경진대회 코드 설명자료

팀 명 : 감자샐러드

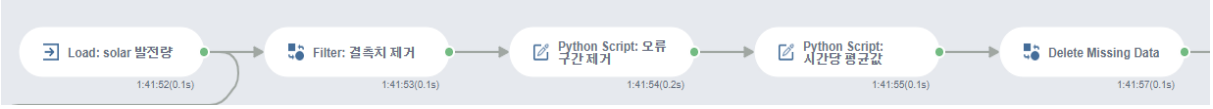
1. 브라이틱스 프로젝트 내 모델 설명

- 태양열과 풍력 발전량을 예측하기 위해 Project 내의 두 가지 모델로 만들어 진행(solar_발전량_예측, wind_발전량_예측)

- solar_발전량_예측

- 발전량 Load 및 전처리

  발전량 데이터를 불러오는 부분과 이상치와 결측치 제거 등 전처리 부분

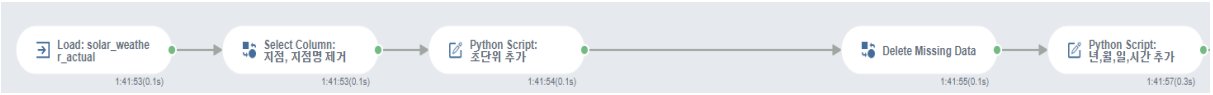


  결측치는 60000 이상의 값은 제거하고 기존 오류로 판단된 구간 외 다른 구간들도 추가하여 10 분단위 데이터를 시간당 평균값으로 산출

```
df = df.loc[
  ~((df['datetime'] >= datetime(2020,12,12)) & (df['datetime'] <= datetime(2020,12,14)))
  & ~((df['datetime'] >= datetime(2021,2,12)) & (df['datetime'] <= datetime(2021,2,15)))
  & ~((df['datetime'] >= datetime(2021,4,1)) & (df['datetime'] <= datetime(2021,4,6)))
  & ~((df['datetime'] >= datetime(2021,7,12)) & (df['datetime'] <= datetime(2021,7,14)))
  & ~((df['datetime'] >= datetime(2021,8,15)) & (df['datetime'] <= datetime(2021,8,17)))
  & ~((df['datetime'] >= datetime(2021,10,2)) & (df['datetime'] <= datetime(2021,10,5)))
  & ~((df['datetime'] >= datetime(2021,11,6)) & (df['datetime'] <= datetime(2021,11,9)))
  & ~((df['datetime'] >= datetime(2021,11,18)) & (df['datetime'] <= datetime(2021,12,2)))
  & ~((df['datetime'] >= datetime(2022,4,13)) & (df['datetime'] <= datetime(2022,4,15)))
]
df['datetime'] = df['datetime'].astype('str')
```

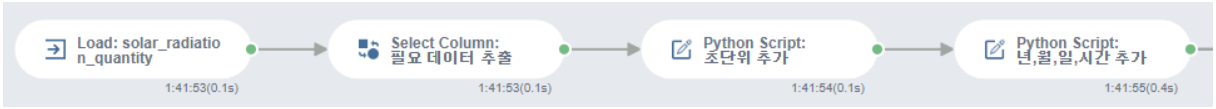
- 실제 기상 데이터 Load 및 전처리

  기상 데이터를 불러와 Select Column 으로 지점, 지점명을 제거하고 datetime 의 초 단위를 추가한다음 Missing Data 를 제거하고 Join 하기 위한 년, 월, 일, 시간을 추가한다.

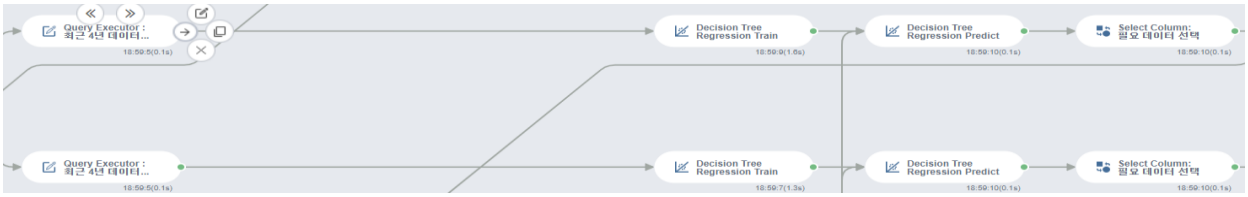


- 일조 데이터 Load 및 전처리

  일조 데이터는 필요 데이터만 추출하고 Join 하기 위한 년, 월, 일, 시간을 추가해준다.



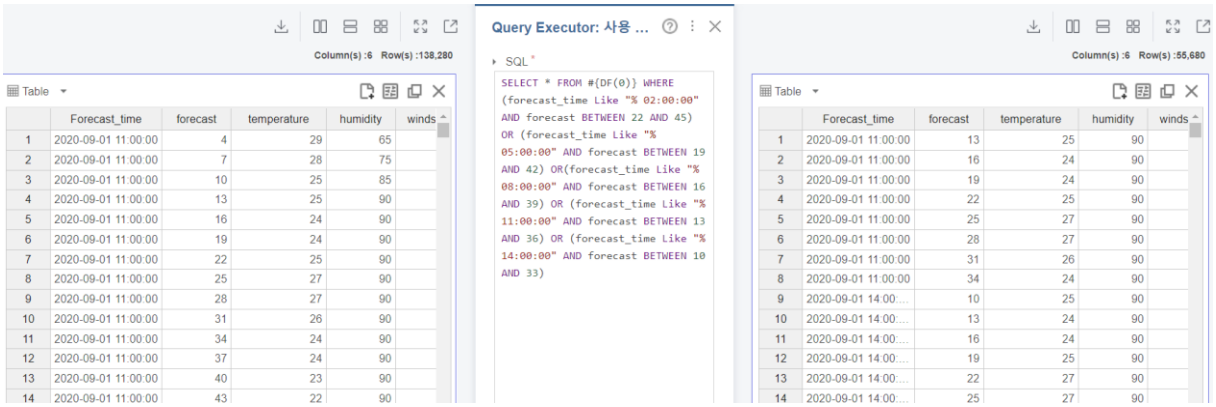
최근 4 년간의 예측에 해당하는 기간을 뽑고 Decision Tree 를 사용해 Validation 구간과 Prediction 구간에 해당하는 기간을 예측한다.



- 예보 데이터 Load 및 전처리

17 시 이전에 예측해야 하므로 17 시이후 데이터는 제거한다.

Query Executor 를 사용해 forecast 에 따른 사용 가능한 forecast_time 을 뽑아준다.



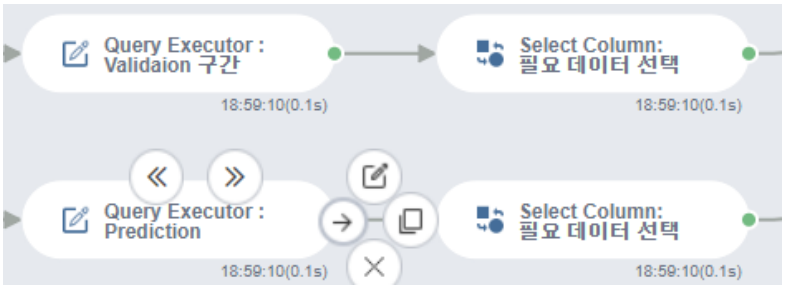
datetime 을 생성하고 예보시간을 1 시간으로 나누어 준다.

```
import pandas as pd
df['forecast_time_'] = pd.to_datetime(df['Forecast_time'])
df['datetime'] = df['forecast_time_'] + df['forecast'].map(lambda x: pd.DateOffset(hours=x))
df['datetime'] = df['datetime'].astype(str)
df.drop(columns=['forecast_time_'], inplace=True)

#df['datetime'] = pd.to_datetime(df['datetime'])
datetime_from = df['datetime'][0]
datetime_to = df['datetime'].values[-1]

every_hour = pd.DataFrame(pd.date_range(start=datetime_from, end=datetime_to, freq='H'), columns=['datetime'])
every_hour['datetime'] = every_hour['datetime'].astype('str')


every_hour = pd.merge(every_hour, df, on='datetime', how='outer')
every_hour['Forecast_time'] = every_hour['Forecast_time'].ffill()
every_hour["temperature"] = every_hour["temperature"].interpolate()
every_hour["humidity"] = every_hour["humidity"].interpolate()
every_hour["windspeed"] = every_hour["windspeed"].interpolate()
every_hour["winddirection"] = every_hour["winddirection"].interpolate()
```

예보 데이터로 뽑은 datetime 의 중복이 있으므로 해당시간대의 temperature, windspeed, winddirection, humidity 의 평균을 구해준다.

