기상청

[Python을 활용한 분석교육실습]

ASOS(종관기상관측) 자료를 활용한 서리 발생 모형 분석 사례

기상청

기상기후
빅데이터 분석 플랫폼

분석 개요

분석 교육 실습 주제인 ASOS(종관기상관측장비)와 현상(이슬,서리)에 대해 알아봅니다.

● 종관기상관측장비(ASOS)

- 기상청은 서울기상관측소를 비롯하여 전국 102개소의 종관기상관측장비(ASOS)와 무인으로 운영되는 510개소의 자동기상관측장비(AWS)를 이용하여 지상기상관측업무를 수행하고 있다.(2018년 기준) 종관기상관측 장비(ASOS)는 지방청, 지청, 기상대, 관측소 등에 설치되어 기상 현상 관측 및 국제 전문을 통한 자료 공유 등의 관측 업무를 수행한다.
- 기압, 기온, 풍향, 풍속, 습도, 강수량, 강수 유무, 일사량, 일조시간, 지면 온도, 초상 온도, 지중 온도, 토양수분, 지하수위 14개 요소에 대해서는 종관기상관측장비(ASOS)로 자동 관측하고, 시정, 구름, 증발량, 일기 현상 등은 일부 자동과 목측(目測)으로 관측한다.
- 현상은 일정 시간(오전, 오후) 동안 발생한 일기 현상을 관측한 것이다. 일반적으로 일기 현상을 관측 하기 위해 목측 또는 CCTV를 통해 원격 관측한다. 특히 이슬과 서리의 경우 다양한 기상요소(기온, 습도, 풍향, 풍속 등)에 의해 영향을 받으며 시간에 따라 변화한다.

● 이슬

- 야간에 복사냉각으로 인하여 이슬점 밑으로 지면 공기의 온도가 내려갈 때 (그러나 기온은 영상인 경우) 풀과 지면의 모든 다른 물체에 응결되어 맺혀있는 물로서, 복사냉각으로 차가워진 지면 물체와 접촉된 공기가 냉각되어 과포화가 되고 과포화 된 양의 수증기가 물로 변화된 것이다.
- 이슬점 온도가 영하인 경우에는 하얀 서리가 형성될 수 있다.

● 서리

- 대기 중의 수증기가 지면이나 지상에 있는 물체의 표면이나 설면 등에 승화하여 생기는 침상, 선상 등의 얼음 결정을 말하며, 때로는 부정형으로 생기기도 한다.
- 서리는 이슬이 만들어질 때와 마찬가지의 원인으로 지표면이 냉각되어, 지면 온도가 0℃ 이하일 때 생기며, 서리가 생기는 때에는 식물의 잎등의 세포조직이 동결이나 저온으로 인해 손상되기 때문에 농작물에 피해가 발생한다.

분석 시나리오

실습할 예제는 ASOS(종관기상관측장비) 기상 데이터와 비기상 데이터를 활용하여 현상(이슬, 서리) 관측을 산출하는 모형을 구축하는 것입니다. 실습 예제를 통해 현재 목측 또는 CCTV를 통해 원격으로 관측하고 있는 현상(이슬,서리) 발생을 자동으로 산출하는 모형을구축해봅니다.

● 현상(이슬, 서리)관측 산출 모형 구축 시나리오

[분석 활용 데이터] **[Analytics]**

기상 데이터(ASOS)

- 서리 발생 유무
- 기온, 강수, 일조, 일사, 풍속, 풍향, 이슬점 온도, 기압, 습도, 운량, 운고, 지면온도
- (파생변수) 관측시간 (최고 - 최저) 기온

비기상 데이터

- (지리적 변수) 수원과의거리, 산맥 형성 정도, 관측 지점의 위도와 고도

현상(이슬, 서리) 영향 변수 선택

정의	변수 중요도가 높은 변수를 기준으로 순서대로 상관계수가 0.45 이상인 변수를 제거하여 후보 변수를 도출하고, 모든 후보 변수 조합에서 가장 최적의 예측력을 갖는 변수 조합을 선택
분석방법	Gradient Boosting Model Pearson Correlation
분석결과	• 현상(이슬, 서리) 영향변수

현상(이슬, 서리) 산출 모형 구축

정의	머신러닝 기법(GBM)을 활용하여 현상(이슬, 서리)을 설명하는 모형 구축
분석방법	Gradient Boosting Model
분석결과	•현상(이슬, 서리) 모형

※본 실습 예제는 서리 발생 모형 분석을 중심으로 합니다.

분석 절차

실습은 데이터 로딩, 데이터 탐색, 데이터 처리, 모형 구축, 모형 검증의 단계에 따라 진행됩니다.

● 현상(이슬, 서리)관측 산출 모형 구축 절차

	[실습 설명]	[실습 단계]
1 데이터 로딩	분석환경을 설정하고 분석에 필요한 기상 데이터 및 비기상데이터를 로딩하여 분석에 필요한 데이터를 준비하는 단계	1. 분석 환경 설정 및 패키지로딩 2. 데이터 불러오기 3. 데이터 결합하기 4. 데이터 타입 변환하기
2 데이터 탐색	분석 데이터의 요약 통계를 살펴보고 상자그림 및 히스토그램을 통해 데이터의 분포를 확인하는 단계	1. 요약 통계 보기 2. 상자 그림 그리기 3. 히스토그램 그리기
3 데이터 처리	이상치를 처리하고 결측치를대체하며, 파생변수를 생성하여 최종 데이터셋을 구성하는 단계	1. 이상치 처리 2. 결측치 처리 3. 파생변수 생성
4 모형 구축	현상(이슬, 서리) 영향 변수를 선택하고 모형의 파라미터를 설정하여 현상(이슬, 서리)관측 산출 모형을 구축하는 단계	1. 변수 선택 2. 모형 구축
5 모형 검증	최종 구축한 산출 모형의 성능을 검증하는 단계	1. 모형 성능 및 예측력 검증 2. 지점별 예측력 검증

※본 실습 예제는 서리 발생 모형 분석을 중심으로 합니다.

분석 데이터(1/2)

현상(이슬, 서리)관측산출 모형 구축에 사용된 파일 및 변수 정보를 확인합니다.

● 현상(이슬, 서리)관측 산출 모형 구축에 사용된 파일 및 변수정보

파일명	파일 설명	변수명	변수 설명	형식	예제
ASOS_DATA	종관기상 관측장비 (ASOS) 관측 자료	STNID	관측 지점 번호	Int	95
		TM	현상 관측 날짜	Char	2009-05-27 00
		MIN_TA	최소 기온	Num	13.6
		MAX_TA	최대 기온	Num	18.0
		AVG_TA	평균 기온	Num	15.54286
		STD_TA	기온의 편차	Num	1.5909181
		MIN_SS	최소 일조	Num	0
		MAX_SS	최대 일조	Num	0.8
		AVG_SS	평균 일조	Num	0.11428572
		STD_SS	일조의 편차	Num	0.27994169
		MIN_SI	최소 일사	Num	0
		MAX_SI	최대 일사	Num	0.00
		AVG_SI	평균 일사	Num	0.00000000
		STD_SI	일사의 편차	Num	0.0000000
		AVG_RN	평균 강수	Num	0.00000000
		SUM_RN	총 강수	Num	0.000
		MIN_WS	최소 풍속	Num	0.6
		MAX_WS	최대 풍속	Num	0.57
		AVG_WS	평균 풍속	Num	1.8285714
		STD_WS	풍속의 편차	Num	1.6271923
		MIN_WD	최소 풍향	Num	20
		MAX_WD	최대 풍향	Num	200
		AVG_WD	평균 풍향	Num	90.0000
		STD_WD	풍향의 편차	Num	61.41196
		MIN_TD	최소 이슬점 온도	Num	10.1
		MAX_TD	최대 이슬점 온도	Num	11.2
		AVG_TD	평균 이슬점 온도	Num	10.48571
		STD_TD	이슬점 온도의 편차	Num	0.3226167
		MIN_PA	최소 현지기압	Num	996.3
		MAX_PA	최대 현지기압	Num	997.4
		AVG_PA	평균 현지기압	Num	996.6857
		STD_PA	현지기압의 편차	Num	0.4015336

※본 실습 예제는 서리 발생 모형 분석을 중심으로 합니다.

