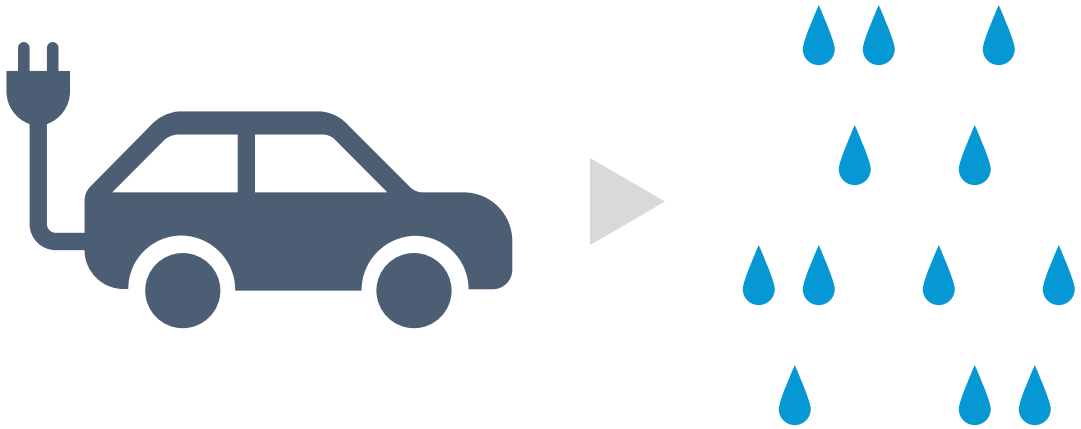
가스공급량 수요예측 모델개발

팀명: 멋쟁이승우처럼
팀장:문승우
팀원:정진우, 오소영, 강수정

목차 a table of contents

Part 1

주제 및 목표



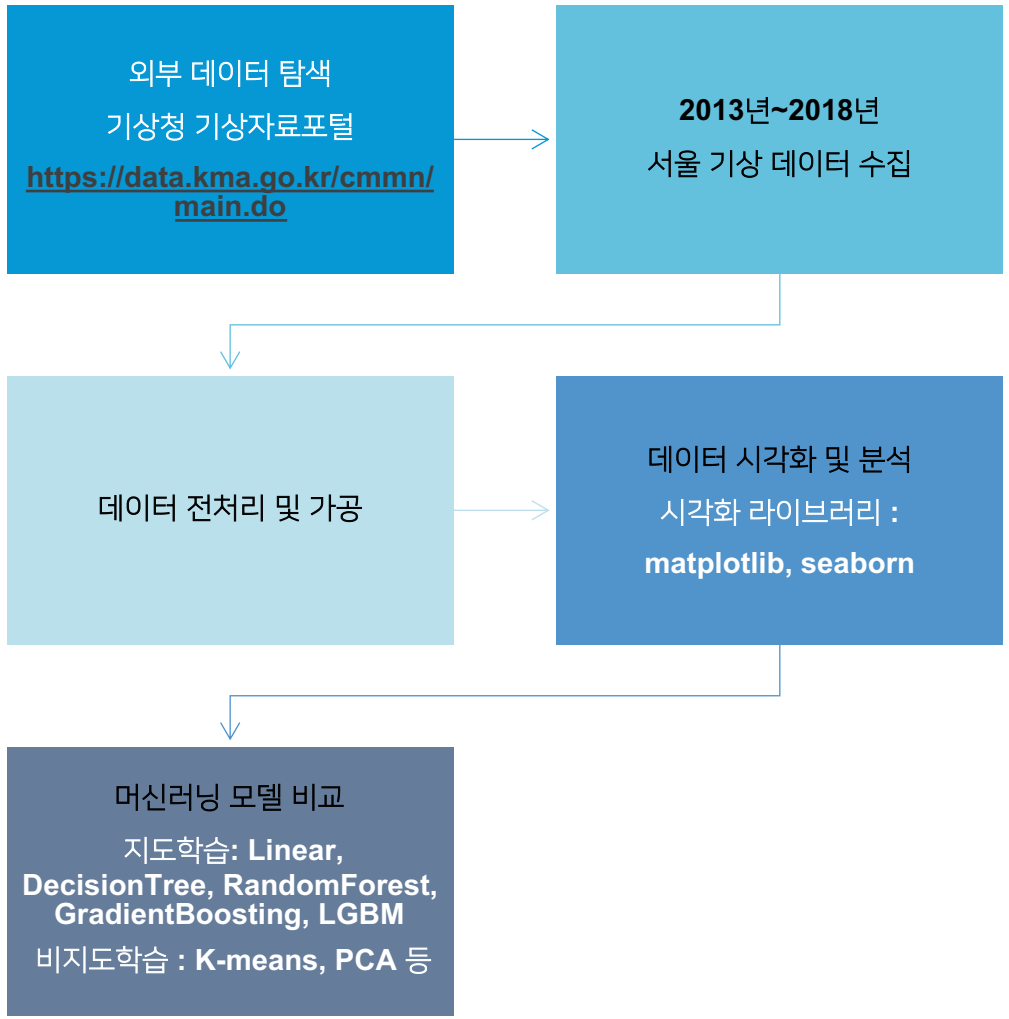
한국가스공사의 시간단위 공급량 내부 데이터와 기상정보 및 가스 외 발전량 등 외부 데이터를 포함한 데이터셋을 구축하여 **90**일 한도 일간 공급량을 예측하는 인공지능 모델을 개발

