 로드 코드 생성

2022.02.21 Test 예정

2022.02.24 Test에서 발견한 문제 수정

식수인원 데이터 예측

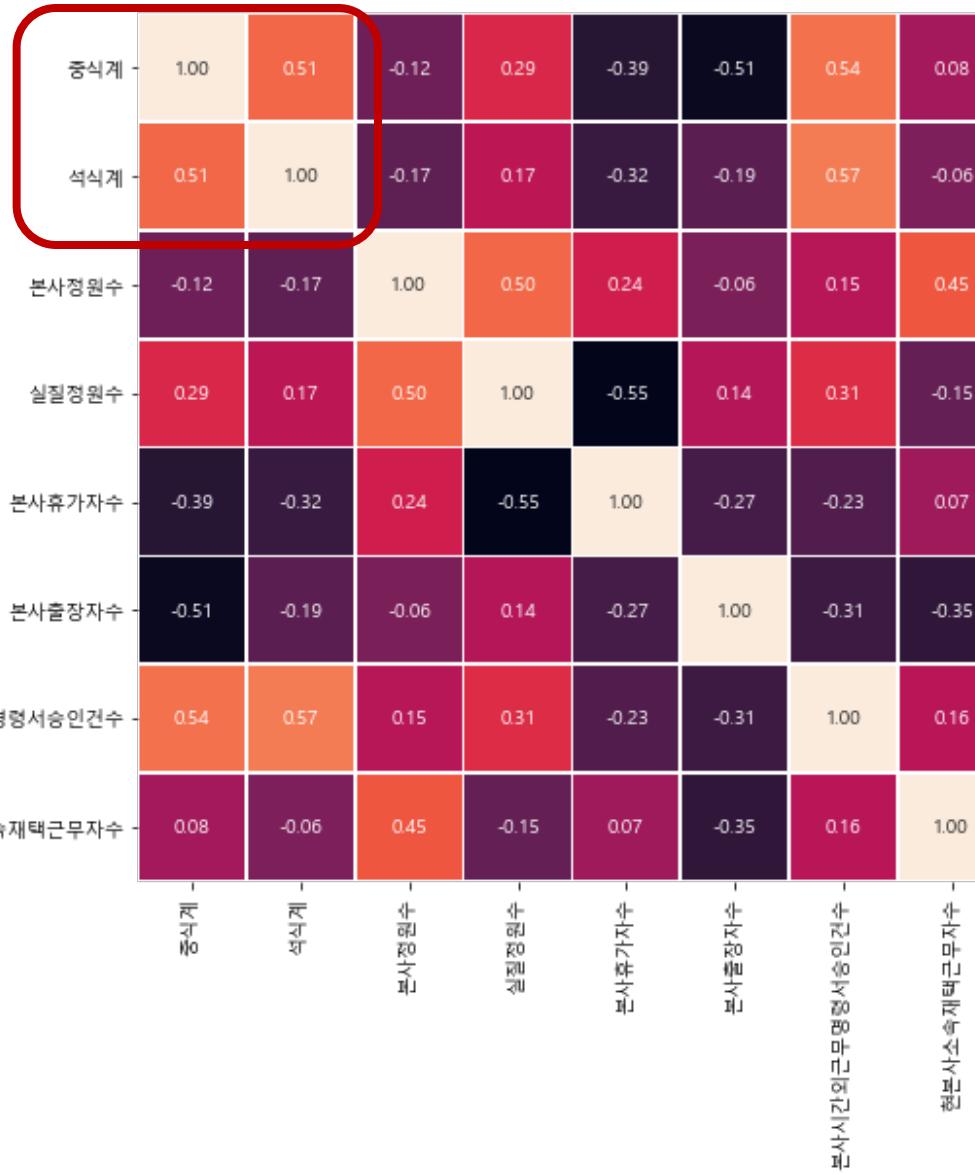


Mission 2.

석식 예측에서 0에 가깝게 떨어지는 값을 전혀 예측하지 못하고 있다. 이에 대한 해결방안 필요
RNN - LSTM 방법으로 해결

Suggestion.