분석 데이터(2/2)

현상(이슬, 서리)관측산출 모형 구축에 사용된 파일 및 변수 정보를 확인합니다.

● 현상(이슬, 서리)관측 산출 모형 구축에 사용된 파일 및 변수정보

파일명	파일 설명	변수명	변수 설명	형식	예제
ASOS_DATA	종관기상 관측장비 (ASOS) 관측 자료	MIN_PS	최소 해면기압	Num	1014.4
		MAX_PS	최대 해면기압	Num	1015.7
		AVG_PS	평균 해면기압	Num	1014.914
		STD_PS	해면기압의 편차	Num	0.4882244
		MIN_PV	최소 수증기압	Num	12.4
		MAX_PV	최대 수증기압	Num	13.3
		AVG_PV	평균 수증기압	Num	12.68571
		STD_PV	수증기압의 편차	Num	0.2747913
		MIN_HM	최소 상대습도	Num	61
		MAX_HM	최대 상대습도	Num	81
		AVG_HM	평균 상대습도	Num	72.71429
		STD_HM	상대습도의 편차	Num	7.795865
		MIN_TCA	최소 전운량	Num	0
		MAX_TCA	최대 전운량	Num	7.079385
		AVG_TCA	평균 전운량	Num	1.575071
		STD_TCA	전운량의 편차	Num	2.627426
		MIN_CA	최소 중하층운량	Num	0
		MAX_CA	최대 중하층운량	Num	4.381838
		AVG_CA	평균 중하층운량	Num	1.048775
		STD_CA	중하층운량의 편차	Num	1.701266
		MIN_CH	최소 운고	Num	8.990096
		MAX_CH	최대 운고	Num	19.50699
		AVG_CH	평균 운고	Num	13.05865
		STD_CH	운고의 편차	Num	3.070608
		MIN_TS	최소 지면 온도	Num	12.6
		MAX_TS	최대 지면 온도	Num	16.3
		AVG_TS	평균 지면 온도	Num	14.04286
		STD_TS	지면 온도의 편차	Num	1.2590891
		Y	서리 발생 여부	**Int**	0
GEO_DATA	종관기상 관측장비 (ASOS) 관측 지점별 이력 정보	STNID	관측 지점 번호	Int	136
		LAT	관측 지점의 위도	Char	36.57293
		HT	관측 지점의 고도	Char	140.1
		MOUNTAIN	산맥 형성의 정도	Int	4
		WATER	수원과의 거리	Num	2300

기상청

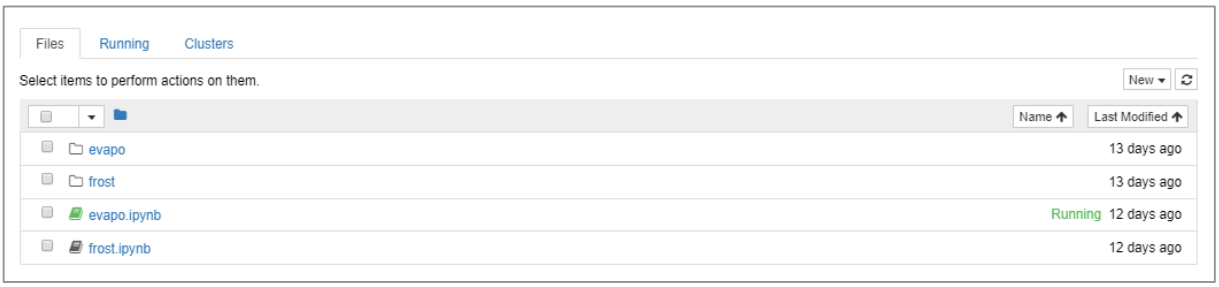
I. 데이터 로딩

1. 분석 환경 설정 및 패키지로딩
2. 데이터 불러오기
3. 데이터 결합하기
4. 데이터 타입 변환하기

1. 분석 환경 설정 및 패키지로딩 (1/2)

- **실습 실행 예제 파일 경로**

- 분석을 실행하기 위한 예제 파일 경로



- frost.ipynb 파일을 클릭

- **패키지 로딩**

- 분석을 위해 사용할 패키지를 로딩

```
import os
import sys
import glob
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
import h2o
import numpy as np

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.ensemble import GradientBoostingClassifier
from h2o.estimators.glm import H2OGeneralizedLinearEstimator
from h2o.estimators.gbm import H2OGradientBoostingEstimator
from h2o.estimators.random_forest import H2ORandomForestEstimator
from h2o.grid.grid_search import H2OGridSearch
```

- import로 사용할 패키지 및 모듈 로딩

1. 분석 환경 설정 및 패키지로딩 (2/2)

- **분석 환경 설정**

- 분석을 실행하기 전 메모리를 초기화 하고 현재 설정된 디렉토리를 확인

```
#============================================================
# 분석 환경 셋팅
#============================================================
#Python 메모리에 생성된 모든 객체  삭제(초기화)
sys.stdout.flush()

#============================================================
# 작업 디렉토리 경로 확인
#============================================================
currentPath=os.getcwd() #현재 위치한 디렉토리 경로확인
print('Current working dir : %s' % currentPath)
```

- sys.stdout.flush()로 Python 메모리 초기화
- os.getcwd() 로 현재 설정된 디렉토리 위치를 확인
  print 문을 사용하여 현재 설정 위치를 확인

2. 데이터불러오기

- **데이터를 불러온 뒤, 구조확인하기**
  - **분석에 활용할 종관기상관측장비(ASOS) 관측 자료 데이터와 각 ASOS 관측 지점별 이력 정보 데이터를 data.table 형태로 불러옴**

```
#============================================================
# 기상 데이터 읽어오기
#============================================================
#loading  weather data
ASOS_DATA = pd.read_csv(currentPath + '/frost/ASOS_DATA.csv', encoding='euc-kr')
#loading geographical data
GEO_DATA = pd.read_csv(currentPath + '/frost/GEO_DATA.csv', encoding='euc-kr')

#============================================================
# 불러온 데이터 구조 확인하기
#============================================================
ASOS_DATA.info()
GEO_DATA.info()
```

- **pd.read_csv()로 기상 및 지리정보 데이터 불러옴**
- **info()로 데이터 구조 확인**

> 실행 결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 143136 entries, 0 to 143135
Data columns (total 61 columns):
TM          143136 non-null object
STNID       143136 non-null int64
MIN_TA      143136 non-null float64
MAX_TA      143136 non-null float64
AVG_TA      143136 non-null float64
STD_TA      143136 non-null float64
AVG_RN      143136 non-null float64
SUM_RN      143136 non-null float64
MIN_SS      143136 non-null int64
MAX_SS      143136 non-null float64
AVG_SS      143136 non-null float64
STD_SS      143136 non-null float64
MIN_SI      143136 non-null int64
MAX_SI      143136 non-null float64
AVG_SI      143136 non-null float64
STD_SI      143136 non-null float64
MIN_WS      143136 non-null float64
MAX_WS      143136 non-null float64
AVG_WS      143136 non-null float64
STD_WS      143136 non-null float64
MIN_WD      143136 non-null float64
MAX_WD      143136 non-null float64
AVG_WD      143136 non-null float64
STD_WD      143136 non-null float64
MIN_TD      143136 non-null float64
MAX_TD      143136 non-null float64
AVG_TD      143136 non-null float64
STD_TD      143136 non-null float64
MIN_PA      143136 non-null float64
MAX_PA      143136 non-null float64
AVG_PA      143136 non-null float64
```

<실행 결과 일부 생략>

3. 데이터결합하기

- **데이터 결합하기**

- 관측 지점 번호(STNID)를 중심으로 기상 데이터(ASOS_DATA) 테이블과 지점 이력 정보 (GEO_DATA) 테이블을 결합

```
#=============================================================
# 테이블 결합 및 확인
#=============================================================
DATA = pd.merge(ASOS_DATA, GEO_DATA, how='left', on='STNID')
print(ASOS_DATA.columns)
print(GEO_DATA.columns)
print(DATA.columns)

len(DATA)
```

- **merge() 로 기상 데이터(ASOS_DATA) 와 지점이력정보(GEO_DATA)를 결합**
- **columns로 결합한 테이블의 Column 이름 확인**
- **len()로 결합한 테이블의 Row 수 확인**

> 실행 결과