외부 데이터: 기상청 기상자료개방포털
https://data.kma.go.kr/cmmn/main.do

가스 공급량을 예측하는 머신러닝 / 딥러닝 모델을 구축 및 평가, 다양한 모델의 비교를 통해 최적의 모델을 적용

Part 2

데이터 수집 및 전처리



데이콘 대회 정보

대회명: 가스공급량 수요예측 모델개발 대회

대회 링크: https://dacon.io/competitions/official/235830/overview/description

주최 및 주관: 한국가스공사

대회 개요: 한국 가스 공사가 보유한 다년간 시간 단위 공급량 데이터를 기반으로 미래 공급량을 예측하는 모델 구축

Part 2

데이터 전처리 및 가공

학습용 데이터

```
total["구분"].unique()
```

```
array(['A', 'B', 'C', 'D', 'E', 'G', 'H'], dtype=object)
```

```
d_map = {}
for i, d in enumerate(total["구분"].unique()) :
    d_map[d] = i
total["구분"] = total["구분"].map(d_map)
```

```
total.head(10)
```

	연월일	시간	구분	공급량
0	2013-01-01	1	0	2497.129
1	2013-01-01	2	0	2363.265
2	2013-01-01	3	0	2258.505
3	2013-01-01	4	0	2243.969

구분 컬럼 문자열 변수 수치화

```
total["연월일"] = pd.to_datetime(total["연월일"])
```

```
total['year'] = total['연월일'].dt.year
total['month'] = total['연월일'].dt.month
total['day'] = total['연월일'].dt.day
total['weekday'] = total['연월일'].dt.weekday
```

```
train_years = [2013, 2014, 2015, 2016, 2017]
val_years = [2018]
```

```
train = total[total['year'].isin(train_years)]
val = total[total['year'].isin(val_years)]
```

```
features = ['구분', 'month', 'day', 'weekday', '시간']
train_x = train[features]
train_y = train['공급량']

val_x = val[features]
val_y = val['공급량']
```

연, 월, 일 데이터 타입 변경 및 추가 변수 생성

Part 2

데이터 전처리 및 가공

기온 데이터

2013년 데이터에 2014 ~ 2018년도 서울 기상 데이터 병합 ¶

```
for i in range(2014, 2019) :
    path = "../CSV/서울 기상 데이터/" + str(i) + "_seoul_weather.csv"
#     print(path)
    temp = pd.read_csv(path, encoding = 'cp949')

    seoul = pd.concat([seoul, temp], ignore_index = True)

    print(f"{i} temp shape : {temp.shape}")
```

```
2014 temp shape : (8760, 38)
2015 temp shape : (8760, 38)
2016 temp shape : (8784, 38)
2017 temp shape : (8760, 38)
2018 temp shape : (8760, 38)
```

2013 ~ 2018년 데이터 병합

기온(°C) 컬럼 결측치 확인

```
# gn.loc[gn["기온(° C)"].isnull(), "기온(° C)"]
# gn.loc[1805 : 1809, ["일시", "기온(° C)"]]
# gn.loc[8830 : 8837, ["일시", "기온(° C)"]]
seoul.loc[seoul["기온(° C)"].isnull(), "기온(° C)"]
```

```
17520    NaN
41513    NaN
41895    NaN
41896    NaN
41897    NaN
51810    NaN
Name: 기온(° C), dtype: float64
```

17520 결측치 제거

- 17519, 17521 평균으로 대체

```
seoul.loc[17518 : 17522, ["일시", "기온(° C)"]]
```

기온 데이터 결측치 처리

```
gas_data["기온(°C)"] = temp["기온(°C)"]
```

```
gas_data.tail(10)
```

기온 데이터 추가한 가스공급량 데이터 저장

```
gas_data.to_csv("../CSV/train_data.csv", index = False)
```

병합된 기온 데이터 저장

Part 2

데이터 전처리 및 가공

테스트 데이터 및 병합

```
test['일자'] = test['일자|시간|구분'].str.split(' ').str[0]
test['시간'] = test['일자|시간|구분'].str.split(' ').str[1].astype(int)
test['구분'] = test['일자|시간|구분'].str.split(' ').str[2]
```

컬럼 생성

```
test["일시"] = test["일자"] + " " + (test["시간"] - 1).astype(str) + ":00:00"
test["일시"] = pd.to_datetime(test["일시"])

test['year'] = test['일시'].dt.year
test['month'] = test['일시'].dt.month
test['day'] = test['일시'].dt.day
test['hour'] = test['일시'].dt.hour
test['weekday'] = test['일시'].dt.weekday
```

일시 컬럼 데이터 타입 변환

19년도 기온 예측

```
pred = model.predict(test_x)
test["기온(°C)"] = np.round(pred, 1) # 예측한 기온을 소수 첫째자리까지 표시
```