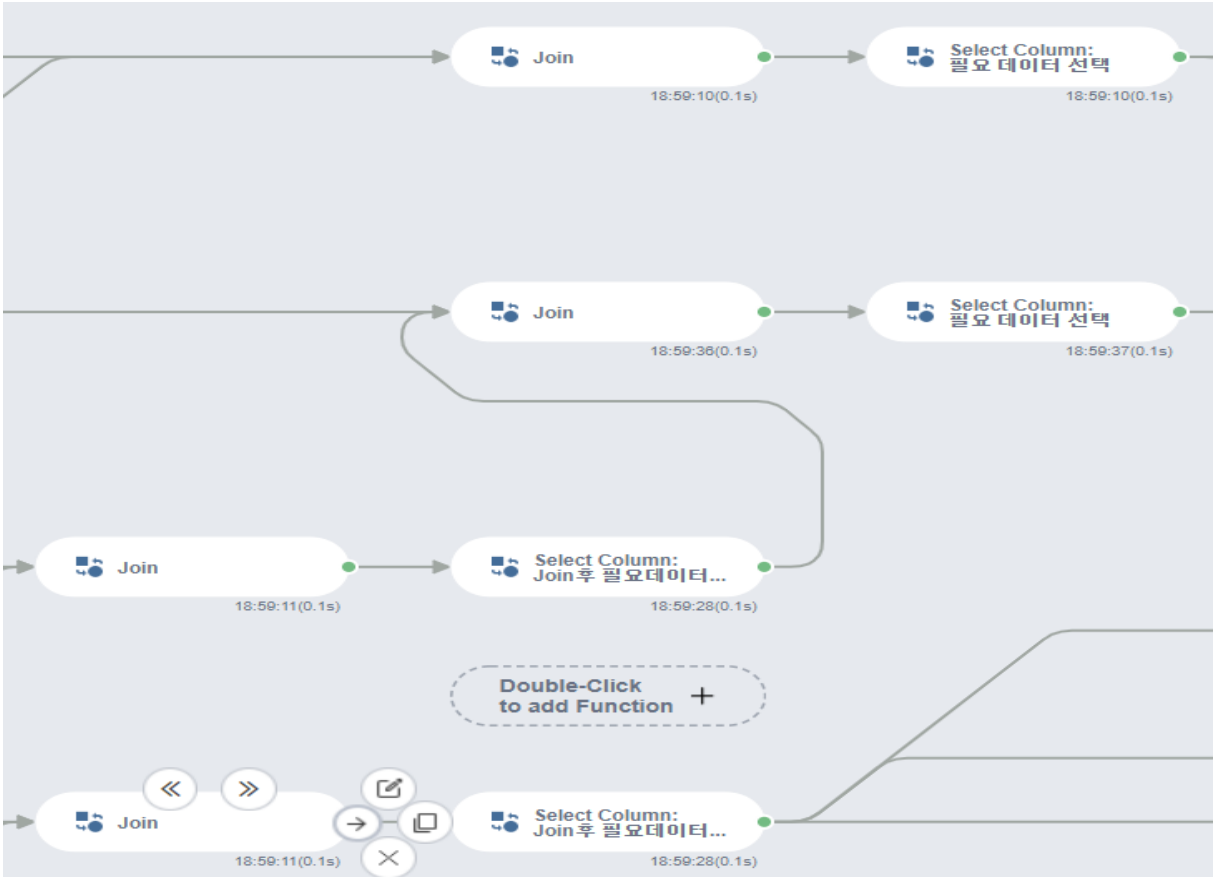
```
import pandas as pd

EH = every_hour.drop(['Forecast_time', 'forecast'], axis = 1)
EH.set_index(pd.to_datetime(EH['datetime']), inplace = True)

EH['temperature'] = EH['temperature'].groupby('datetime').mean()
EH['humidity'] = EH['humidity'].groupby('datetime').mean()
EH['windspeed'] = EH['windspeed'].groupby('datetime').mean()
EH['winddirection'] = EH['winddirection'].groupby('datetime').mean()

df = EH.drop_duplicates()
```

Validation 구간과 Prediction 구간 예측에 사용할 데이터를 추출해준다



- 준비 데이터 Join

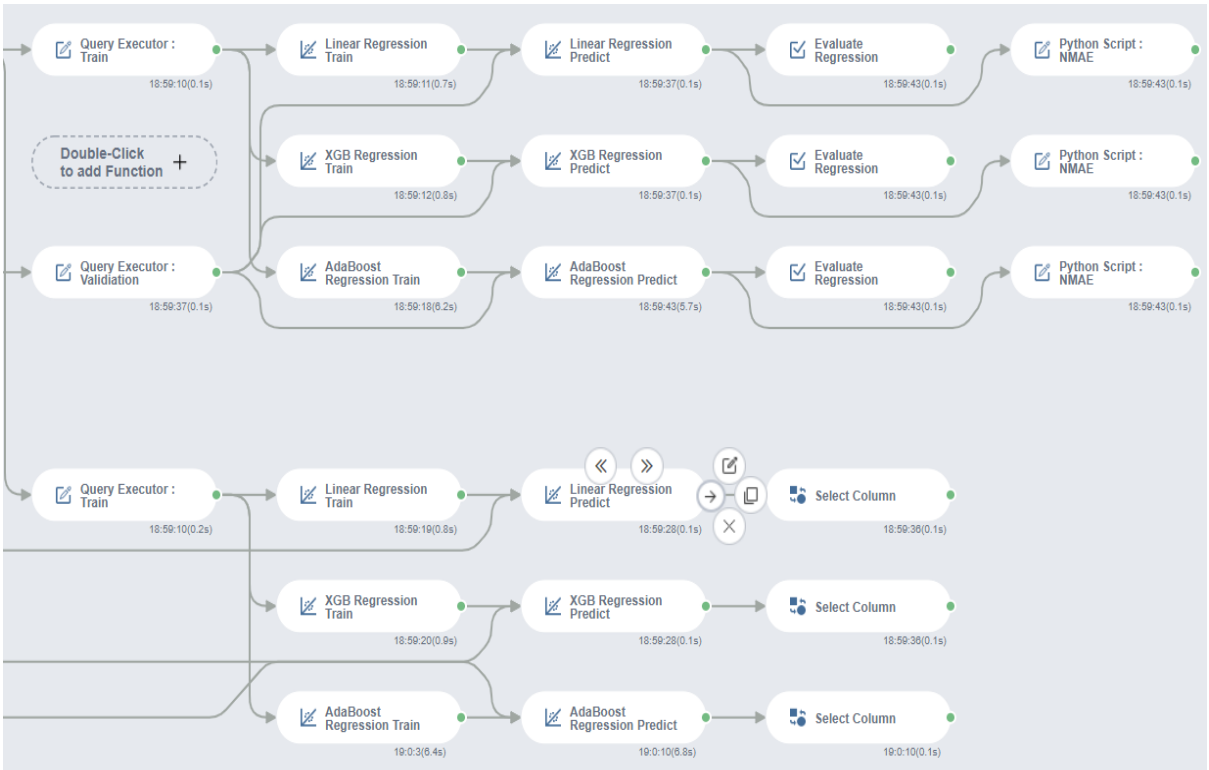
  먼저 일조량 데이터와 기상 데이터, 예측한 일조량 데이터와 예보데이터를 Join 하고 new_weather, new_forecast 를 만들고 Join 된 데이터를 각각 Train, Validation,
  Prediction 을 하기 위한 준비를 한다.



- 모델 학습 및 예측

  모델 성능 평가를 위해 Train Data 를 Train 구간에 넣어 모델을 학습시킨다.

  학습시킨 모델에 Validation, Prediction Data 를 넣어 예측 값을 도출해낸다.