```
Index(['TM', 'STNID', 'MIN_TA', 'MAX_TA', 'AVG_TA', 'STD_TA', 'AVG_RN',
       'SUM_RN', 'MIN_SS', 'MAX_SS', 'AVG_SS', 'STD_SS', 'MIN_SI', 'MAX_SI',
       'AVG_SI', 'STD_SI', 'MIN_WS', 'MAX_WS', 'AVG_WS', 'STD_WS', 'MIN_WD',
       'MAX_WD', 'AVG_WD', 'STD_WD', 'MIN_TD', 'MAX_TD', 'AVG_TD', 'STD_TD',
       'MIN_PA', 'MAX_PA', 'AVG_PA', 'STD_PA', 'MIN_PS', 'MAX_PS', 'AVG_PS',
       'STD_PS', 'MIN_PV', 'MAX_PV', 'AVG_PV', 'STD_PV', 'MIN_HM', 'MAX_HM',
       'AVG_HM', 'STD_HM', 'MIN_TCA', 'MAX_TCA', 'AVG_TCA', 'STD_TCA',
       'MIN_CA', 'MAX_CA', 'AVG_CA', 'STD_CA', 'MIN_CH', 'MAX_CH', 'AVG_CH',
       'STD_CH', 'MIN_TS', 'MAX_TS', 'AVG_TS', 'STD_TS', 'V'],
      dtype='object')
Index(['STNID', 'LAT', 'HT', 'MOUNTAIN', 'WATER'], dtype='object')
Index(['TM', 'STNID', 'MIN_TA', 'MAX_TA', 'AVG_TA', 'STD_TA', 'AVG_RN',
       'SUM_RN', 'MIN_SS', 'MAX_SS', 'AVG_SS', 'STD_SS', 'MIN_SI', 'MAX_SI',
       'AVG_SI', 'STD_SI', 'MIN_WS', 'MAX_WS', 'AVG_WS', 'STD_WS', 'MIN_WD',
       'MAX_WD', 'AVG_WD', 'STD_WD', 'MIN_TD', 'MAX_TD', 'AVG_TD', 'STD_TD',
       'MIN_PA', 'MAX_PA', 'AVG_PA', 'STD_PA', 'MIN_PS', 'MAX_PS', 'AVG_PS',
       'STD_PS', 'MIN_PV', 'MAX_PV', 'AVG_PV', 'STD_PV', 'MIN_HM', 'MAX_HM',
       'AVG_HM', 'STD_HM', 'MIN_TCA', 'MAX_TCA', 'AVG_TCA', 'STD_TCA',
       'MIN_CA', 'MAX_CA', 'AVG_CA', 'STD_CA', 'MIN_CH', 'MAX_CH', 'AVG_CH',
       'STD_CH', 'MIN_TS', 'MAX_TS', 'AVG_TS', 'STD_TS', 'V', 'LAT', 'HT',
       'MOUNTAIN', 'WATER'],
      dtype='object')

143136
```

4. 데이터 타입 변환하기

- **데이터 타입 변환하기**

  - 연속형(Numeric) 변수가 아닌 변수들을 범주형 변수로 변환

  (MOUNTAIN = 산맥 형성의 정도, Y = 서리 발생 여부)

```
#============================================================
# 데이터 타입 확인
#============================================================
print('** (변경전) MOUNTAIN, Y 컬럼 데이터 타입 확인')
print('MOUNTAIN : ', DATA['MOUNTAIN'].dtype)
print('Y : ', DATA['Y'].dtype)

#============================================================
# 데이터 타입 변환
#============================================================
DATA['MOUNTAIN'] = DATA['MOUNTAIN'].astype('category')
DATA['Y'] = DATA['Y'].astype('category')

#============================================================
# 변환된 타입 확인
#============================================================
print('** (변경후) MOUNTAIN, Y 컬럼 데이터 타입 확인')
print('MOUNTAIN : ', DATA['MOUNTAIN'].dtype)
print('Y : ', DATA['Y'].dtype)
```

- dtype으로 변수 형태를 확인
- astype('category')으로 범주형 변수로 변환
- dtype으로 변환된 변수가 범주형 변수인지를 확인

> 실행 결과

```
** (변경전) MOUNTAIN, Y 컬럼 데이터 타입 확인
MOUNTAIN :  int64
Y :  int64
** (변경후) MOUNTAIN, Y 컬럼 데이터 타입 확인
MOUNTAIN :  category
Y :  category
```

기상청

Ⅱ. 데이터 탐색

1. 요약 통계 보기
2. 상자 그림 그리기
3. 히스토그램 그리기

1. 요약 통계보기(1/2)

- **요약 통계 한 번에 보기**

- describe 함수를 사용하여 요약 통계량 한 번에 보기

```
#=============================================================
# 요약 통계 한번에 보기
#=============================================================
DATA.describe(include='all')
```

- **describe()로 기술 통계량을 살펴봄**
  - count, mean, std, min, 25%, 50%, 75%, max 등 제공

> 실행 결과

	TM	STNID	MIN_TA	MAX_TA	AVG_TA	STD_TA	AVG_RN	SUM_RN	MIN_SS	MAX_SS	...
count	143136	143136.000000	143136.000000	143136.000000	143136.000000	143136.000000	143136.000000	143136.000000	143136.0	143136.000000	...
unique	5112	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
top	2012-11-16 00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
freq	28	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
mean	NaN	142.785714	10.941795	13.602120	12.126269	0.900754	0.157505	1.102532	0.0	0.202136	...
std	NaN	27.786729	9.916761	9.923183	9.834469	0.685351	0.805227	5.636589	0.0	0.341927	...
min	NaN	95.000000	-26.500000	-22.600000	-24.314290	0.000000	0.000000	0.000000	0.0	0.000000	...
25%	NaN	118.000000	3.100000	5.700000	4.285714	0.395897	0.000000	0.000000	0.0	0.000000	...
50%	NaN	141.500000	11.800000	14.600000	13.128570	0.711996	0.000000	0.000000	0.0	0.000000	...
75%	NaN	162.750000	19.400000	21.900000	20.514290	1.211678	0.000000	0.000000	0.0	0.300000	...
max	NaN	192.000000	32.600000	36.600000	34.114290	9.234711	35.857140	251.000000	0.0	1.000000	...

11 rows × 65 columns

1. 요약 통계보기(2/2)

- **요약 통계 그룹별로 보기**

- groupby 함수를 사용하여 지점별로 묶어 요약 통계 보기