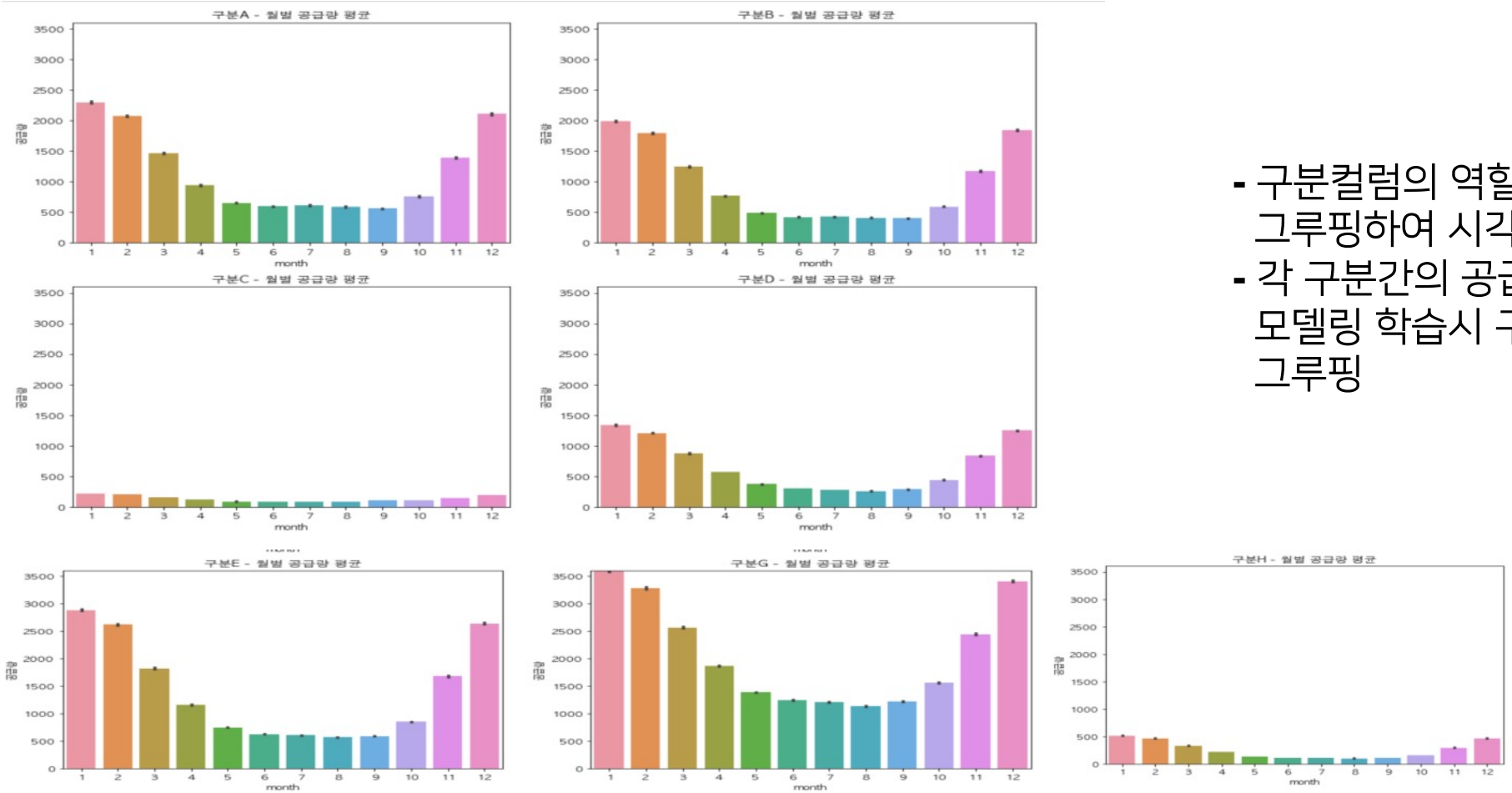
예측한 기온 데이터 병합

- '일자|시간|구분' 컬럼으로 새로운 컬럼을 생성
- 일시 컬럼의 데이터 타입을 datetime으로 변환
- 일시 컬럼으로 새로운 컬럼 생성
- 19년도 기온을 예측한 모델을 통해 예측한 기온데이터와 병합