- wind_발전량_예측

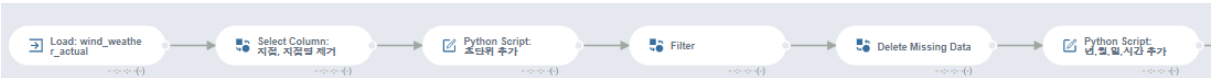
- 발전량 Load 및 전처리

  발전량 데이터를 불러오는 부분과 이상치와 결측치 제거 등 전처리 부분

  결측치는 0 < target < 60,000 값을 제거하고 10 분 단위 데이터를 시간당 평균값으로 산출한 다음 Box plot 을 사용해 이상치를 제거

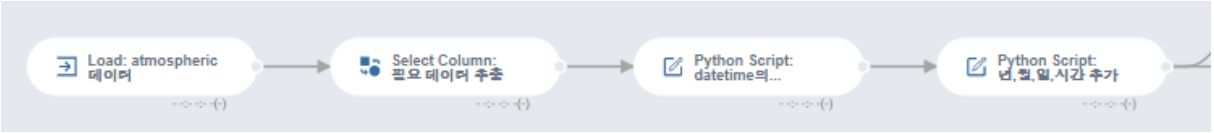
- 실제 기상 데이터 Load 및 전처리

  기상 데이터를 불러와 Select Column 으로 지점, 지점명을 drop 후 Filter 를 통해 EDA 진행 datetime 의 초단위를 추가한 다음 Missing Data 를 제거하고 Join 하기 위한 년, 월, 일, 시간을 추가한다.

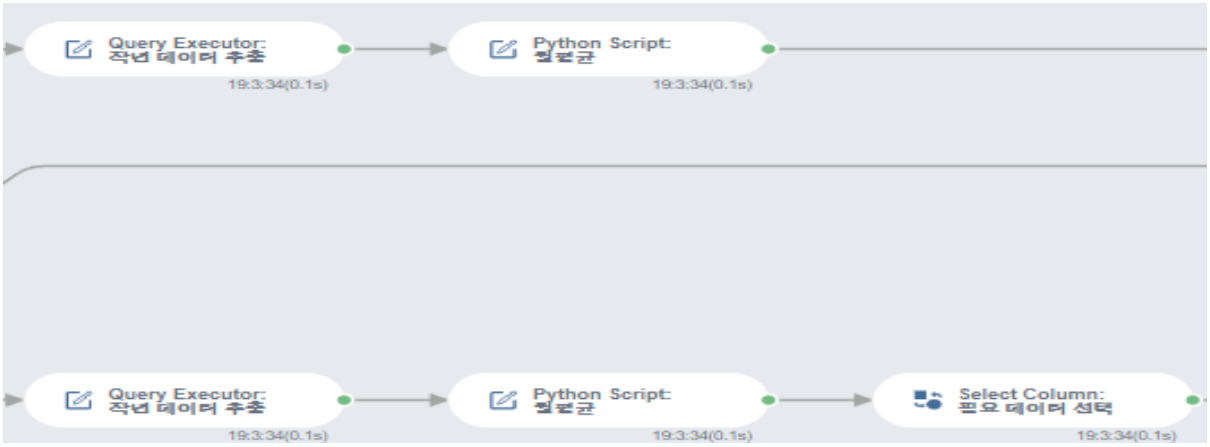


- 기압 데이터 Load 및 전처리

  기압 데이터는 Select Column 으로 지점, 지점명을 drop 후 Join 하기 위한 년, 월, 일, 시간을 추가해준다.



  작년 데이터를 추출하여 월 평균으로 계산 하고 Prediction 은 Select Column 을 사용하여 준비한다.

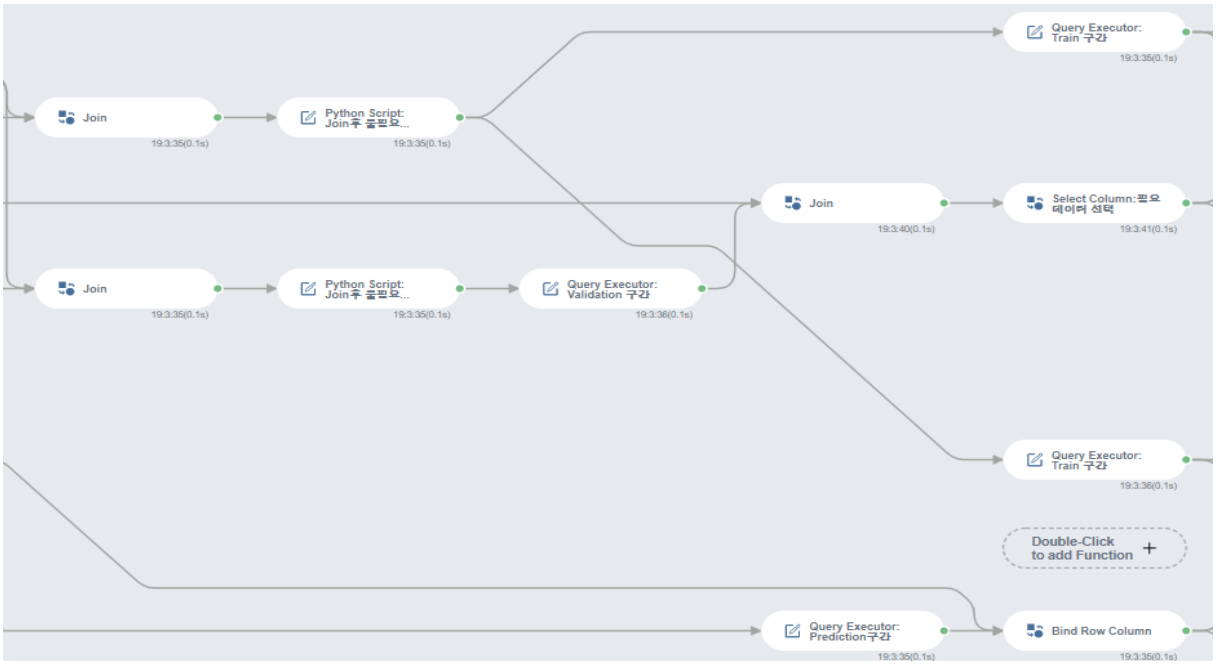


- 예보 데이터 Load 및 전처리

17 시 이전에 예측해야 하므로 17 시 이후 데이터는 제거한다. datetime 이 중복되는 부분은 예보시간이 가장 빠른 값을 선택 후 최신 예보 시간 데이터만 남겨놓는다. 3 시간 단위 예보 시간을 1 시간 단위로 변경하고 forecast_time 과 forecast 제거한다.

- 준비 데이터 Join

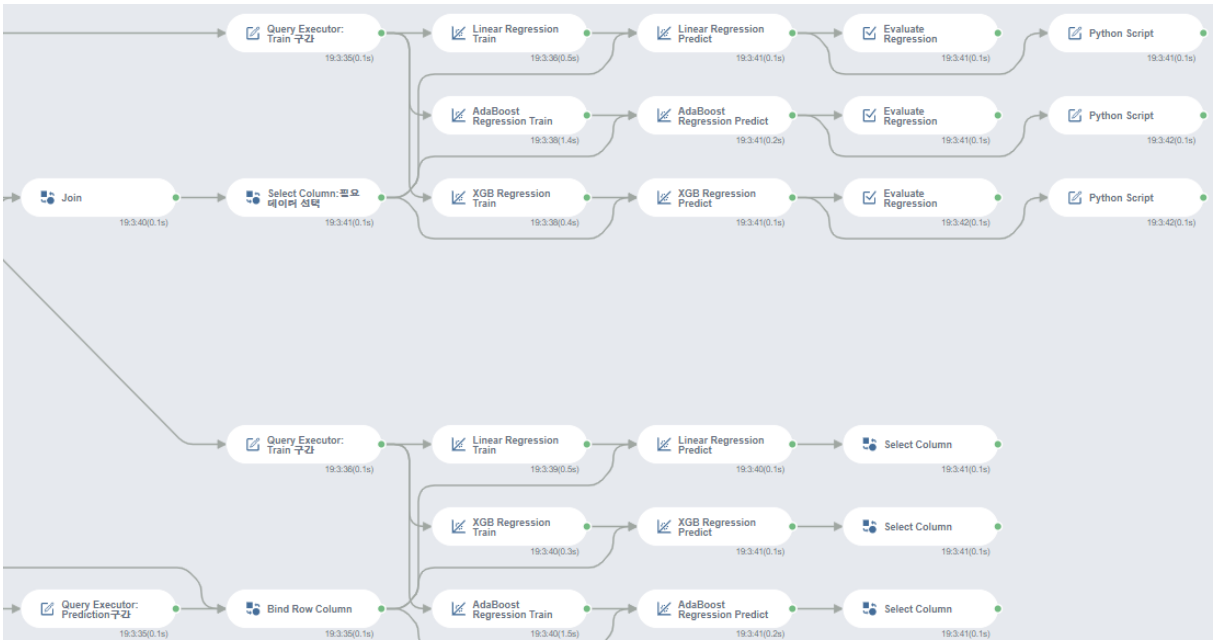
  기압 데이터와 실제 기상 데이터를 Join, 월평균 기압 데이터와 예보 데이터를 Bind 하고 각 데이터를 Train, Validation, Prediction 을 하기 위한 준비를 한다.