매주 수요일이 자기계발의 날이라 운영을 하지 않음
하나, 이에 대한 데이터가 같이 훈련되었음
석식 이용 수가 0인 값은 제거 후 다시 훈련



Mission 2.

석식 예측에서 0에 가깝게 떨어지는 값을 전혀 예측하지 못하고 있다. 이에 대한 해결방안 필요

RNN – LSTM 방법으로 해결

Suggestion.

또 다른 해결방안으로 EDA 도중 파악한 중식인원 수와 석식인원 수의 상관관계를 이용해 석식 예측에 중식인원 수를 넣어 머신러닝하는 방안.

다만, 이를 위해선 매일 중식인원 수가 바로 업데이트되고 그 값을 포함해 머신러닝을 돌려야 하는 부담이 있다.

조식메뉴	중식메뉴	석식메뉴
모닝롤/찐빵 우유/두유/주스 계란후라이 호두죽/쌀밥 (쌀:국내산) 된장찌개 쥐...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 오징어찌개 쇠불고기 (쇠고기:호주산) 계란찜 ...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 육개장 자반고등어구이 두부조림 건파래무침 ...
모닝롤/단호박샌드 우유/두유/주스 계란후라이 팥죽/쌀밥 (쌀:국내산) 호박젓국찌...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 김치찌개 가자미튀김 모둠소세지구이 마늘쫑무...	콩나물밥*양념장 (쌀,현미흑미:국내산) 어묵국 유산슬 (쇠고기:호주산) 아삭고추무...
모닝롤/베이글 우유/두유/주스 계란후라이 표고버섯죽/쌀밥 (쌀:국내산) 콩나물국...	카레덮밥 (쌀,현미흑미:국내산) 팽이장국 치킨핑거 (닭고기:국내산) 쫄면야채무침 ...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 청국장찌개 황태양념구이 (황태:러시아산) 고기...
모닝롤/토마토샌드 우유/두유/주스 계란후라이 닭죽/쌀밥 (쌀,닭:국내산) 근대국...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 쇠고기무국 주꾸미볶음 부추전 시금치나물 ...	미니김밥*겨자장 (쌀,현미흑미:국내산) 우동 멕시칸샐러드 군고구마 무피를포...
모닝롤/와플 우유/두유/주스 계란후라이 쇠고기죽/쌀밥 (쌀:국내산) 재첩국방...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 떡국 돈육씨앗강정 (돼지고기:국내산) 우엉잡채...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 차돌박이찌개 (쇠고기:호주산) 닭갈비 (닭고기):...

Mission 3.

메뉴 너무 많고 카테고리화하기 어려워 머신러닝에선
쌀밥 / 특식으로 나눈 나머지 데이터는 다제거

Suggestion.

식품 분류 API(카테고리, 이미지)가 제공되고 있음.
API 데이터를 CNN으로 카테고리별 분류한 다음
원핫인코딩하여 메뉴 데이터가 식수 인원에 영향을 주는지 파악하고
영향을 주면 머신러닝, 딥러닝에 적극 활용



Mission 4.

사내근처 주변음식점이나 먹자골목 유무 등 추가적인 변수에 대한 학습도 필요하지 않을까?

Suggestion.

비용이 많이 들지만, 이를 통해 정확한 분석이 가능하다면 추가비용을 받고 사옥주변 상권의 상황을 추가입력해 예측하는 서비스를 제공하는 방향으로 진행 가능하다 판단

:: 예) 반경 500m 내 주변음식점 수, 업종 등 칼럼 추가



Mission 5.

기온, 강수량, 풍속 등을 일일 평균 데이터로 활용했음.
더 정확한 방법으로 훈련시켜보기

Suggestion.

기상청에서 시간별 기온, 강수량, 풍속 데이터 제공 중
중식은 12시, 석식은 6시 데이터를 적용해 다시 훈련

다만, 이 경우 모듈을 완전히 재수정해야 하는 상황 발생

0. Question?

- 1. Thank You! ☺