Part 3

데이터 시각화 및 분석

Train Data - 구분별 월평균 공급량 **(barplot)**

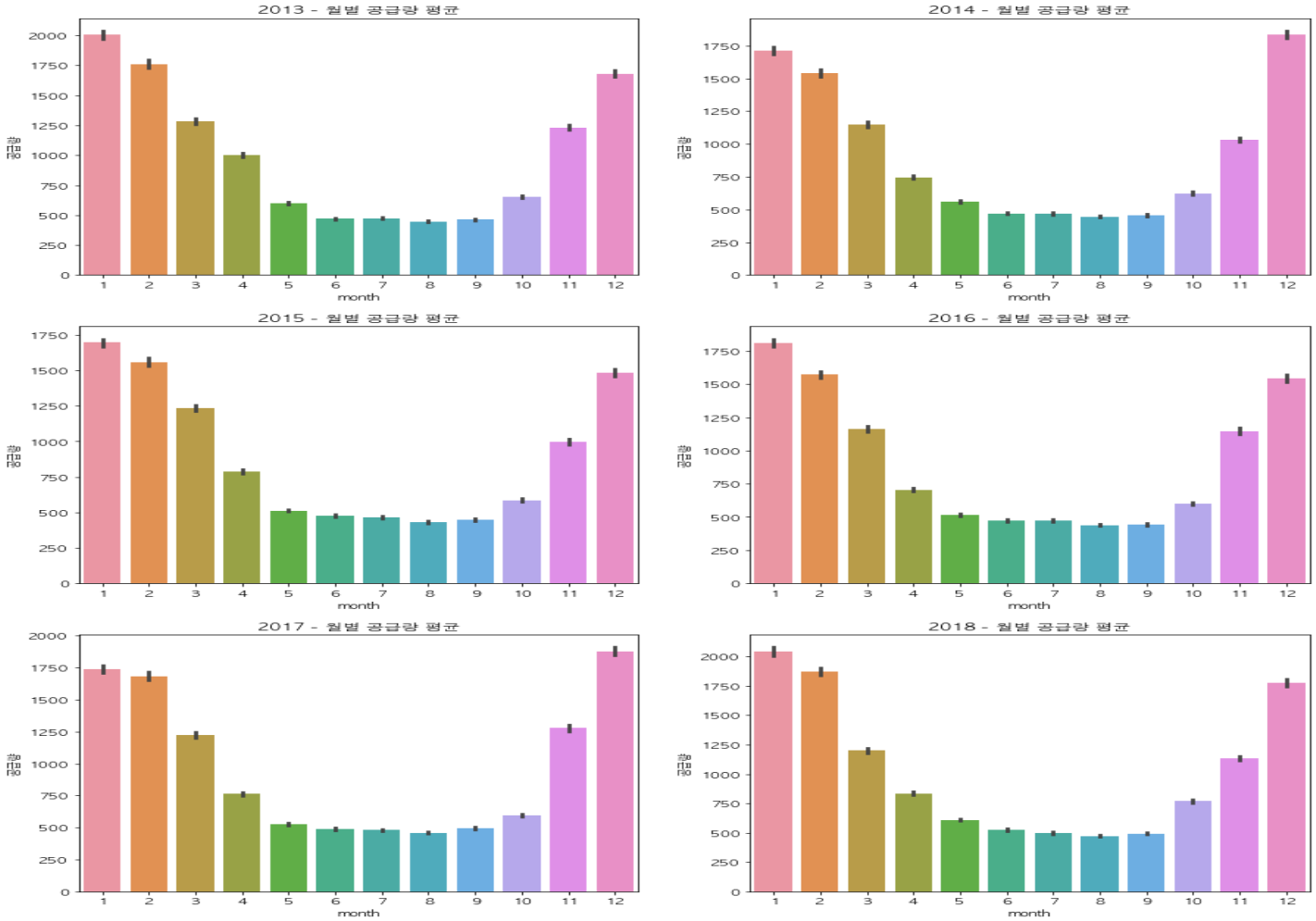


- 구분컬럼의 역할을 파악하기 위해 그루핑하여 시각화
- 각 구분간의 공급량 편차를 고려 모델링 학습시 구분별로 학습데이터를 그루핑

Part 3

데이터 시각화 및 분석

Train Data - 연도별 월평균 공급량 (barplot)



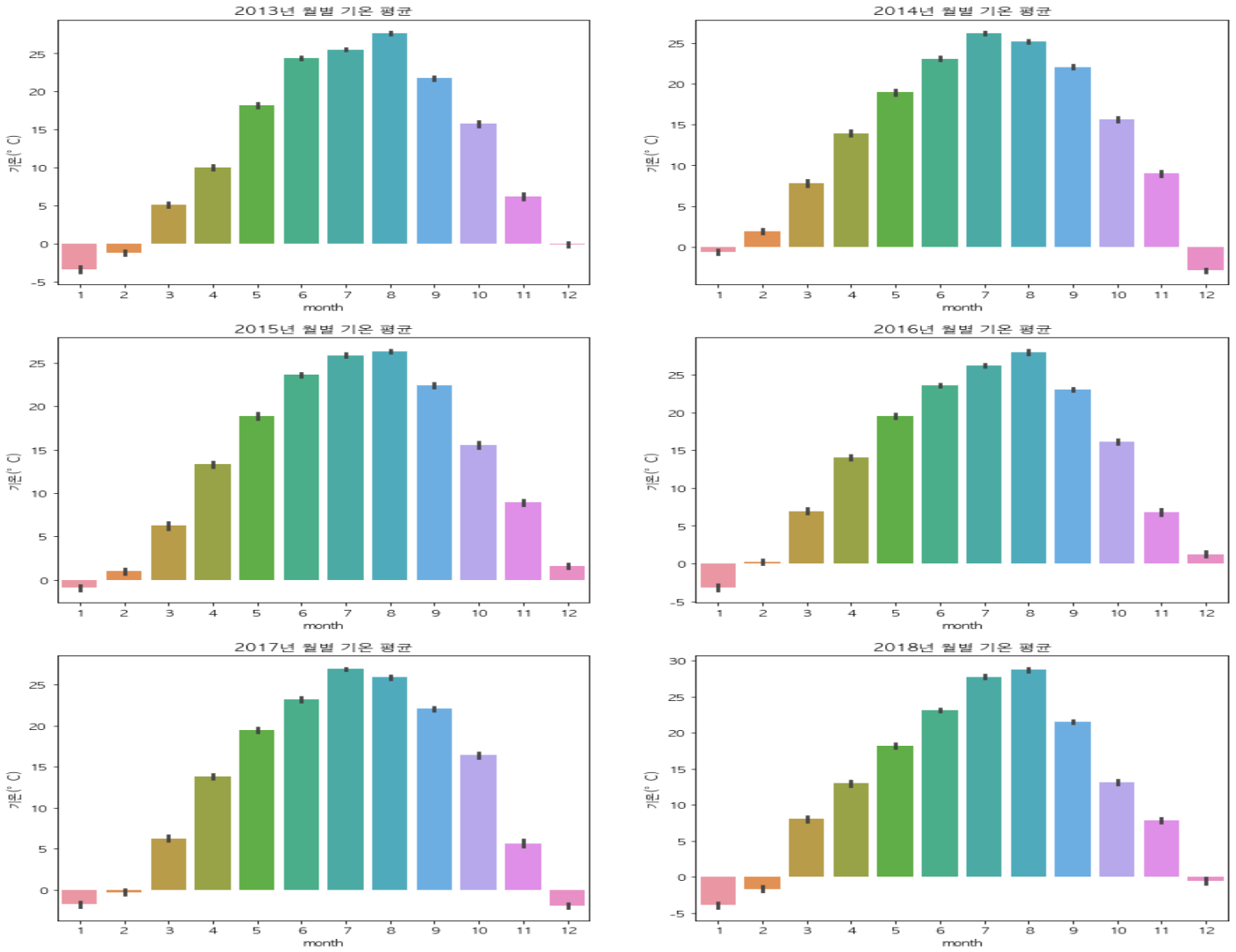
- **2014**년 **12**월과 **2017**년 **12**월의 월평균 공급량이 약 **1800** 정도로 다른 해의 **12**월 월평균 공급량에 비해 높다.