```
#============================================================
# 지점별 요약통계 보기
#============================================================
grouped = DATA.groupby('STNID')
grouped.describe(include='all')
```

- **groupby() 로 지점별 데이터를 묶음**
- **묶은 데이터를 describe()로 지점별로 요약통계 보기**

> 실행 결과

	AVG_CA										...	Y								
	count	unique	top	freq	mean	std	min	25%	50%	75%	...	unique	top	freq	mean	std	min	25%	50%	75%	max
STNID																					
95	5112.0	NaN	NaN	NaN	3.661321	2.589887	0.0	1.312798	3.307679	5.884499	...	2.0	0.0	4244.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
100	5112.0	NaN	NaN	NaN	4.898497	2.582260	0.0	2.571674	4.870705	7.198383	...	2.0	0.0	4669.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
101	5112.0	NaN	NaN	NaN	4.662811	2.262039	0.0	2.755976	4.394010	6.691730	...	2.0	0.0	4346.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
108	5112.0	NaN	NaN	NaN	4.118393	2.232109	0.0	2.181267	3.643676	6.091827	...	2.0	0.0	4837.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
112	5112.0	NaN	NaN	NaN	4.137529	2.277682	0.0	2.191817	3.698012	6.012954	...	2.0	0.0	4916.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
114	5112.0	NaN	NaN	NaN	4.436642	2.120303	0.0	2.671929	4.226737	6.195174	...	2.0	0.0	4590.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
115	5112.0	NaN	NaN	NaN	4.810795	2.955282	0.0	2.333333	5.200000	7.400000	...	2.0	0.0	5109.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
119	5112.0	NaN	NaN	NaN	4.253942	2.193981	0.0	2.382028	3.845543	6.175037	...	2.0	0.0	4615.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
130	5112.0	NaN	NaN	NaN	3.013033	2.820232	0.0	0.000000	2.753634	5.500000	...	2.0	0.0	5047.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
131	5112.0	NaN	NaN	NaN	4.148088	2.061594	0.0	2.421723	3.881645	5.989999	...	2.0	0.0	4782.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
133	5112.0	NaN	NaN	NaN	4.192216	2.018648	0.0	2.532485	3.920532	5.998214	...	2.0	0.0	4364.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
136	5112.0	NaN	NaN	NaN	4.041478	2.128472	0.0	2.238194	3.737795	5.794672	...	2.0	0.0	4612.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
138	5112.0	NaN	NaN	NaN	3.892291	2.167875	0.0	2.013217	3.466095	5.848194	...	2.0	0.0	5068.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
140	5112.0	NaN	NaN	NaN	3.368716	2.824480	0.0	0.000000	3.282359	6.000000	...	2.0	0.0	4799.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
143	5112.0	NaN	NaN	NaN	3.993250	1.974631	0.0	2.432053	3.744518	5.511656	...	2.0	0.0	4896.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
146	5112.0	NaN	NaN	NaN	4.396388	2.035635	0.0	2.731766	4.285423	6.211644	...	2.0	0.0	4593.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
152	5112.0	NaN	NaN	NaN	2.849829	2.912718	0.0	0.000000	2.000000	5.800000	...	2.0	0.0	4954.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
155	5112.0	NaN	NaN	NaN	3.813403	2.103665	0.0	2.000018	3.350585	5.568674	...	2.0	0.0	5046.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
156	5112.0	NaN	NaN	NaN	4.364794	1.982983	0.0	2.737210	4.294638	6.047630	...	2.0	0.0	4627.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
159	5112.0	NaN	NaN	NaN	3.917457	2.160816	0.0	2.052930	3.426632	5.801651	...	2.0	0.0	5091.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
162	5112.0	NaN	NaN	NaN	2.851369	2.849704	0.0	0.000000	2.666667	5.333333	...	2.0	0.0	4791.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
165	5112.0	NaN	NaN	NaN	4.414319	1.999278	0.0	2.805214	4.347836	6.000000	...	2.0	0.0	4898.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
168	5112.0	NaN	NaN	NaN	2.972049	2.809169	0.0	0.000000	2.666667	5.666667	...	2.0	0.0	5081.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
170	5112.0	NaN	NaN	NaN	3.356987	2.658972	0.0	0.200000	3.209877	5.800000	...	2.0	0.0	4977.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
184	5112.0	NaN	NaN	NaN	4.547879	2.076686	0.0	2.845978	4.554490	6.200060	...	2.0	0.0	5098.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
185	5112.0	NaN	NaN	NaN	4.713266	2.065157	0.0	3.093832	4.670075	6.285714	...	2.0	0.0	5111.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
189	5112.0	NaN	NaN	NaN	3.910223	2.434922	0.0	2.000000	3.764587	6.000000	...	2.0	0.0	5098.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192	5112.0	NaN	NaN	NaN	3.837746	2.003439	0.0	2.243054	3.522764	5.416441	...	2.0	0.0	4348.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN

28 rows × 704 columns

2. 상자 그림그리기 (1/2)

- **지점별 상자 그림 그리기**

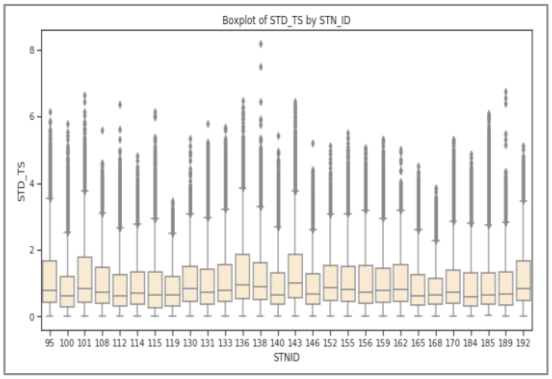
- 분석 데이터를 지점별로 상자 그림을 그려 분포를 확인

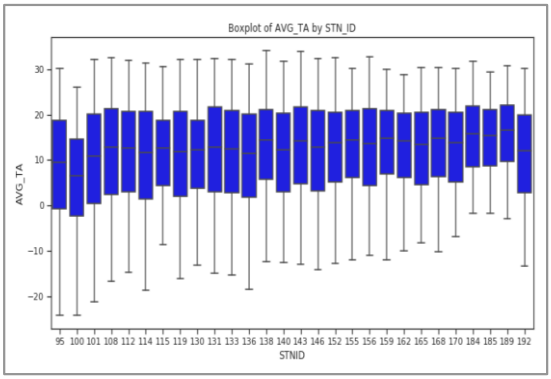
```
#=============================================================
# STN별 박스플롯 그리기
#=============================================================
sns.set()
sns.set_style("ticks")
%matplotlib inline

col_bisque = ["#ffebcd"]
col_blue = ["#0000ff"]
col_gray = ["#a9a9a9"]
col_red = ["#ff0000"]

plt.figure(figsize=(12,6))
plt.title("Boxplot of STD_TS by STN_ID")
sns.boxplot(x="STNID", y="STD_TS", data=DATA, palette = sns.color_palette(col_bisque))
plt.show()

plt.figure(figsize=(12,6))
plt.title("Boxplot of AVG_TA by STN_ID")
sns.boxplot(x="STNID", y="AVG_TA", data=DATA, palette = sns.color_palette(col_blue))
plt.show()

plt.figure(figsize=(12,6))
plt.title("Boxplot of MAX_WS by STN_ID")
sns.boxplot(x="STNID", y="MAX_WS", data=DATA, palette = sns.color_palette(col_gray))
plt.show()