- 모델 학습 및 예측

  모델 성능 평가를 위해 Train Data 를 Train 구간에 넣어 모델을 학습시킨다.

  학습시킨 모델에 Validation, Prediction Data 를 넣어 예측 값을 도출해낸다.



2. Input 파일 (활용 데이터) 설명

- Solar_발전량_예측

  - solar_radiation_quantity(출처: 기상청 기상자료개방포털, URL: https://data.kma.go.kr/data/grnd/selectAsosRltmList.do?pgmNo=36)
  'sunshine'은 일조시간으로 태양이 무언가에 의해 가려지지 않고 지표면을 비친 시간에 대한 데이터(단위: hr) 각 변수는 지점, 지점명, 일시(시간), 일조량(hr)로 구분되어 있다.

  - solar_power_2204: 태양광 발전량 데이터

  - weather_solar_actual: 실제 날씨 데이터

  - solar_forecast_weather: 기상예보 데이터

- Solar_power_2204

  - Load: solar_power_2204 데이터 로드
  - Query Executor: 그래프를 통해 이상치 및 오류 데이터 구간 확인
  - Filter: 60000 이상의 이상치 제거
  - Python Script: 그래프와 데이터 확인을 통해 찾아낸 오류 데이터 구간 제거
  - Python Script: 추후 데이터 통합을 위해 datetime 1 시간 단위로 변경
  - Delete Missing Data: 시간단위 변경으로 생긴 null 값 제거

- Python Script: 추후 데이터 통합을 위해 year, month, day, hour 변수 추가

- Weather_solar_actual
  - Load: solar_weather_actual 데이터 로드 및 한글로 된 변수명 영문으로 변경

    (일시 → datetime, 풍속 → windspeed, 풍향 → winddirection, 습도→ humidity)
  - Select column: 필요 없는 변수인 지점명, 지점 제거
  - Python script: 추후 데이터 통합을 위해 datetime 에 초 단위 추가하여 서식 맞춤
  - Python script: 추후 데이터 통합을 위해 year, month, day, hour 변수 추가
- Solar_radiation_quantity
  - Load: solar_radiation_quantity 데이터 로드 및 한글로 된 변수명 변경

    (일시 → datetime, 일조량(hr) → sunshine)
  - Select column: 필요 없는 변수인 지점명, 지점 제거
  - Python script: 추후 데이터 통합을 위해 datetime 에 초 단위 추가 및 year, month, day, hour 변수 추가
  - Query Executor: 년도마다 1 월~ 4 월에 해당하는 데이터터만 추출 또는 5 월~ 6 월에만 해당하는 데이터 추출
  - Decision Tree Regression: 위에 나눈 구간을 각각 Decision tree regression 에서 hour, month, year, day 만을 사용해 뒤에 모델에서 사용할 sunshine 값 예측
- Solar_forecast_weather
  - Load: solar_forecast_weather 데이터 로드
  - Filter: 17 시 이전의 데이터만 가지고 예측을 해야 하기 때문에 17 시 이후의 예보데이터 제거
  - Query Executor: 다음날에 대한 Forecast_time 만 남을 수 있도록

forecast_time 에 대한 forecast 의 조건을 지정하여 데이터 제거

- Python script: 예보 대상 날짜를 나타내는 datetime 변수 생성 및 한 시간 단위로 데이터 변경
- Python script: 필요 없는 변수인 forecast_time 과 forecast 제거 및 datetime 이 같은 데이터에 대해 각 변수의 평균값을 값으로 사용하도록 만들고 중복데이터를 제거하여 각 시간별 데이터가 하나씩만 남도록 변경
- Python script: 추후 데이터 통합을 위해 year, month, day, hour 변수 추가
- Query Executor: datetime 기준으로 2022-01 ~ 2022-04 까지 데이터와 2022-05 ~ 2022-06 까지 데이터로 구분해 Decision Tree predict 구간에 input 후 기존 solar_forecast_weather 과 join

- Weather_solar_actual, solar_radiation_quantity 결합하여 new_weather 데이터 생성
  - Join: 일조시간 데이터와 실제 기상 데이터를 결합
  - 일조시간 데이터가 19 시~7 시까지의 데이터가 없기 때문에 기상 예보 데이터를 기준으로 left join
  - Datetime 을 key 로 결합할 경우 0 시부터 9 시까지의 데이터가 결합이 안되는 현상이 있어 key 는 year, month, day, hour 로 설정
  - Select column: 필요 없는 변수, 중복 변수 제거 및 변수명 변경
  - Replacing missing number: 일조시간 값이 null 인 부분 0 으로 대체
- solar_forecast_weather, solar_radiation_quantity 결합하여 new_forecast 데이터 생성
  - Join: 일조시간 데이터와 기상 예보 데이터를 join 을 활용하여 결합
  - 일조시간 데이터가 19 시~7 시까지의 데이터가 없기 때문에 기상 예보 데이터를 기준으로 left join
  - Datetime 을 key 로 결합할 경우 0 시부터 9 시까지의 데이터가 결합이 안되는 현상이 있어 key 는 year, month, day, hour 로 설정
  - Select column: 필요 없는 변수, 중복 변수 제거 및 변수명 변경

- Replacing missing number: 일조시간 값이 null 인 부분 0 으로 대체
- Decision Tree 에서 예측한 Sunshine 값과 Solar_forecast 를 Join

- Solar_power_2204, new_weather 결합하여 train 데이터 생성
  - Join: year, month, day, hour 를 key 로 inner join
  - Select column: 필요 없는 변수, 중복 변수 제거 및 변수명 변경
  - Query executor: 2022 년 4 월 30 일까지의 데이터를 모델 성능 확인을 위한 train 데이터로 생성
- Solar_power_2204, new_forecast 결합하여 validation 데이터 생성
  - Join: year, month, day, hour 를 key 로 inner join
  - Select column: 필요 없는 변수, 중복 변수 제거 및 변수명 변경
  - Query executor: 2022 년 1 월 1 일부터 4 월 30 일까지의 데이터를 모델 성능 확인을 위한 validation 데이터로 생성
- Query executor: new_forecast 의 2022 년 5 월 1 일부터 6 월 30 일까지의 데이터를 prediction 데이터로 생성

- Wind_발전량_예측

  - Atmospheric(출처: 기상청 기상자료개방포털
  URL: https://data.kma.go.kr/data/grnd/selectAsosRltmList.do?pgmNo=36)
  'Atmospheric pressure'은 기압으로 대기의 압력(기압)은 수평면의 단위 면적당 작용하는 힘이다. (단위: hPa) 각 변수는 지점, 지점명, 일시(시간), 현지기압(hPa)로 구분되어 있다.
  - wind_power_2204: 풍력 발전량 데이터
  - weather_wind_actual: 실제 날씨 데이터
  - wind_forecast_weather: 기상 예보 데이터
- wind_power_2204
  - Load: wind_2204.csv
  - Filter: 발전량이 음수인 구간과 60,000 을 넘어가는 구간은 0 으로 치환하지 않고 결측치라고 판단하여 filter 를 통해 제거