- **2013**년 **1**월과 **2018**년 **1**월의 월평균 공급량이 **2000**이상으로 다른 해의 **1**월 월평균 공급량에 비해 높다.

Part 3

데이터 시각화 및 분석

Train Data - 연도별 월평균 기온 (barplot)



- **2014**년 **12**월과 **2017**년 **12**월의 월평균 기온이 약 **-3°C**정도로 다른 해의 **12**월 월평균 기온에 비해 낮다.

- **2013**년 **1**월과 **2018**년 **1**월의 월평균 기온이 약 **-3°C**정도로 다른 해의 **1**월 월평균 기온에 비해 낮다.

해당 연도의 월 평균 기온이 해당 연도의 공급량에 영향을 미치고 있다.

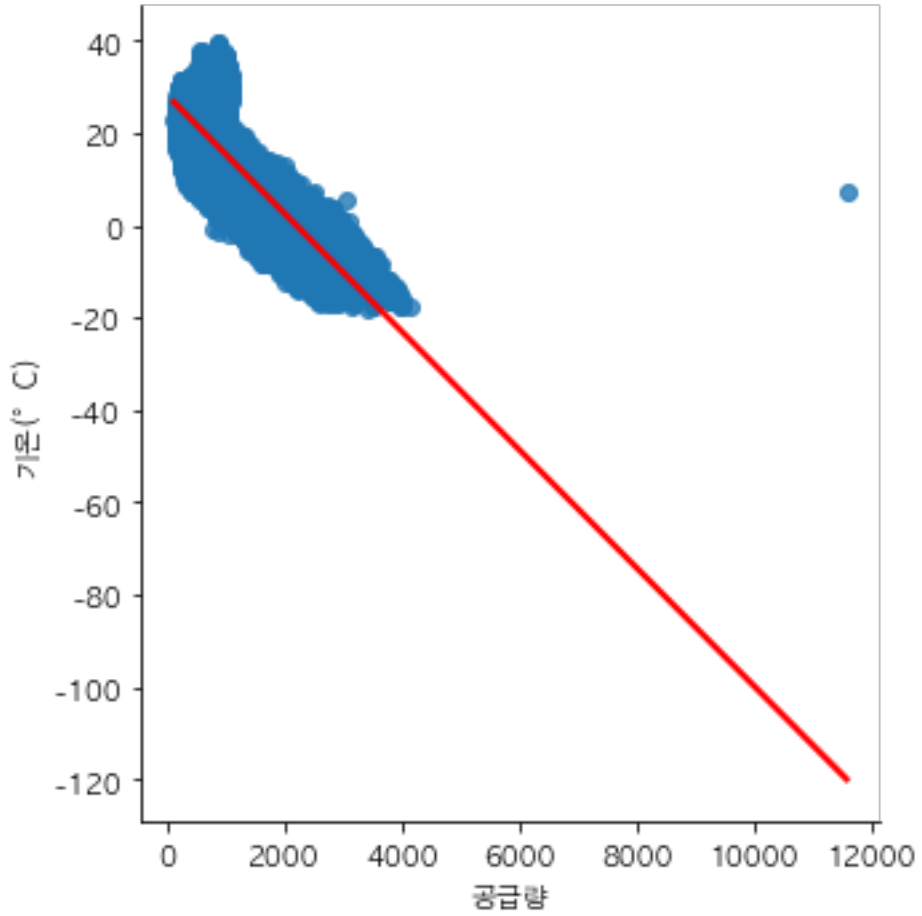
데이터 시각화 및 분석

Train Data - 구분별 데이터간 상관관계(heatmap)

- 기온과 공급량의 음의 상관관계 존재
- 외부 기온데이터를 활용시 유의미한 결과가 예상

데이터 시각화 및 분석

Train Data - 구분별 공급량과 기온의 상관관계 그래프(Implot)

