

◆ 2022 / 10 / 04

Rain in Australia

2022년 파일럿 프로젝트 : 호주 강우 예측

Predict next-day rain in Australia

Name : 김태찬

E-Mail : ohapjijin135@gamil.com

목차

1. 개요

2. 데이터 설명

3. 데이터 분석 / 전처리


4. 데이터 시각화

5. 모델링

6. 평가

7. 결론

1. 개요

JOE YOUNG AND 1 COLLABORATOR · UPDATED 2 YEARS AGO

1465

New Notebook

Download (4 MB)

weatherAUS.csv (14.09 MB)

10 of 23 columns

Rain in Australia


Predict next-day rain in Australia

Data

Code (482)

Discussion (20)

Metadata



About this file

This dataset contains about 10 years of daily weather observations from numerous Australian weather stations.

RainTomorrow is the target variable to predict. It means -- did it rain the next day, Yes or No? This column is Yes if the rain for that day was 1mm or more.

About Dataset

Context

Predict next-day rain by training classification models on the target variable RainTomorrow.

Content

This dataset contains about 10 years of daily weather observations from many locations across Australia.

RainTomorrow is the target variable to predict. It means -- did it rain the next day, Yes or No? This column is Yes if the rain for that day was

Usability

10.00

License


Other (specified in description)