- python script: 10 분 단위의 발전량을 1 시간 단위로 변환
- python script: 사용해 상자수염 그림을 통해 이상치 구간을 제거
- Delete Missing Data: null 값 제거
- python script: 추후 데이터 통합을 위해 year, month, day, hour 변수를 생성

- Weather_wind_actual
  - Load: weather_wind_actual.csv 로드
  - Select column: 필요 없는 변수인 지점명, 지점 제거
  - Python script: datetime 에 초 단위 추가
  - Filter: datetime 17 시부터 23 시까지의 데이터 제거
  - Delete Missing Data: temperature, windspeed, winddirection, humidity null 값 제거
  - Python script: 추후 데이터 통합을 위해 python script 를 통해 year, month, day, hour 변수를 생성
- atmospheric
  - Load: atmospheric.csv 로드
  - Select column: 필요 없는 변수인 지점명, 지점 제거
  - Python script: datetime 의 dtype 을 String 으로 변경
  - Python script: datetime 에 년, 월, 일 추가
  - Query Executor: Validation 및 Prediction 구간에 해당하는 작년 기간 추출
  - Python Script: 해당 월의 평균 기압
  - Select Column: 필요 데이터 추출
- wind_forecast_weather
  - Load: wind_forecast_weather.csv 로드
  - Filter: 17 시 이전의 데이터만 가지고 예측을 해야 하기 때문에 filter 를 통해 17 시 이후의 예보데이터 제거
  - Query Executor: 다음날 이후에 대한 Forecast_time 만 남을 수 있도록 query

executor 를 통해 forecast_time 에 대한 forecast 의 조건을 지정하여 데이터 제거

- Python script: 예보 대상 날짜를 나타내는 datetime 변수 생성
- Query Executor: datetime 을 비교하여 예보 데이터 중 가장 최신 예보 데이터를 선택
- Python script: 3 시간 단위 예보를 1 시간 단위 예보로 변경
- Python script: 필요 없는 변수인 forecast_time 과 forecast 제거
- Python script: 추후 데이터 통합을 위해 python script 를 통해 year, month, day, hour 변수 추가

- Weather_wind_actual, atmospheric 결합하여 new_weather 데이터 생성
  - Join: 대기압 데이터와 실제 기상 데이터를 inner join 을 활용하여 결합
  - Python script: Join 후 사용하지 않는 불필요한 변수들 제거
  - Join: Datetime 을 key 로 결합할 경우 0 시부터 9 시까지의 데이터가 결합이 안되는 현상이 있어 key 는 year, month, day, hour 로 설정
  - Python script 를 사용하여 필요 없는 변수 제거 및 변수명 변경
- wind_2204, new_weather 결합하여 train 데이터 생성
  - Join: Year, month, day, hour 를 key 로 inner join
  - Datetime 을 key 로 결합할 경우 0 시부터 9 시까지의 데이터가 결합이 안되는 현상이 있어 key 는 year, month, day, hour 로 설정
  - Python script: Join 후 사용하지 않는 불필요한 변수들 제거
  - Query executor: 2022 년 2 월 까지의 데이터를 모델 성능 확인을 위한 train 데이터로 생성
  - Query executor: 2022 년 4 월 30 일까지의 데이터를 train 데이터로 생성
- wind_2204, wind_forecast_weather, 예측한 atmosphere 을을 결합하여 validation 데이터 생성

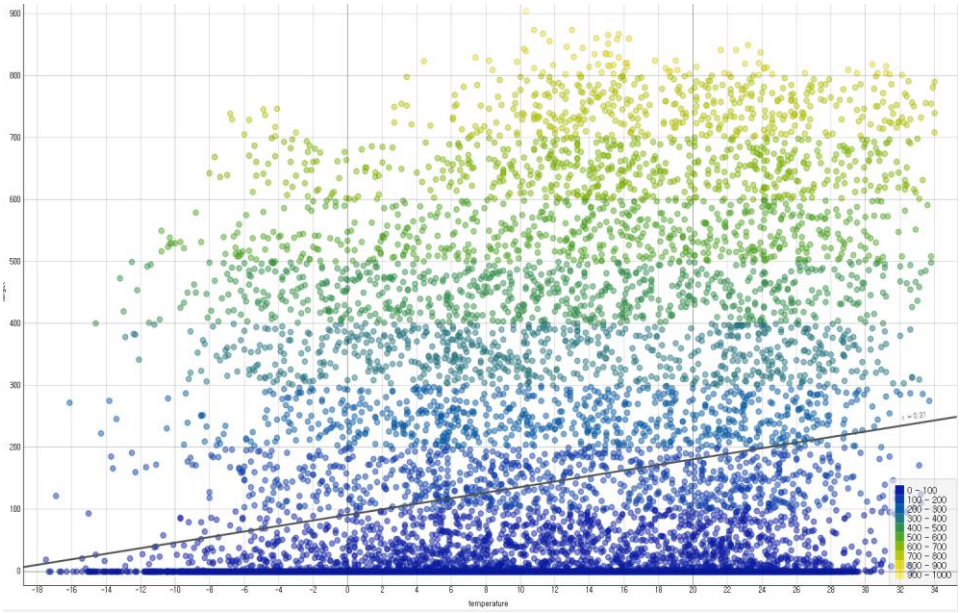
- Join: Year, month, day, hour 를 key 로 inner join
- Python script: Datetime 을 key 로 결합할 경우 0 시부터 9 시까지의 데이터가 결합이 안되는 현상이 있어 key 는 year, month, day, hour 로 설정
- Python script: Join 후 사용하지 않는 불필요한 변수들 제거
- Query executor: 2022 년 3 월 1 일부터 4 월 30 일까지의 데이터를 모델 성능 확인을 위한 validation 데이터로 생성
- Join: 예측한 atmosphere 과 Join 된 wind_2204 와 wind_forecast_weather 을 month, day, hour 를 기준으로 left join
- Select Column: Join 한 데이터에서 필요한 데이터들 만 선택 후 Validation 구간 생성

● Bind Row Column: Prediction 구간을 위해 예측한 atmospheric 과 wind_forecast_weather 를 Bind 하여 Prediction 데이터 생성

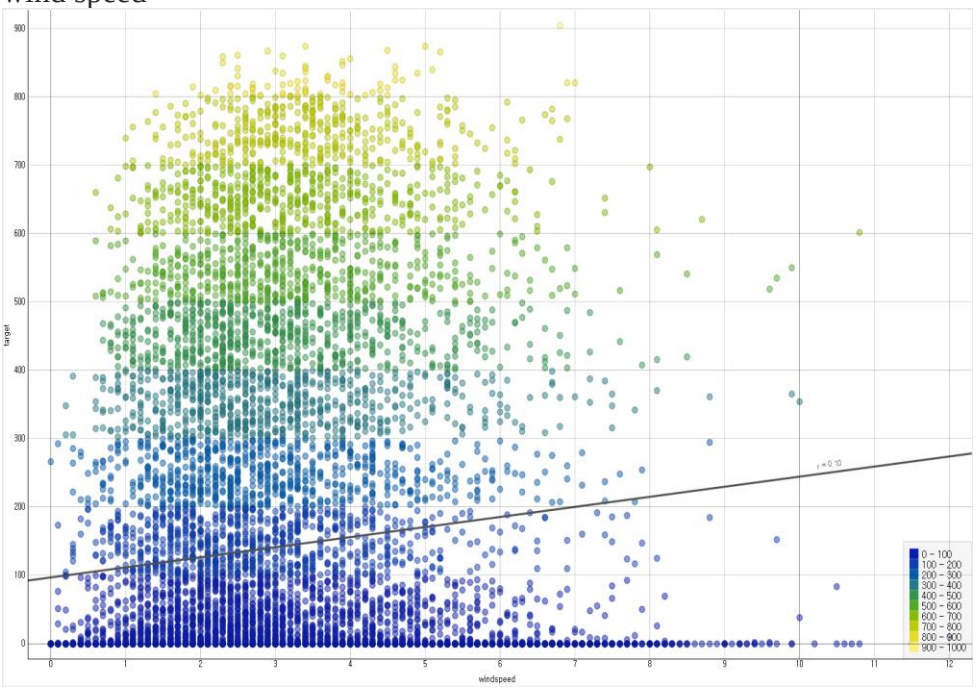
3. 탐색적 자료 분석 (Exploratory Data Analysis)