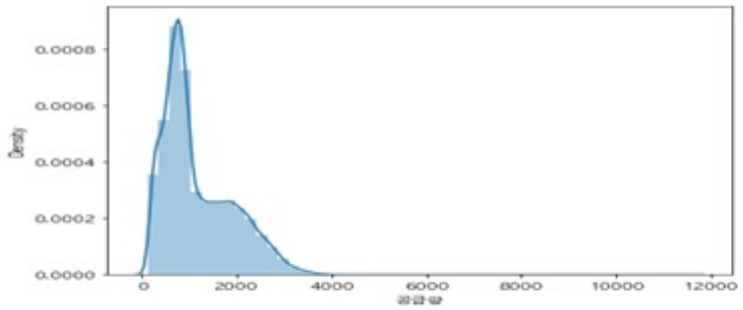


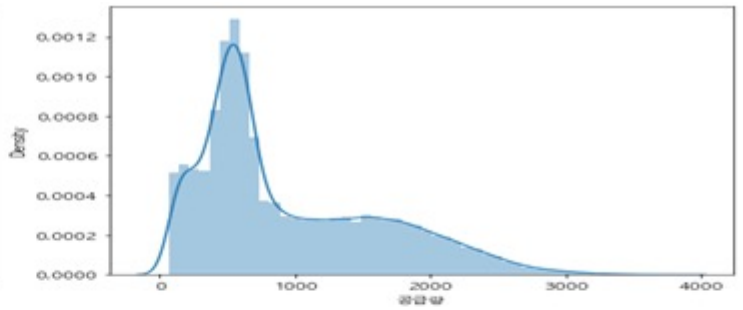
이상치값

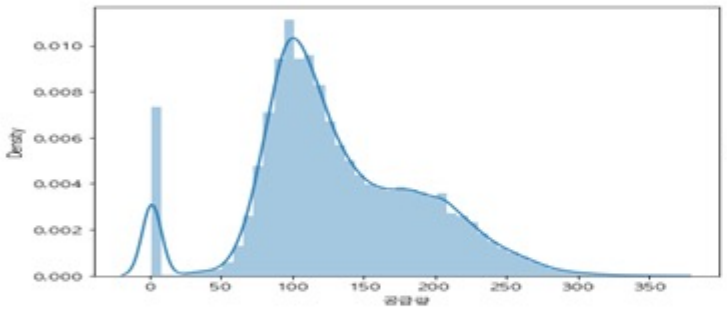
- 기본 데이터의 이상치 값으로 확인
- 정확도 하락의 요인으로 추측

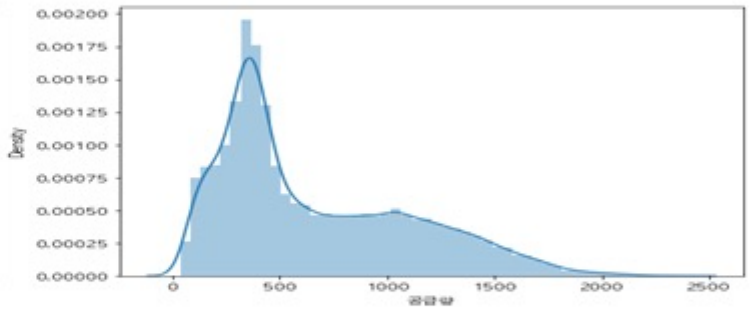
데이터 시각화 및 분석

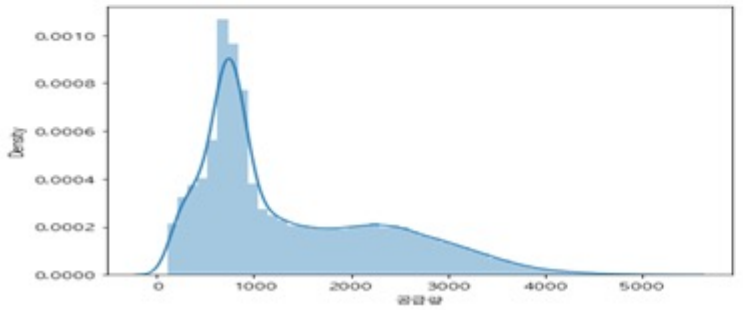
Train Data - 구분별 공급량 분포(displot)