plt.figure(figsize=(12,6))
plt.title("Boxplot of MIN_TCA by STN_ID")
sns.boxplot(x="STNID", y="MIN_TCA", data=DATA, palette = sns.color_palette(col_red))
plt.show()
```

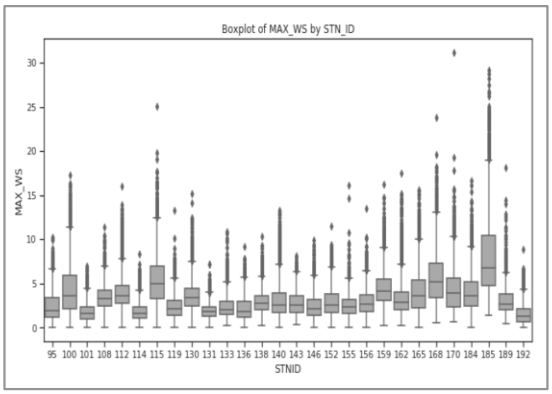
- **plt.title() 로 상자 그림의 제목을 지정**
- **sns.boxplot() 으로 STNID별 각 컬럼의 상자 그림을 그리고, x축, y축 라벨을 붙임**
- **plt.show() 로 상자 그림을 표출**

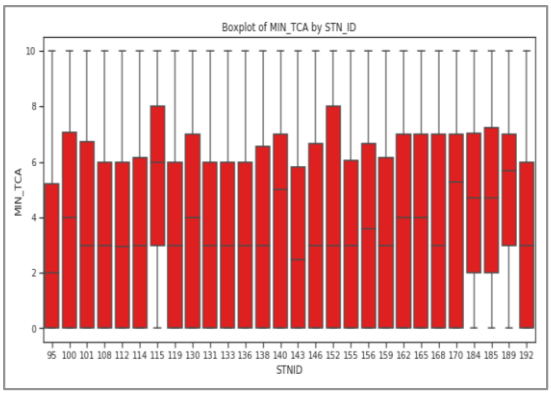
2. 상자 그림그리기 (2/2)

> boxplot 결과









3. 히스토그램그리기 (1/2)

- **히스토그램 그리기**

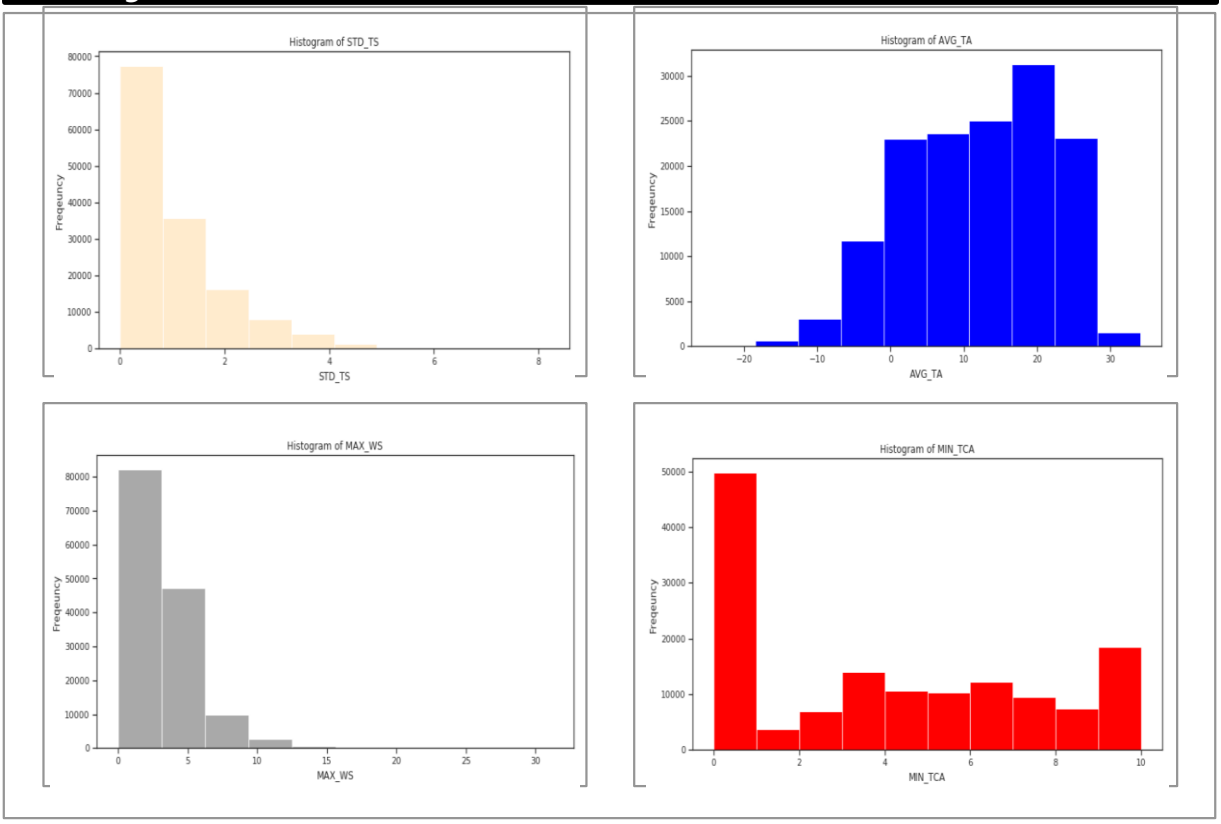
- 분석 데이터를 히스토그램을 그려 분포를 확인

```
#=============================================================
# Histogram 그리기
#=============================================================
plt.figure(figsize=(12,6))
plt.title("Histogram of STD_TS")
plt.xlabel('STD_TS')
plt.ylabel('Freqeuncy')
plt.hist(DATA['STD_TS'].dropna(), color = col_bisque)
plt.show()

plt.figure(figsize=(12,6))
plt.title("Histogram of AVG_TA")
plt.xlabel('AVG_TA')
plt.ylabel('Freqeuncy')
plt.hist(DATA['AVG_TA'].dropna(), color = col_blue)
plt.show()

plt.figure(figsize=(12,6))
plt.title("Histogram of MAX_WS")
plt.xlabel('MAX_WS')
plt.ylabel('Freqeuncy')
plt.hist(DATA['MAX_WS'].dropna(), color = col_gray)
plt.show()

plt.figure(figsize=(12,6))
plt.title("Histogram of MIN_TCA")
plt.xlabel('MIN_TCA')
plt.ylabel('Freqeuncy')
plt.hist(DATA['MIN_TCA'].dropna(), color = col_red)
plt.show()
```

- **plt.title() 로 히스토그램의 제목을 지정**
- **plt.hist() 로 각 컬럼의 히스토그램을 그리고 x축, y축 라벨을 붙임**
- **plt.show() 로 히스토그램을 표출**

3. 히스토그램그리기 (2/2)

> histogram 결과



기상청

Ⅲ. 데이터 처리

1. 이상치 처리
2. 결측치 처리
3. 파생변수 생성

1. 이상치처리

- **이상치 처리**

- ASOS 물리 한계 검사표를 기준으로 기상 데이터(ASOS_DATA) 테이블 내 기상 변수의 이상치 확인 및 처리

```
#============================================================
# 이상치 확인
#============================================================
DATA.describe(include='all')

#============================================================
# 예시 : 기온 관련 변수 이상치 처리 방법 (기준값: -80~60℃)
#============================================================
DATA.at[(DATA['MIN_TA'] < -80) | (DATA['MIN_TA'] > 60), 'MIN_TA'] = np.nan
DATA.at[(DATA['MAX_TA'] < -80) | (DATA['MAX_TA'] > 60), 'MAX_TA'] = np.nan
DATA.at[(DATA['AVG_TA'] < -80) | (DATA['AVG_TA'] > 60), 'AVG_TA'] = np.nan
DATA.at[(DATA['STD_TA'] < -80) | (DATA['STD_TA'] > 60), 'STD_TA'] = np.nan
DATA.describe(include = 'all')
```

- **describe()로 이상치 확인**
- **기온(TA) 관련 변수가 -80 이하이거나 60 이상인 값은 NA로 치환**

> 실행 결과

	TM	STNID	MIN_TA	MAX_TA	AVG_TA	STD_TA	AVG_RN	SUM_RN	MIN_SS	MAX_SS	...
count	143136	143136.000000	143136.000000	143136.000000	143136.000000	143136.000000	143136.000000	143136.000000	143136.0	143136.000000	...
unique	5112	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
top	2012-11-16 00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
freq	28	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
mean	NaN	142.785714	10.941795	13.602120	12.126269	0.900754	0.157505	1.102532	0.0	0.202136	...
std	NaN	27.786729	9.916761	9.923183	9.834469	0.685351	0.805227	5.636589	0.0	0.341927	...
min	NaN	95.000000	-26.500000	-22.600000	-24.314290	0.000000	0.000000	0.000000	0.0	0.000000	...
25%	NaN	118.000000	3.100000	5.700000	4.285714	0.395897	0.000000	0.000000	0.0	0.000000	...
50%	NaN	141.500000	11.800000	14.600000	13.128570	0.711996	0.000000	0.000000	0.0	0.000000	...
75%	NaN	162.750000	19.400000	21.900000	20.514290	1.211678	0.000000	0.000000	0.0	0.300000	...
max	NaN	192.000000	32.600000	36.600000	34.114290	9.234711	35.857140	251.000000	0.0	1.000000	...

11 rows × 65 columns

2. 결측치처리

- **결측치 0으로 대체**
  - 강수량(RN) 관련 변수 : 강수가 발생하지 않은 상태로 판단하여 결측치를 0으로 대체
  - 일조(SS), 일사(SI) 관련 변수 : 일조, 일사 시간이 없다고 판단하여 결측치를 0으로 대체

```
#=============================================================
# 결측치 확인
#=============================================================
DATA.isnull().sum()

#=============================================================
# 예시 : 강수량(RN), 일조(SS), 일사(SI) 관련 변수를 0으로 치환
#=============================================================
DATA['AVG_RN'] = DATA['AVG_RN'].fillna(0)
DATA['SUM_RN'] = DATA['SUM_RN'].fillna(0)
DATA['MIN_SS'] = DATA['MIN_SS'].fillna(0)
DATA['MAX_SS'] = DATA['MAX_SS'].fillna(0)
DATA['AVG_SS'] = DATA['AVG_SS'].fillna(0)
DATA['STD_SS'] = DATA['STD_SS'].fillna(0)
DATA['MIN_SI'] = DATA['MIN_SI'].fillna(0)
DATA['MAX_SI'] = DATA['MAX_SI'].fillna(0)
DATA['AVG_SI'] = DATA['AVG_SI'].fillna(0)
DATA['STD_SI'] = DATA['STD_SI'].fillna(0)