● solar_발전량_예측
- 그래프를 통해 독립변수(temperature, wind speed, wind direction, humidity, sunshine)와 종속변수(target)간 관계를 확인
- wind direction 과 sunshine 의 경우 카테고리형 데이터의 형상을 띄고 있음
- 각 독립변수와 종속변수간 추세선을 확인해 보았을 때의 r 값
  ⇒ temperature = 0.21, wind speed = 0.10, wind direction = 0.11, humidity = -0.39, sunshine = 0.77
- 태양광 발전량에 가장 큰 영향을 미칠 것으로 예상되는 변수는 일조시간이고 가장 영향력이 없을 것으로 예상되는 변수는 습도

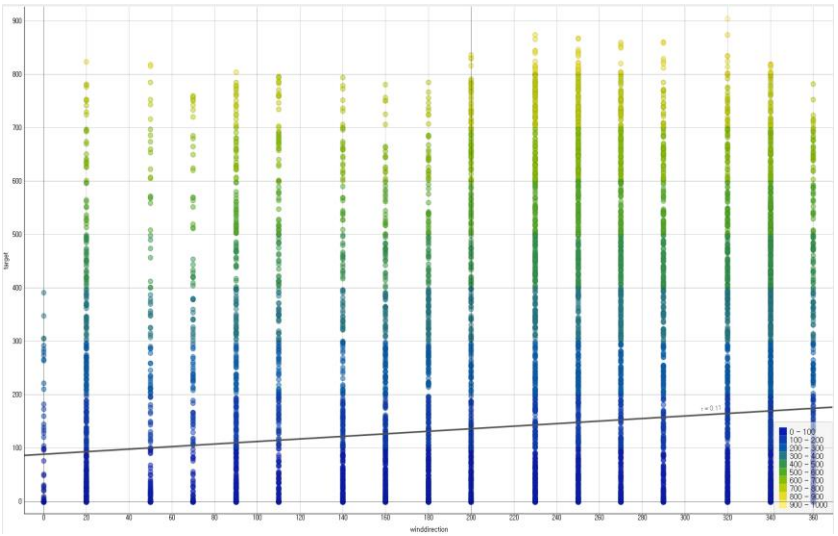
- temperature



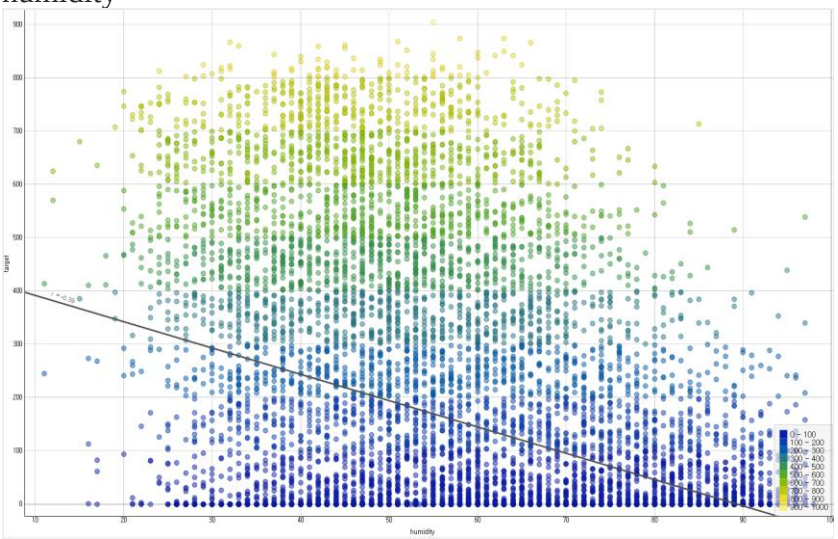
- wind speed



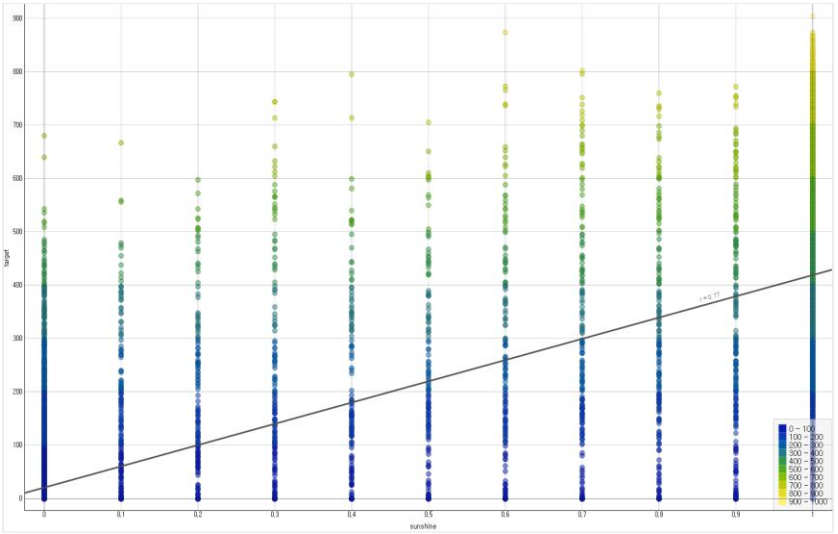
- wind direction



- humidity



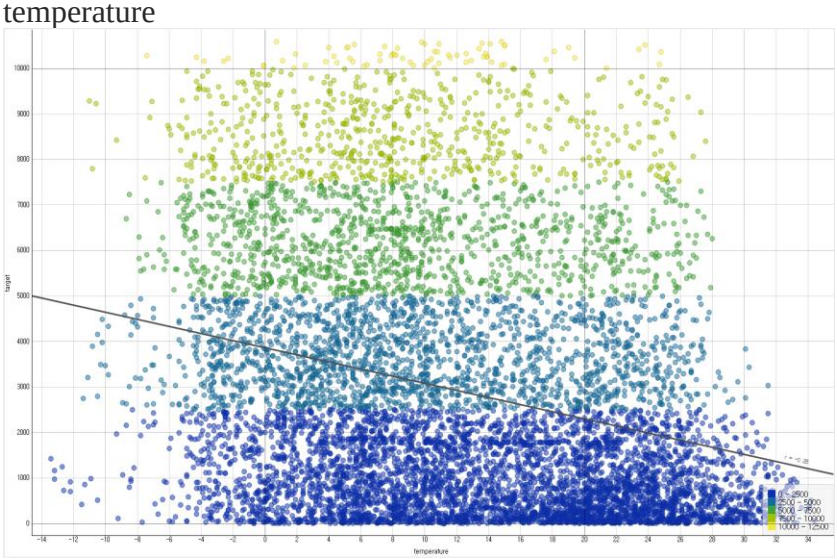
- sunshine



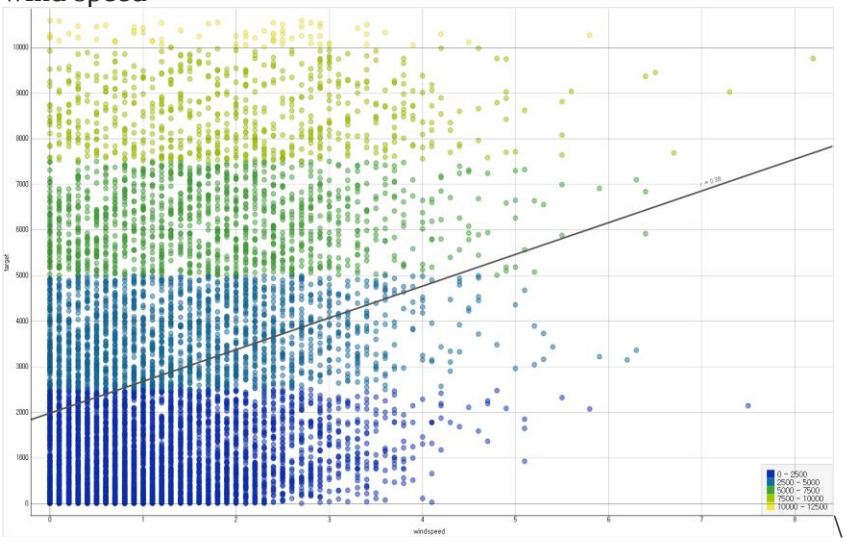
- wind_발전량_예측
  - 그래프를 통해 독립변수(temperature, wind speed, wind direction, humidity, atmospheric)와 종속변수(target)간 관계를 확인
  - wind direction 의 경우 카테고리형 데이터의 형상을 띄고 있음
  - 각 독립변수와 종속변수간 추세선을 확인해 보았을 때의 r 값