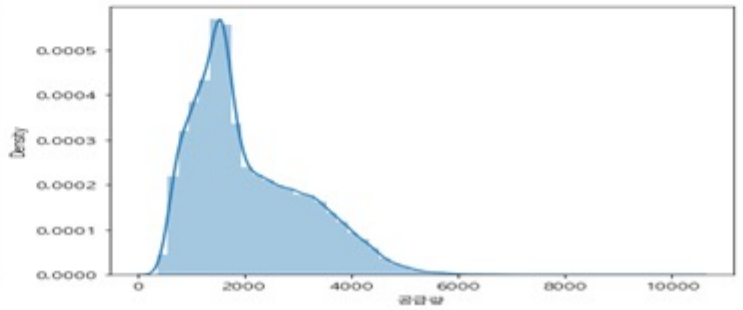




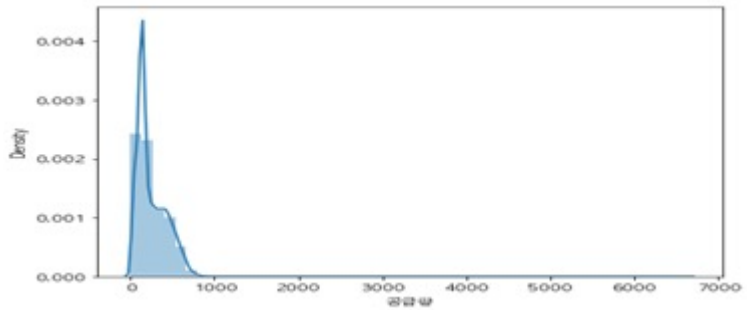






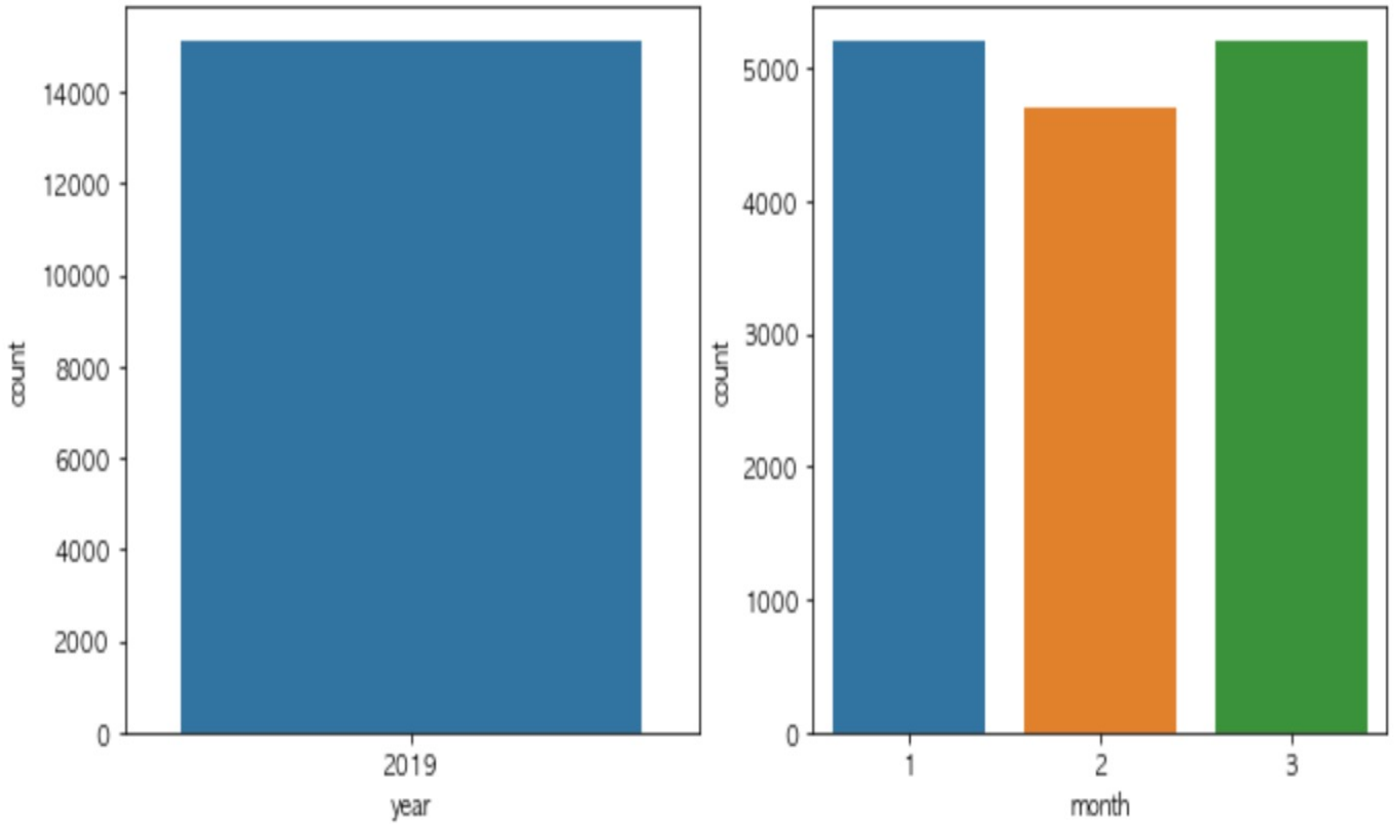


구분마다 공급량의 편차가 크다
- 모델링 학습시 훈련 데이터를 구분별로 나눠야한다고 판단



데이터 시각화 및 분석

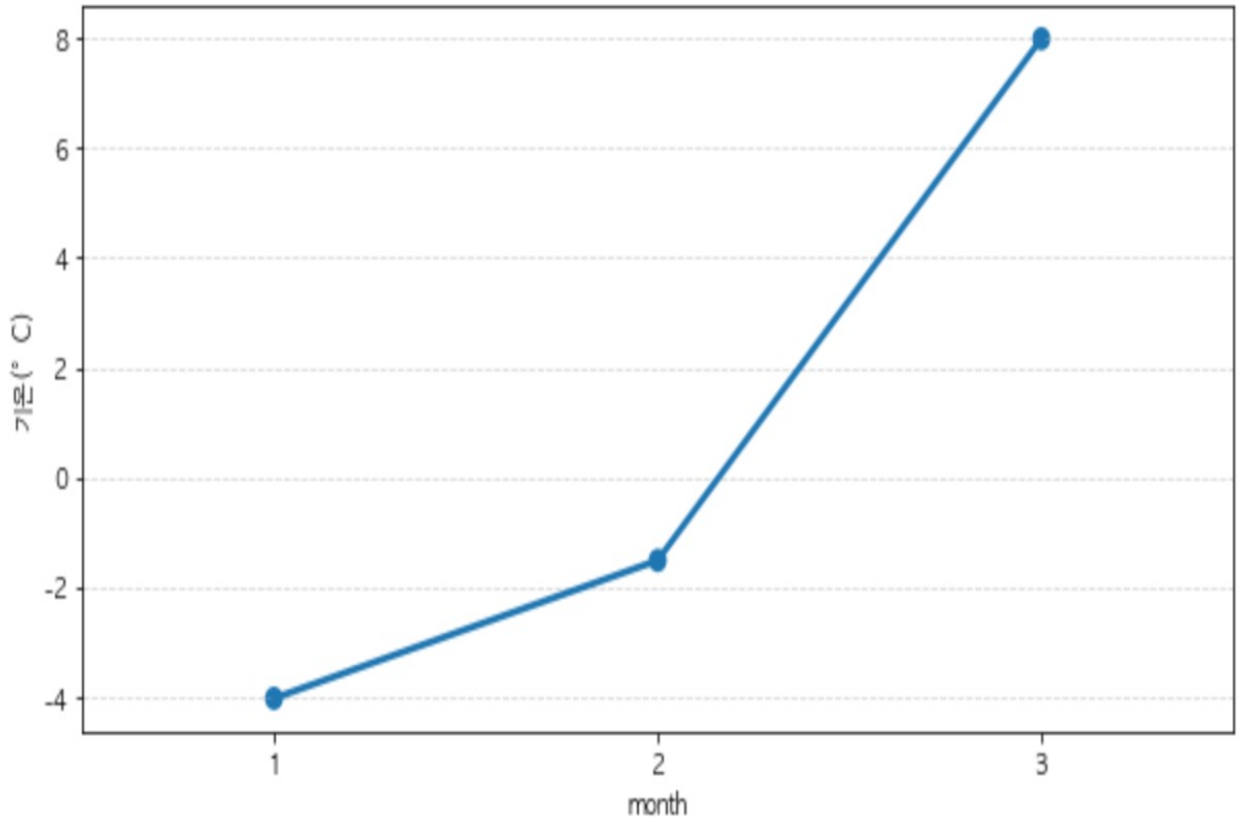
Test Data - 예측 일시 확인



- 테스트 데이터에 날짜 범주 확인
- 예측해야하는 날짜는
  2019년 **1**월**~3**월까지로 확인

데이터 시각화 및 분석

Test Data - 월평균 기온**(pointplot)**



예측한 **2019**년 월평균 기온이
1월부터 **3**월까지 점점 증가하는 것을 확인

Part 4 최적 모델 선정 RandomForestRegressor

1. 기본 제공 데이터를 활용한 머신러닝 모델 구축**(**사용 컬럼**:** “year”, “month”, “day”, “hour”, “weekday”, “구분_int”**)**