# 데이터 결측치 확인
DATA.isnull().sum()
DATA.isnull().sum().sum()
```

- isnull().sum()으로 변수들의 결측치 확인
- fillna()로 해당 기상변수가 NA인 값 선택하여 0으로 치환
- isnull().sum().sum() 으로 최종적으로 모든 NA가 0으로 치환되었는지 확인

> 실행 결과

```
TM          0
STNID       0
MIN_TA      0
MAX_TA      0
AVG_TA      0
STD_TA      0
AVG_RN      0
SUM_RN      0
MIN_SS      0
MAX_SS      0
```

<isnull().sum() 실행결과일부생략>

```
0
```

3. 파생변수생성 (1/2)

- **파생변수 생성**

- 최고 기온과 최저 기온의 차로 MMT_TA라는 파생변수를 생성

- 현상 발생 시간에서 문자열을 추출해 파생변수를 생성

```
#=============================================================
# 기온 (최고-최저) 파생변수 생성
#=============================================================
DATA['MMT_TA'] = DATA['MAX_TA'] - DATA['MIN_TA']

#=============================================================
# 현상 발생 월, 시간 파생변수 생성 및 타입 변환
#=============================================================
DATA['MONTH'] = DATA['TM'].str.slice(5,7)
DATA['HOUR'] = DATA['TM'].str.slice(11,13)

DATA['MONTH'] = DATA['MONTH'].astype('category')
DATA['HOUR'] = DATA['HOUR'].astype('category')

DATA.info()
```

- **관측 시간 (최고 - 최저) 기온(MAX_TA - MIN_TA)으로 MMT_TA 파생변수생성**
- **slice()로 TM 변수 내 문자열을 추출해 파생변수 생성**
- **astype('category') 로 MONTH 및 HOUR 변수 범주형 변수로 변환**
- **info()로 데이터 구조를 확인하여 파생변수 생성 및 타입 확인**

3. 파생변수생성 (2/2)

> 실행 결과

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 143136 entries, 0 to 143135
Data columns (total 68 columns):
TM          143136 non-null object
STNID       143136 non-null int64
MIN_TA      143136 non-null float64
MAX_TA      143136 non-null float64
AVG_TA      143136 non-null float64
STD_TA      143136 non-null float64
AVG_RN      143136 non-null float64
SUM_RN      143136 non-null float64
MIN_SS      143136 non-null int64
MAX_SS      143136 non-null float64
AVG_SS      143136 non-null float64
STD_SS      143136 non-null float64
MIN_SI      143136 non-null int64
MAX_SI      143136 non-null float64
AVG_SI      143136 non-null float64
STD_SI      143136 non-null float64
MIN_WS      143136 non-null float64
MAX_WS      143136 non-null float64
AVG_WS      143136 non-null float64
STD_WS      143136 non-null float64
MIN_WD      143136 non-null float64
MAX_WD      143136 non-null float64
AVG_WD      143136 non-null float64
STD_WD      143136 non-null float64
MIN_TD      143136 non-null float64
MAX_TD      143136 non-null float64
AVG_TD      143136 non-null float64
STD_TD      143136 non-null float64
MIN_PA      143136 non-null float64
MAX_PA      143136 non-null float64
AVG_PA      143136 non-null float64
STD_PA      143136 non-null float64
MIN_PS      143136 non-null float64
MAX_PS      143136 non-null float64
AVG_PS      143136 non-null float64
STD_PS      143136 non-null float64
MIN_PV      143136 non-null float64
MAX_PV      143136 non-null float64
AVG_PV      143136 non-null float64
STD_PV      143136 non-null float64
MIN_HM      143136 non-null float64
MAX_HM      143136 non-null float64
AVG_HM      143136 non-null float64
STD_HM      143136 non-null float64
MIN_TCA     143136 non-null float64
MAX_TCA     143136 non-null float64
AVG_TCA     143136 non-null float64
STD_TCA     143136 non-null float64
MIN_CA      143136 non-null float64
MAX_CA      143136 non-null float64
AVG_CA      143136 non-null float64
STD_CA      143136 non-null float64
MIN_CH      143136 non-null float64
MAX_CH      143136 non-null float64
AVG_CH      143136 non-null float64
STD_CH      143136 non-null float64
MIN_TS      143136 non-null float64
MAX_TS      143136 non-null float64
AVG_TS      143136 non-null float64
STD_TS      143136 non-null float64
Y           143136 non-null category
LAT         143136 non-null float64
HT          143136 non-null float64
MOUNTAIN    143136 non-null category
WATER       143136 non-null int64
MMT_TA      143136 non-null float64
MONTH       143136 non-null category
HOUR        143136 non-null category
dtypes: category(4), float64(59), int64(4), object(1)
memory usage: 76.5+ MB
```

기상청

Ⅳ. 모형 구축

1. 변수 선택
2. 모형 구축

1. 변수선택(1/5)

- **H2O 서버 세팅**

- H2O는 빅데이터 처리를 위한 분산처리 프로세스를 통해 빠른 처리속도를 지원하고 다양한 머신러닝 분석 기술을 제공하는 패키지

- H2O 그래디언트 부스팅 모델(Gradient Boosting Model)을 활용하여분석

```
#============================================================
# 로컬에 H2O 가상서버 설정하기
#============================================================
h2o.init(max_mem_size = "4G", nthreads = 4) # h2o 서버 세팅
```

- **h2o.init()으로 H2O 자바가상머신을 세팅**

> 실행 결과

```
Checking whether there is an H2O instance running at http://localhost:54321 . connected.
```

H2O cluster uptime:	4 mins 50 secs
H2O cluster timezone:	Asia/Seoul
H2O data parsing timezone:	UTC
H2O cluster version:	3.28.0.1
H2O cluster version age:	2 days
H2O cluster name:	H2O_from_python_root_65lnbm
H2O cluster total nodes:	1
H2O cluster free memory:	3.372 Gb
H2O cluster total cores:	4
H2O cluster allowed cores:	4
H2O cluster status:	locked, healthy
H2O connection url:	http://localhost:54321
H2O connection proxy:	{'http': None, 'https': None}
H2O internal security:	False
H2O API Extensions:	Amazon S3, XGBoost, Algos, AutoML, Core V3, TargetEncoder, Core V4
Python version:	3.6.9 final

1. 변수선택(2/5)

- **데이터 셋 분할**

- 모델을 학습하는데 사용할 데이터 셋(train)과 모델을 검증하는데 사용할 데이터 셋(test)으로 나눔

```
#============================================================
# 데이터 분할
#============================================================
DATA = DATA.drop(['TM', 'MIN_SI', 'MIN_SS'], axis=1)
data_hex = h2o.H2OFrame(DATA)

data_hex['Y']=data_hex['Y'].asfactor()
data_hex['MOUNTAIN']=data_hex['MOUNTAIN'].asfactor()
data_hex['MONTH']=data_hex['MONTH'].asfactor()
data_hex['HOUR']=data_hex['HOUR'].asfactor()

train_data, test_data = data_hex.split_frame([0.8], seed=1234)

#============================================================
# 분할된 데이터 셋 확인
#============================================================
print(len(data_hex))
print(len(train_data))
print(len(test_data))
```

- **h2o.H2OFrame()으로 분석 데이터셋을 H2OFrame구조의 데이터로 변환**
- **asfactor() 로 범주형 변수 처리**
- **split_frame()으로 데이터 셋을 분할**
- **len()으로 전체 데이터 셋과 분할된 데이터 셋의 행 수 확인**

> 실행 결과

```
Parse progress: |████████████████████████████████████████████████████████████████████████████████| 100%
143136
114591
28545
```

1. 변수선택(3/5)