    ⇒ temperature = -0.28, wind speed = 0.28, wind direction = 0.01, humidity = 0.04, atmospheric = 0.08
  - 풍력 발전량에 가장 큰 영향을 미칠 것으로 예상되는 변수는 풍속이고 가장 영향력이 없을 것으로 예상되는 변수는 기온

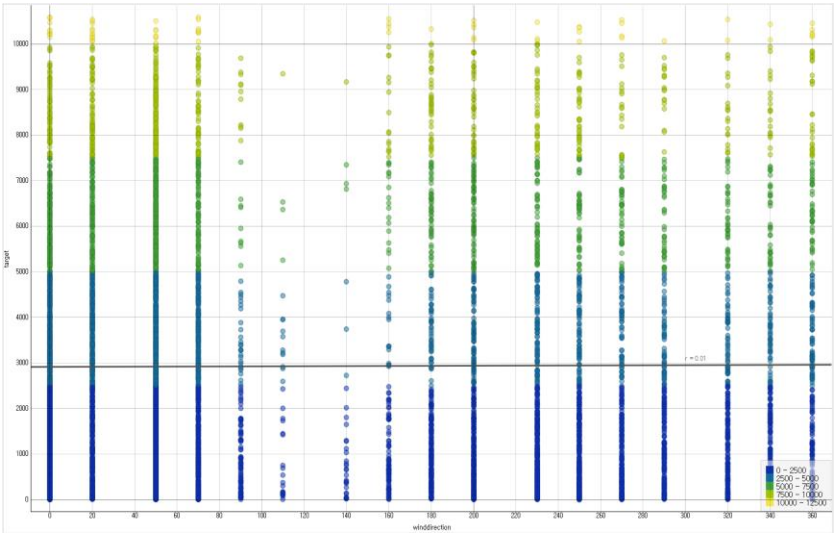
  - temperature



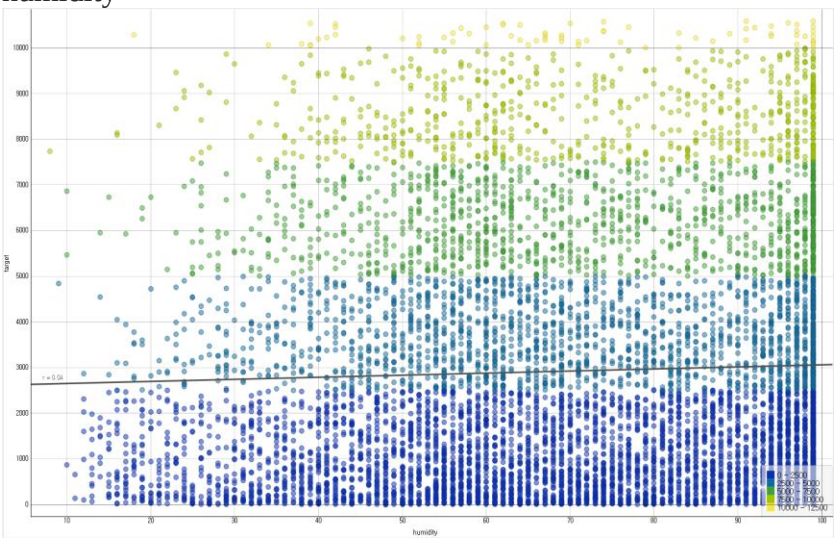
  - wind speed



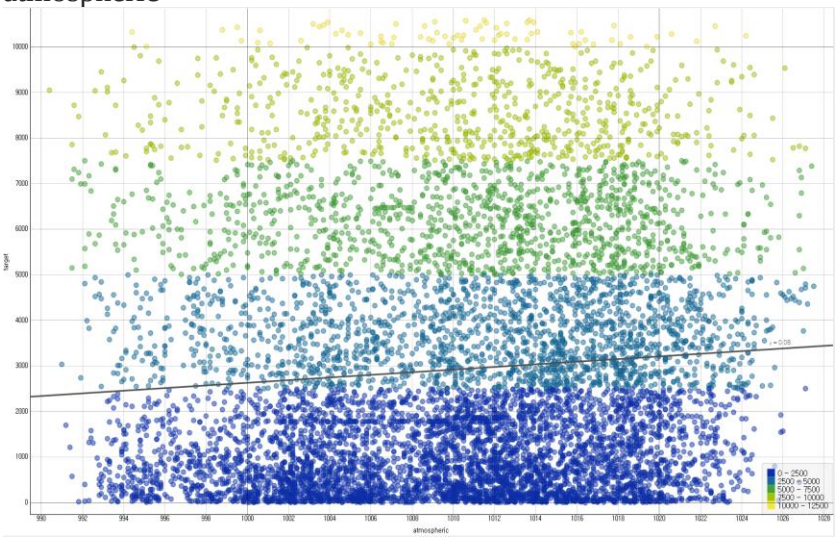
- wind direction



- humidity



- atmospheric



- Atmospheric Pressure
    - 기본적으로 바람은 고기압에서 저기압으로 즉, 기압이 바람을 생성
    - 기압은 공기의 무게 때문에 생기는 압력으로, 고기압과 저기압으로 분류 가능
    - 기압은 날씨에 여러 방면으로 영향을 미침
    - 바람의 운동에너지 공식:

$$운동에너지 = \frac{바람의\ 질량(kg) * 풍속\left(\frac{m}{s^2}\right)}{2} = \frac{단면적(m^2) * 공기\ 밀도\left(\frac{kg}{m^3}\right) * 풍속^3\left(\frac{m}{s}\right)}{2}$$

    - 공기 밀도의 공식:

$$공기밀도 = \frac{\left(표준상태에서의\ 공기밀도\right)*273}{(273+t)} * p\ (p = 기압,\ 표준\ 상태에서의\ 공기밀도 = \frac{1.293Kg}{m^3})$$

    - 두 공식을 정리하면

$$운동에너지 = \frac{단면적(m^2) * 표준상태에서의\ 공기밀도\left(\frac{kg}{m^3}\right) * 풍속^3\left(\frac{m}{s}\right) * 273*p}{2(273+t)}$$

$$(p = 기압,\quad t = 온도\ 표준\ 상태에서의\ 공기밀도 = \frac{1.293Kg}{m^3})$$

    - 위의 공식에서 불변 값은 단면적, 표준상태의 공기밀도, 절대온도로 변경하기 위한 상수 값과 가변 값은 풍속, 온도, 기압이 있음
    - 다중 공선성을 해결하기 위해 공기 밀도와 같이 계산된 식이 아닌 측정된 data를 사용
    - 기압 데이터는 계절성이 있는 데이터로 작년의 5~6월과 올해의 5~6월의 데이터가 비슷할 것으로 예상
    - 작년 5~6월 실제 기압 데이터의 평균을 각각 구해 올해 5월, 6월에 넣어 계산

4. 변수 선택 및 모델 구축 (Feature Engineering & Initial Modeling)

- Solar
    - Feature Columns: Month, Day, Hour, Temperature, Wind speed, Humidity, Sunshine, Wind direction

- Feature Importance 확인 결과 Sunshine, Hour 의 영향력이 가장 큰 것으로 확인
- Linear Regression 의 다중 공선성 확인 결과 VIF 가 5 이상의 Feature 는 존재하지 않음
- Sunshine data 는 계절성과 추세가 존재하지 않는다고 판단해 Decision Tree Regression 을 사용해 Sunshine 을 따로 예측함