Model	Test_size	Train_score	Test_score	MAE	MSE	RMSE	NMAE
Linear Regression	0.4	3.903318	3.807003	705.069958	826141.217153	908.923108	6.687121
Linear Regression	0.8	4.055309	3.809685	705.900011	826363.177704	909.045201	6.669365
DecisionTree Regressor	0.1	100.000000	98.704075	47.882783	11266.179675	106.142261	0.053466
RandomForest Regressor	0.1	99.889329	99.225722	37.458720	6731.217788	82.043999	0.044543
GradientBoosting Regressor	0.6	91.492946	91.507102	178.220452	72694.287407	269.618782	0.951729
XGBRegressor	0.2	98.398153	98.395758	74.465153	13820.674945	117.561367	0.479315
XGBRegressor	0.3	98.457612	98.352964	73.306760	14170.371153	119.039368	0.505775
LGBM Regressor	0.5	97.351338	97.219483	95.461049	23789.734282	154.239211	0.587989
LGBM Regressor	0.8	97.292267	97.111893	97.632771	24811.493274	157.516644	0.546220

Part 4

최적 모델 선정 RandomForestRegressor

2. 19년도 기온 예측 모델 구축(사용 컬럼: “year”, “month”, “day”, “hour”, “weekday”)

Model	Test_size	Train_score	Test_score	MAE	MSE	RMSE	NMAE
Linear Regression	0.5	32.267667	32.162134	578.600880	580411.742242	761.847585	4.224121
Linear Regression	0.9	32.385276	32.184565	580.335669	582621.802422	763.296667	4.149284
DecisionTree Regressor	0.9	100.000000	96.281230	97.234126	31949.020146	178.742888	0.192316
RandomForest Regressor	0.1	99.872323	99.065209	43.288447	8126.644841	90.147905	0.065775
GradientBoosting Regressor	0.7	95.566740	95.492504	137.911102	38648.118572	196.591247	0.945711
GradientBoosting Regressor	0.8	95.378574	95.319769	140.082871	40207.488211	200.518050	0.942795
XGBRegressor	0.2	99.177069	99.143179	53.050791	7381.581264	85.916129	0.469383
LGBM Regressor	0.2	98.660970	98.672492	67.460102	11436.590272	106.941995	0.541551
LGBM Regressor	0.3	98.707716	98.605099	67.409104	12001.113130	109.549592	0.551524

Part 4

최적 모델 선정 RandomForestRegressor

3. 기온 데이터를 추가한 머신러닝 모델 구축**(**사용 컬럼**:** “year”, “month”, “day”, “hour”, “weekday”, “구분_int”, “기온(°C)”**)**

Model	Test_size	Train_score	Test_score	MAE	MSE	RMSE	NMAE
Linear Regression	0.5	32.267667	32.162134	578.600880	580411.742242	761.847585	4.224121
Linear Regression	0.8	32.123475	32.226587	580.008349	582235.882776	763.043828	4.171860
DecisionTree Regressor	0.1	100.000000	98.319467	58.284638	14609.784299	120.870941	0.074891
RandomForest Regressor	0.1	99.872323	99.065209	43.288447	8126.644841	90.147905	0.065775
GradientBoosting Regressor	0.7	95.566740	95.492504	137.911102	38648.118572	196.591247	0.945711
GradientBoosting Regressor	0.8	95.378574	95.319769	140.082871	40207.488211	200.518050	0.942795
XGBRegressor	0.2	99.177069	99.143179	53.050791	7381.581264	85.916129	0.469383
LGBM Regressor	0.3	98.707716	98.605099	67.409104	12001.113130	109.549592	0.551524
LGBM Regressor	0.9	98.704939	98.561491	69.041222	12358.645205	111.169444	0.527669

Part 4

DACON Score

MODEL	COLUMN	기온	Train_score	Test_score	MSE	RMSE	NMAE	DACON
RandomForest Regressor	7	O	0.999	0.991	7742.093434	87.989167	0.073383	0.187347
LightGBM	2	X	·	·	685465	·	·	1.602921
RandomForest Regressor	6	X	0.999	0.992	6601.293267	120.870941	0.049288	0.190851
LGBMRegressor	7	O	0.994	0.994	5410.368952	73.555210	0.459494	0.210881
GradientBoosting Regressor	6	O	95.566740	95.492504	38648.118572	196.591247	0.945711	1.710214

현재까지 가장 좋은 모델

- ML Model : RandomForestRegressor
- DACON Score : 0.187347

Part 5

결론

가장 좋았던 모델, 점수

Model	Test_size	Train_score	Test_score	MAE	MSE	RMSE	NMAE	DACON
RandomForest Regressor	0.1	99.872323	99.065209	43.288447	8126.644841	90.147905	0.065775	0.187347

생각했던 것과 달랐던 점

- 내부 데이터 훈련 모델, 외부 데이터 훈련 모델 정확도 향상 미비
- 기온 데이터에 오류가 있는 것일까?
- 훈련이 충분하지 않았던 것일까?

향후 계획

- 서울의 기온데이터 외 각 지방 데이터를 활용하여 구분별로 서로 다른 지역의 기온 데이터를 훈련시켜 정확도 향상
- 파라미터 튜닝, 정규화 처리 등으로 모델을 개선시켜 정확도 향상
- 딥러닝 모델 추가

감사합니다