- **변수들 간 다중공선성 제거(1/2)**

- 그래디언트 부스팅 모형의 변수 중요도를 기준으로 상관계수가 높은 변수들을 제거해나가는 방법을 활용하여 변수들 간 다중공선성 문제를 해결

```
#============================================================
# 변수들간 다중공선성 제거
#============================================================
# 상관계수를 구하기 전 범주형 변수 제외
X = DATA.drop(['STNID', 'Y', 'MOUNTAIN', 'MONTH', 'HOUR'], axis=1)

# 변수들 간의 상관계수 산출
corr = X.corr()
Abs_corr = pd.DataFrame(abs(corr))
Abs_corr['variable'] = Abs_corr.index
Abs_corr.reset_index(drop=Ture, inplace=Ture)
Abs_corr.info()

#============================================================
# variable importance 산출하기
#============================================================
new_data_hex = train_data.drop(['STNID','Y'], axis=1)

xList = new_data_hex.columns
y = "Y“

gbm = H2OGradientBoostingEstimator(ntrees=50, max_depth=5, seed=1234)
gbm.train(x = xList, y = y, training_frame = data_hex)

varimp = gbm.varimp(True)
varimp = varimp[varimp.variable != 'MOUNTAIN']
varimp = varimp[varimp.variable != 'MONTH']
varimp = varimp[varimp.variable != 'HOUR']
varimp
```

- **Corr()로 STNID, Y(서리 발생 여부), MOUNTAIN(산맥 형성 정도), MONTH(현상 발생 월), HOUR(현상 발생 시간)와 같은 범주형 변수를 제외한 상관계수를 산출**
- **abs()로 상관계수에 절댓값을 적용**
- **index로 행 이름으로 설정되어있는 변수명을 variable이라는 이름의 칼럼으로 추출**
- **X값 변수목록 및 Y값 변수명을 세팅 (xList, y)**
- **H2OGradientBoostingEstimator() 및 train()으로 H2O 그래디언트 부스팅 모델을 활용하여 모형 구축**
  - 함수의 파라미터는 H2OGradientBoostingEstimator 함수의 default 값으로 설정
- **varimp()로 구축한 모형에 활용된 변수들의 변수 중요도를 산출**
- **변수중요도에서 상관계수 산출에서 제외한 범주형 변수를 제외**

1. 변수선택(4/5)

- **변수들 간 다중공선성 제거(2/2)**

- 변수중요도가 높은 변수부터 순서대로 하나씩 선택하여, 해당 변수와 상관계수의 절댓값이 0.45가 넘는 변수를 제거

```
#============================================================
# 다중공선성 제거
#============================================================
finalvar = varimp
j=0

while j < len(finalvar.variable):
    tmp = Abs_corr[(Abs_corr.variable != 'y') & (Abs_corr.variable != finalvar.variable[j])].loc[:,['var
iable',finalvar.variable[j]]]
    tmp.columns = ['variable', 'Pearson']
    tmp.sort_values(['Pearson'], axis=0, ascending=False, inplace=True)
    tmp = tmp[tmp.Pearson > 0.45]

    finalvar = pd.merge(finalvar, tmp, how='left', on='variable')
    finalvar = finalvar.loc[finalvar.isnull()['Pearson'], :]
    finalvar = finalvar.reset_index(drop=True)
    finalvar = finalvar.drop('Pearson', 1)
    finalvar.sort_values(['scaled_importance'], axis=0, ascending=False, inplace=True)
    j = j + 1

#최종선택변수
finalvar
```

- **최종선택변수 데이터 프레임을 선언**
- **while() 문으로 최종선택변수 데이터 프레임의 길이보다 작을 동안, 변수 중요도가 높은 변수를 하나씩 선택하여 작업을 수행**
- **선택한 변수와 다른 변수들 간의 상관계수의 절댓값을 추출**
- **columns로 칼럼 이름 지정**
- **sort_values()로 상관계수의 절댓값이 높은 순서로 정렬**
- **상관계수의 절댓값이 0.45가 넘는 변수들을 추출**
- **merge로 변수 중요도 순서로 정렬된 데이터(varimp)에 결합**
- **결합한 Pearson 칼럼이 NA인 것만 남기고 나머지 변수 삭제한 후 변수 중요도가 높은 순서 대로 정렬**

1. 변수선택(5/5)

> 실행 결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 61 columns):
MIN_TA      60 non-null float64
MAX_TA      60 non-null float64
AVG_TA      60 non-null float64
STD_TA      60 non-null float64
AVG_RN      60 non-null float64
SUM_RN      60 non-null float64
MAX_SS      60 non-null float64
AVG_SS      60 non-null float64
STD_SS      60 non-null float64
MAX_SI      60 non-null float64
AVG_SI      60 non-null float64
STD_SI      60 non-null float64
MIN_WS      60 non-null float64
MAX_WS      60 non-null float64
AVG_WS      60 non-null float64
STD_WS      60 non-null float64
MIN_WD      60 non-null float64
MAX_WD      60 non-null float64
AVG_WD      60 non-null float64
STD_WD      60 non-null float64
MIN_TD      60 non-null float64
```

<Abs_corr 실행결과 일부 생략>

● ● ●

finalvar

	variable	relative_importance	scaled_importance	percentage
0	MIN_TA	7373.253418	1.000000	0.337619
1	AVG_WS	3931.138672	0.533162	0.180006
2	AVG_TCA	1185.468750	0.160780	0.054282
3	MAX_HM	1140.171021	0.154636	0.052208
4	AVG_RN	363.638306	0.049319	0.016651
5	WATER	208.827240	0.028322	0.009562
6	AVG_WD	169.308487	0.022963	0.007753
7	MMT_TA	78.008545	0.010580	0.003572
8	STD_TCA	71.025902	0.009633	0.003252
9	STD_TD	68.185516	0.009248	0.003122
10	MIN_CH	60.136482	0.008156	0.002754
11	STD_WD	24.296446	0.003295	0.001113
12	STD_CH	9.878351	0.001340	0.000452
13	STD_PA	9.288697	0.001260	0.000425
14	MAX_SI	3.786789	0.000514	0.000173

2. 모형구축(1/2)

- **하이퍼 파라미터 최적화(Hyper-parameter Optimization)**
  - 그래디언트 부스팅 모델의 하이퍼 파라미터를 조정하여 AUC(Area Under the Curve)가 가장 높은 모형을 선택
  - sample_rate, max_depth, ntrees 파라미터를 조정

```
#============================================================
# 하이퍼파라미터 최적화
#============================================================
varList = list(finalvar['variable'])
extList = ['MOUNTAIN', 'MONTH', 'HOUR']
varList = varList + extList
y = "Y"

# 하이퍼파라미터 조합만들기
hyper_params = {'sample_rate': [0.3, 0.4],
                'max_depth': [18, 20, 25],
                'ntrees': [25, 50]}

# 조합 모형 돌리기
m = H2OGridSearch(H2OGradientBoostingEstimator, grid_id = 'gbm_grid', hyper_params = hyp
er_params)
m.train(x = varList, y = y, training_frame = train_data)

# AUC가 높은 순으로 정렬하기
sorted_grid = m.get_grid(sort_by = 'auc', decreasing = True)
print('===== sorted_grid =====')
print(sorted_grid)

# 베스트 모형 선택
best_model = h2o.get_model(sorted_grid.models[0])
print('===== best_model =====')
print(best_model)
```

  - 최종선택변수를 varList로, 서리발생여부를 y값으로 변수 지정
  - 상관계수를 구하기 전 제거했던 범주형 변수를 추가
  - 하이퍼 파라미터 조합 리스트를 생성
  - H2OGridSearch()로 하이퍼 파라미터 조합들을 활용하여 모형을 구축한 "gbm_grid"를 생성
  - get_grid()로 "m"의 모형 구축 결과를 AUC(Area Under theCurve)가 높은 순으로 정렬
  - get_model()로 정렬된 모형 결과 중, 가장 상위에 있는 모형을 베스트 모형으로 선택

2. 모형구축(2/2)

> 실행 결과

```
gbm Grid Build progress: |████████████████████████████████████████████████████████████████| 100%
===== sorted_grid =====
     max_depth ntrees sample_rate          model_ids                 auc
0           25     50         0.4  gbm_grid_model_11  0.9996280264984069
1           20     50         0.4  gbm_grid_model_10  0.9996153391044977
2           18     50         0.4   gbm_grid_model_9  0.9995175946273815
3           25     50         0.3   gbm_grid_model_5  0.9986589915758569
4           20     50         0.3   gbm_grid_model_4  0.9984742451240443
5           18     50         0.3   gbm_grid_model_3  0.9984283763545944
6           18     25         0.4   gbm_grid_model_6  0.9970902376418257
7           20     25         0.4   gbm_grid_model_7  0.9970257049397353
8           25     25         0.4   gbm_grid_model_8  0.9970116164673443
9           25     25         0.3   gbm_grid_model_2  0.9948754068329287
10          18     25         0.3   gbm_grid_model_0  0.9947779643240292
11          20     25         0.3   gbm_grid_model_1    0.99470320388413