Expected update frequency

Never

Date

The date of observation



Location

The common name of the location of the weather station

MinTemp

The minimum temperature in degrees celsius

MaxTemp

The maximum temperature in degrees celsius

506 unique values

Rainfall

The amount of rainfall recorded for the day in mm

063%0.26%Other (45619)31%

Evaporati

The so-c: evaporati 24 hours

NA4Other (79

Canberra2%NA1%

Sydney2%111%

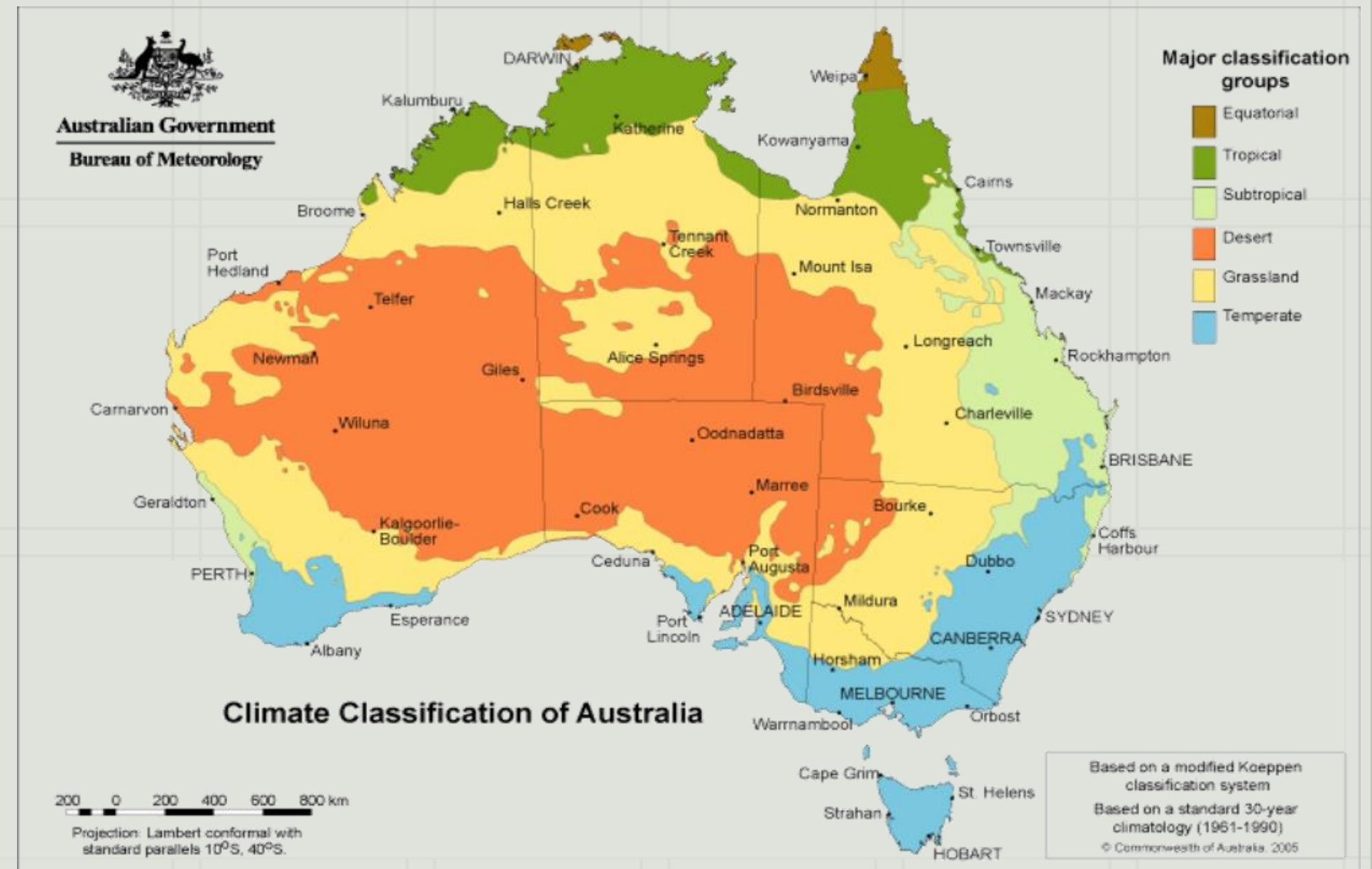
Other (138680)95%Other (143076)98%

<https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package>

1. 개요

목표 : 다음날 비가 올지 여부를 분류 모델을 사용하여 목표변수 **RainTomorrow**로써 예측한다.

설명 : 이 데이터셋은 호주 전역 다양한 장소에서 10년동안 매일 관측된 자료를 포함한다.
예측해야하는 목표 변수는 RainTomorrow로써 Yes or No로 나뉠 수 있다.
만약 Yes라면 다음날 비가 1mm 이상이라는 의미가 된다.



2. 데이터 설명

Features

Date : 관측 날짜
Location : 관측 장소
MinTemp : 최저 기온(섭씨) MaxTemp : 최고 기온(섭씨)
Rainfall : 일일 강수량(mm)
Evaporation : 증발량(mm)
Sunshine : 일사 시간
WindGustDir : 일일 최고 풍향
WindGustSpeed : 일일 최고 풍속
WindDir9am : 오전 9시 풍향
WindDir3pm : 오후 3시 풍향
WindSpeed9am : 오전 9시까지의 평균 풍속
WindSpeed3pm : 오후 3시까지의 평균 풍속
Humidity9am : 오전 9시의 습도(%)
Humidity3pm : 오후 3시의 습도(%)
Pressure9am : 오전 9시의 평균 해수면 대기압(hpa)
Pressure3pm : 오후 3시의 평균 해수면 대기압(hpa)
Cloud9am : 오전 9시의 운량(okta)
Cloud3pm : 오후 3시의 운량(okta)
Temp9am : 오전 9시의 온도(섭씨)
Temp3pm : 오후 3시의 온도(섭씨)
RainToday : 일일 강우 유무(1mm이상)

Target

RainTomorrow : 다음날 일일 강우 유무(1mm 이상)

3. 데이터 분석 / 전처리

연산/데이터 관련 라이브러리 호출

```
import numpy as np
import pandas as pd
```

시각화 관련 라이브러리 호출

```
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
```

기계학습 관련 라이브러리 호출

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
```

데이터 불러오기

```
df = pd.read_csv('weatherAUS.csv', index_col=None)
```

데이터 확인
df

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Hu
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	
...	

145460 rows × 23 columns 크기의 데이터

3. 데이터 분석 / 전처리

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype  
---  --
 0   Date                  145460 non-null object  
 1   Location              145460 non-null object  
 2   MinTemp               143975 non-null float64 
 3   MaxTemp               144199 non-null float64 
 4   Rainfall              142199 non-null float64 
 5   Evaporation           82670 non-null float64 
 6   Sunshine              75625 non-null float64 
 7   WindGustDir           135134 non-null object  
 8   WindGustSpeed          135197 non-null float64 
 9   WindDir9am            134894 non-null object  
10   WindDir3pm            141232 non-null object  
11   WindSpeed9am           143693 non-null float64 
12   WindSpeed3pm           142398 non-null float64 
13   Humidity9am            142806 non-null float64 
14   Humidity3pm            140953 non-null float64 
15   Pressure9am            130395 non-null float64 
16   Pressure3pm            130432 non-null float64 
17   Cloud9am               89572 non-null float64 
18   Cloud3pm               86102 non-null float64 
19   Temp9am                143693 non-null float64 
20   Temp3pm                141851 non-null float64 
21   RainToday              142199 non-null object  
22   RainTomorrow           142193 non-null object  
dtypes: float64(16), object(7)
```

살펴본 결과...

1. 결측치가 존재, 이를 해결해야 한다.
2. 기계학습을 위해 데이터 타입을 변환시킬 필요가 있어 보인다.
3. 정규화가 필요해 보인다.

다음 날 비가 올지를 예측하기 위해서는 기준이 필요
호주는 아주 큰 대륙이기 때문에 지역에 따라서 같은 시간이어도 날씨가
매우 다르기 때문
따라서 해당 데이터를 지역에 따라서 나누어 생각해 볼 필요가 있음.