- Wind
  - Feature Columns: Month, Day, Hour, Temperature, Wind speed, Humidity, Atmospheric, Wind direction
  - Feature Importance 확인결과 Temperature, Day, Wind speed, Atmospheric, Humidity, Hour, Month, Wind direction 순으로 영향력이 있음
  - Linear Regression 의 다중 공선성 확인 결과 VIF 가 5 이상의 Feature 는 존재하지 않음
  - Linear Regression 의 coef 와 다른 양상 존재
  - Atmospheric 은 계절성을 가지며 작년 월평균 기압을 적용
- AdaBoost Regression
  - Classification 과 Regression 에 모두 적용 가능
  - 과적합의 영향을 덜 받아 에러를 낮추면서 Overfitting 방지
  - Ensemble 모델로 Boosting 방법에 해당
  - 약한 classifier 에 가중치 부여하는 과정을 반복하여 오류를 개선하는 방식
  - Boosting 과정을 반복하면서 최종적으로 강한 classifier 을 생성

5. 모델 학습 및 검증 (Model Tuning & Evaluation)

- Solar
  - Linear Regression, XGB Regression, AdaBoost Regression, Random Forest Regression 중 가장 성능이 좋은 모델을 사용하기 위해 train, validation 데이터로 모델 성능 확인
  - R-squared score 는 AdaBoost, XGB, Random Forest, Linear 순으로 높음
  - NMAE 는 XGB, AdaBoost, Random Forest, Linear 순으로 낮음
  - XGB 파라미터 max_depth = 5, learning_rate = 0.12, number_of_trees = 100 일 경우 R2 = 0.746, NMAE = 8.670 로 성능이 가장 좋았음
  - Ada 파라미터 max_depth = 18, learning_rate = 0.1, number_of_estimators = 55 일 경우 R2 = 0.732, NMAE = 9.182 로 성능이 가장 좋았음

- 성능이 가장 좋았던 AdaBoost, XGB 모델 train, prediction 데이터에 적용
- AdaBoost 의 경우 MAE 값이 34, XGB 의 경우 MAE 값이 38 로 AdaBoost 가 좀더 우세
- 두 모델 모두 태양이 뜨기 전인 새벽 시간대(0 시~5 시, 계절마다 차이 있음)의 발전량은 0 으로 잘 예측하였으나, 태양이 지고 난 후 저녁시간대(6 시~0 시, 계절마다 차이 있음)는 발전량이 0 이 아닌 수로 예측하며 예측 성능이 떨어지는 모습을 보여줌

● Wind

- Linear Regression, XGB Regression, AdaBoost Regression 중 가장 성능이 좋은 모델을 사용하기 위해 train, validation 데이터로 모델 성능 확인
- NMAE 값을 기준으로 모델 성능 확인 결과 AdaBoost Regression 의 성능이 뛰어남
- AdaBoost Regression 파라미터 max_depth = 5, learning_rate = 100, number_of_estimators = 0.5 일 경우 NMAE = 11.7 로 성능이 가장 좋았음

● Hyper-parameter Tuning

- Solar 와 Wind 모두 Hyper-parameter 값 설정은 Grid search 을 사용해 진행
- 모델별로 각각 Grid search 를 진행하여 최적의 모델 확인
- Jupyter notebook 에서 진행했던 것과 Samsung Brightics 에서 모델 적용한 값에서 오차 발생
- 이를 해결하기 위해 근사값을 대상으로 Random Search 의 매개변수 간격을 근사치로 제한 후 진행하여 최적의 Hyper-parameter 값 결정

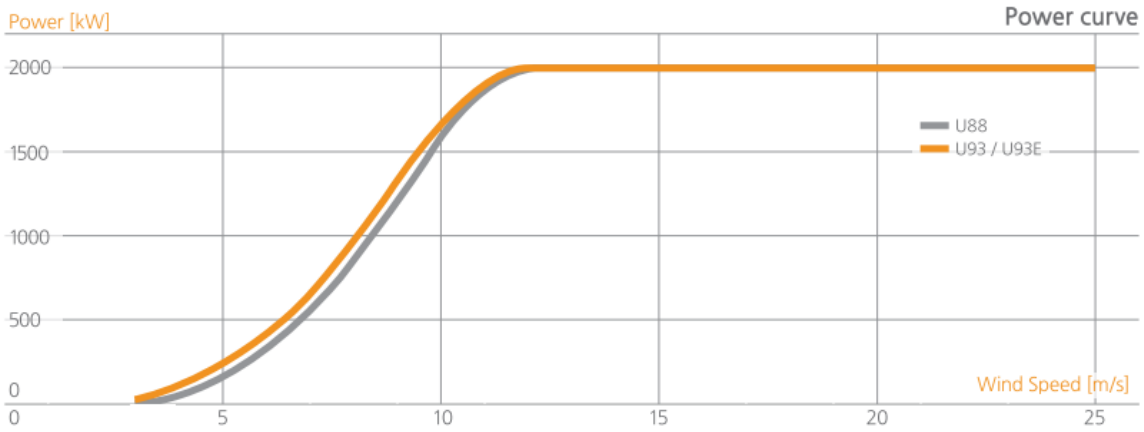
6. 결과 및 결언 (Conclusion & Discussion)

● 결과

- Solar, Wind 모델 모두 AdaBoost Regression 의 성능이 뛰어남
- Solar 의 경우 일조량 dataset 의 추가로 인해 불확실성 감소로 인한 NMAE 값 감소
- Wind 의 경우 MAE 값이 11 에 근사

- 추가적으로 wind speed 에 따른 Box plot 을 그려 이상치 제거를 해도 R-squared 값과 MAE 값의 유의미한 변화가 발생하지 않음
- Wind 의 상대적으로 큰 오차 값은 target 예측에 영향력이 큰 wind speed 의 반복성 부재와 큰 분산 값의 한계로 인해 발생한 것으로 판단

- 결언
  - 일조시간에 대한 data 를 가져와 발전에 대한 불확실성을 감소시킴 Solar 의 경우 일사량에 대한 값도 중요한 요인
  - 평균 강수, 안개와 같이 햇빛을 가리는 기상현상에 대한 data 를 얻어 join 하여 기상예측과 그에 따른 태양광 발전량을 같이 예측할 수 있을 것으로 기대
  - 또한 신인천발전본부 태양광 발전에 사용된 모듈의 면적, 효율에 대한 data 를 사용해 보다 높은 신뢰성 확립 가능
  - wind data 의 경우 광주지역에 대한 데이터 셋이 제공되었지만 실제 화순풍력발전소의 고도는 대략 650m 으로 13m 인 광주 고도와 637m 라는 차이에 tower 높이 80m 까지 높이까지 반영한 조정 필요, 이로 인해 wind target 의 영향력 있는 변수인 wind speed 에 오차 증가로 인해 모델 R-squared 감소 발생
  - 기존에 있는 dataset 을 사용하기 위해 고도에 따른 wind speed 를 보정 후 모델에 적용하면 R-squared 가 감소하는 현상 발생
  - 이를 해결하기 위해 UNISON 사 U93 모델 Wind Sensor 에 wind direction 을 인식하는 것 외에 wind speed 를 측정하는 센서를 부착해 현장에 맞는 dataset 필요



- Power curve 그래프를 보면 11m/s 이후에는 발전량이 2000kW 으로 고정됨을 알 수 있음
- 기존에 사용했던 학습 방법이 아닌 가중이동평균법을 사용하면 0~11m/s 에 대한 Power 예측 정확도가 향상될 것으로 판단
- wind speed 와 target 에 대한 수식을 작성한 후 미분을 통해 wind speed 변화율에 대한 풀이를 진행하여 추가적인 연구를 진행해 볼 수 있을 것이라 기대됨
- 현재 공공데이터 포털에서 예측에 관한 기압데이터와 일조시간을 제공하고 있는 데이터는 없는 것으로 보인다.
- 하지만 이 두 data 는 실제 모델에서 영향력이 높은 변수이므로 작년의 평균치 또는 Decision Tree Regression 으로 예측하여 사용했다.
- 실제로 기상청에서 일조, 기압데이터를 예측할 때는 더 많은 변수를 사용할 것으로 예상되지만 사용하는데 기간이 제한되며 데이터 공급에 제약이 있어 보다 낮은 정확도가 나온 것으로 예상한다.
- 예측할 기간의 직전 달 데이터가 들어가면 정확도를 높일 수 있을 것으로 기대한다. 하지만 5 월, 6 월 데이터를 알 수 없다는 한계점이 있어 이를 위해서는 Expanding Window 를 사용하거나 Sliding Window 를 사용해 최신의 날씨를 추가적으로 학습하면서 target 을 예측하면 더 좋은 결과가 나올 것으로 기대함