===== best_model =====
Model Details
=============
H2OGradientBoostingEstimator :  Gradient Boosting Machine
Model Key:  gbm_grid_model_11
Model Summary:
```

number_of_trees	number_of_internal_trees	model_size_in_bytes	min_depth	max_depth	mean_depth	min_leaves	max_leaves	mean_leaves
50.0	50.0	1141397.0	25.0	25.0	25.0	574.0	2276.0	1807.74

```
ModelMetricsBinomial: gbm
** Reported on train data. **

MSE: 0.005203713620296316
RMSE: 0.07213677023748925
LogLoss: 0.02312968505852359
Mean Per-Class Error: 0.007208964924290684
AUC: 0.9996280264984069
Gini: 0.9992560529968137
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.44732706881503603:
```

	0	1	Error	Rate
0	107471.0	249.0	0.0023	(249.0/107720.0)
1	268.0	6603.0	0.039	(268.0/6871.0)
Total	107739.0	6852.0	0.0045	(517.0/114591.0)

● ● ●

Variable Importances:

variable	relative_importance	scaled_importance	percentage
MIN_TA	6795.3554688	1.0	0.3508437
AVG_WS	2282.8693848	0.3359455	0.1178644
MAX_HM	1524.7783203	0.2243854	0.0787242
AVG_TCA	1426.5539551	0.2099307	0.0736529
MONTH	1015.1885376	0.1493945	0.0524141
AVG_WD	746.5147705	0.1098566	0.0385425
MMT_TA	685.6093140	0.1008938	0.0353980
STD_TCA	650.3401489	0.0957036	0.0335770
STD_WD	603.2615967	0.0887756	0.0311464
STD_TD	593.2841797	0.0873073	0.0306312
MOUNTAIN	584.9393311	0.0860793	0.0302004
STD_CH	538.7757568	0.0792859	0.0278170
STD_PA	505.9125977	0.0744498	0.0261202
HOUR	498.6494141	0.0733809	0.0257452
MIN_CH	349.4960022	0.0514316	0.0180445
WATER	331.6039734	0.0487986	0.0171207
MAX_SI	125.8939514	0.0185265	0.0064999
AVG_RN	109.5849762	0.0161265	0.0056579

기상청

V. 모형 검증

1. 모형 성능 및 예측력 검증
2. 지점별 예측력 검증

1. 모형 성능 및 예측력검증

- **모형 성능 및 예측력 검증**

- 최종 선택한 모형을 AUC(Area Under the Curve)와 혼동행렬(Confusion Matrix)을 확인함으로써 모형의 성능을 검증하고, 검증 데이터 셋(test)을 활용하여 예측력을 검증

```
#============================================================
# 모형 성능 및 예측력 검증
#============================================================
# 최종 선택 모형 구축
gbm_fit = H2OGradientBoostingEstimator(ntrees=50, max_depth=25, sample_rate=0.4,
seed=1234)
gbm_fit.train(x = varList, y = y, training_frame = train_data)

# 모형 성능 검증
gbm_fit.auc()
gbm_fit.confusion_matrix()

# 모형 예측력 검증
per_gbm = gbm_fit.model_performance(test_data = test_data)
per_gbm.auc()
per_gbm.confusion_matrix()
```

- **H2OGradientBoostingEstimator()로 최종 선택 모형을 구축**
- **auc()로 AUC 값을계산**
- **confusion_matrix()로 혼동행렬을 출력**
- **model_performance()로 모형의 예측력 검증**

> 실행 결과

```
gbm Model Build progress: |████████████████████████████████████████████████| 100%
0.9996293147610225
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.4704132992596627:
```

	0	1	Error	Rate
0	107492.0	228.0	0.0021	(228.0/107720.0)
1	301.0	6570.0	0.0438	(301.0/6871.0)
Total	107793.0	6798.0	0.0046	(529.0/114591.0)

```
0.9812803421620298
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.3188170787448533:
```

	0	1	Error	Rate
0	26284.0	603.0	0.0224	(603.0/26887.0)
1	417.0	1241.0	0.2515	(417.0/1658.0)
Total	26701.0	1844.0	0.0357	(1020.0/28545.0)

2. 지점별 예측력검증(1/3)

- **지점별 예측력 검증**

- 한 지점을 제외한 나머지 지점의 데이터를 모형의 훈련 데이터로 사용하고, 훈련하지 않은 한 지점을 테스트 데이터로 활용하여 모형을 평가

```
#============================================================
# 지점별 예측력 검증
#============================================================
# STN별 리스트 생성
STNList = DATA['STNID'].unique()

# 결과 리스트 생성
auc = list()
confusion = list()

# 지점별 데이터 셋 분할, 모형 구축, 예측력 검증
for i in range(len(STNList)):
    stn = STNList[i]
    print("STNID : ", stn, "...computing")
    train = data_hex[data_hex['STNID'] != int(stn), :]
    valid = data_hex[data_hex['STNID'] == int(stn), :]

    m = H2OGradientBoostingEstimator(ntrees=50, max_depth=25, sample_rate = 0.4, seed=1)
    m.train(x = varList, y = y, training_frame = train)

    per_m = m.model_performance(test_data = valid)

    auc_df = pd.DataFrame({'STNID':[stn], '':[per_m.auc()]})
    auc.append(auc_df)

    confusion_df = pd.DataFrame({'STNID':[stn], '':[per_m.confusion_matrix()]})
    confusion.append(confusion_df)

# STN별 교차검증 결과
auc_stnid = pd.concat(auc)
confusion_stnid = pd.concat(confusion)
```

2. 지점별 예측력 검증(2/3)

- unique()로 STN 리스트 생성
- 지점별 결과값을 저장할 리스트 변수 생성
- for() 문으로 지점별로 예측력 검증 수행
- print()로 반복문 프로세스 진행 상황 출력
- 해당 지점을 제외한 나머지 지점 데이터를 train 데이터로 생성
- 해당 지점의 데이터를 valid 데이터 생성
- 모형 학습 수행
- auc()와 confusion_matrix()로 모형의 성능과 변수 중요도를 산출하여 각각의 리스트에 저장
- for 문이 수행 완료된 후, concat()으로 리스트를 하나의 데이터 프레임으로 합쳐서 출력

> 실행 결과

```
STNID :  95 ...computing
gbm Model Build progress: |████████████████████████████████████████████████████████████| 100%
STNID :  101 ...computing
gbm Model Build progress: |████████████████████████████████████████████████████████████| 100%
STNID :  108 ...computing
gbm Model Build progress: |████████████████████████████████████████████████████████████| 100%
STNID :  114 ...computing
gbm Model Build progress: |████████████████████████████████████████████████████████████| 100%
STNID :  130 ...computing
gbm Model Build progress: |████████████████████████████████████████████████████████████| 100%
STNID :  133 ...computing
gbm Model Build progress: |████████████████████████████████████████████████████████████| 100%
STNID :  140 ...computing
gbm Model Build progress: |████████████████████████████████████████████████████████████| 100%
STNID :  143 ...computing
gbm Model Build progress: |████████████████████████████████████████████████████████████| 100%
STNID :  146 ...computing
gbm Model Build progress: |████████████████████████████████████████████████████████████| 100%
```

● ● ●

auc_stnid

		STNID
0	0.959664	95
0	0.964828	101
0	0.958076	108
0	0.961040	114
0	0.970561	130
0	0.974591	133
0	0.975835	140
0	0.966820	143
0	0.972334	146
0	0.967359	156
0	0.913110	159
0	0.959527	162
0	0.995891	185
0	0.951964	112
0	0.981568	115
0	0.977483	131
0	0.983264	189
0	0.972865	192
0	0.984700	170
0	0.983161	138
0	0.937787	100
0	0.966235	119
0	0.960656	155
0	0.965416	136
0	0.969032	152
0	0.979427	165
0	0.957136	168
0	0.989772	184

< 실행결과 일부 생략>

본 문서의 내용은 기상청 날씨마루(https://bd.kma.go.kr)의
분석 플랫폼 활용을 위한 Python 프로그래밍 교육 자료 입니다.