16 개의 float64 데이터와 7개의 object 타입의 데이터가 존재

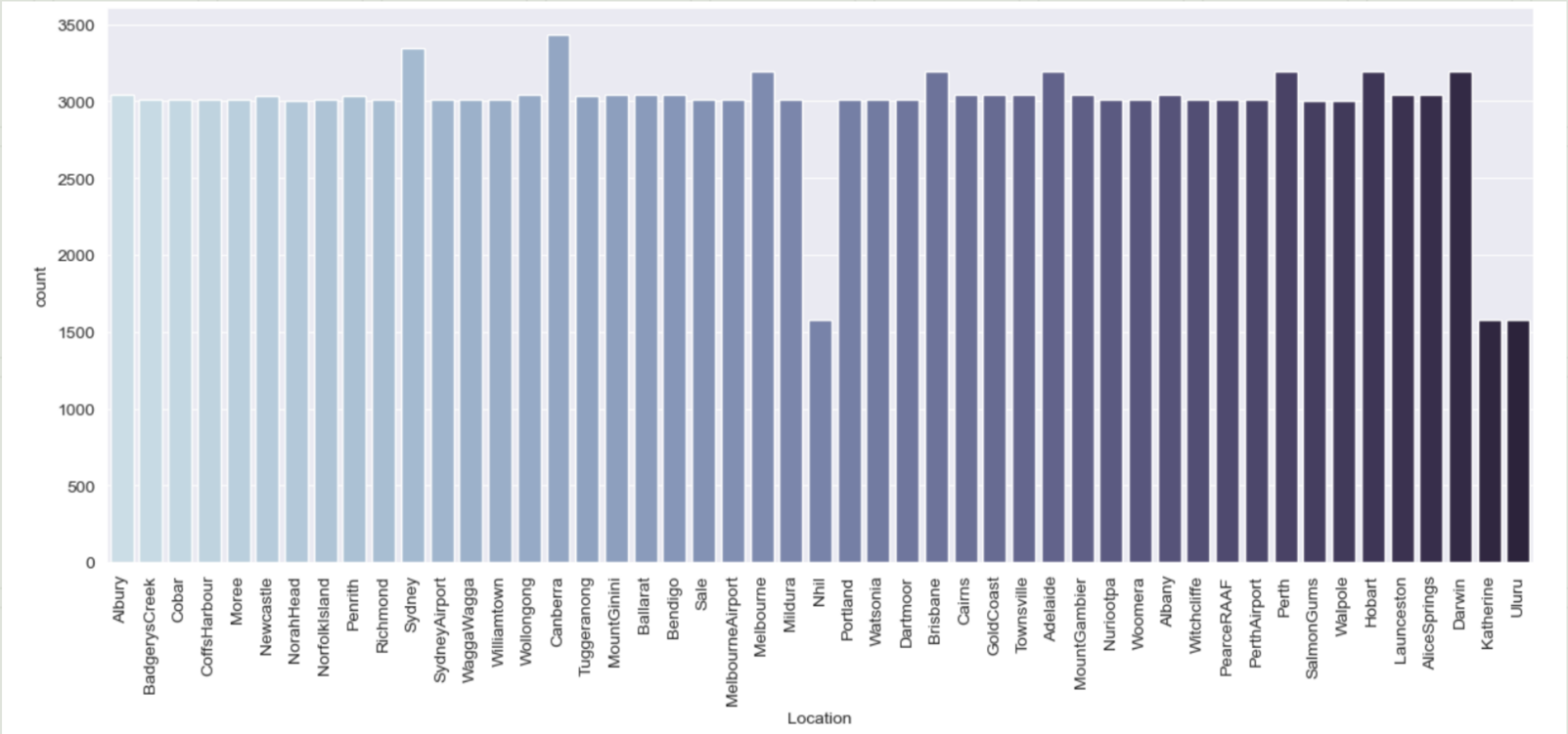
```
# 지역 확인
```

```
df['Location'].unique(), df['Location'].nunique()
```

```
(array(['Albury', 'BadgerysCreek', 'Cobar', 'CoffsHarbour', 'Moree',  
       'Newcastle', 'NorahHead', 'NorfolkIsland', 'Penrith', 'Richmond',  
       'Sydney', 'SydneyAirport', 'WaggaWagga', 'Williamtown',  
       'Wollongong', 'Canberra', 'Tuggeranong', 'MountGinini', 'Ballarat',  
       'Bendigo', 'Sale', 'MelbourneAirport', 'Melbourne', 'Mildura',  
       'Nhill', 'Portland', 'Watsonia', 'Dartmoor', 'Brisbane', 'Cairns',  
       'GoldCoast', 'Townsville', 'Adelaide', 'MountGambier', 'Nuriootpa',  
       'Woomera', 'Albany', 'Witchcliffe', 'PearceRAAF', 'PerthAirport',  
       'Perth', 'SalmonGums', 'Walpole', 'Hobart', 'Launceston',  
       'AliceSprings', 'Darwin', 'Katherine', 'Uluru'], dtype=object),  
49)
```

호주 전역의 49개 지역을 확인

3. 데이터 분석 / 전처리



2번째로 데이터가 많은 sydney를 선택

3. 데이터 분석 / 전처리

```
# 결측치 확인
sydney_df.isnull().sum()
```

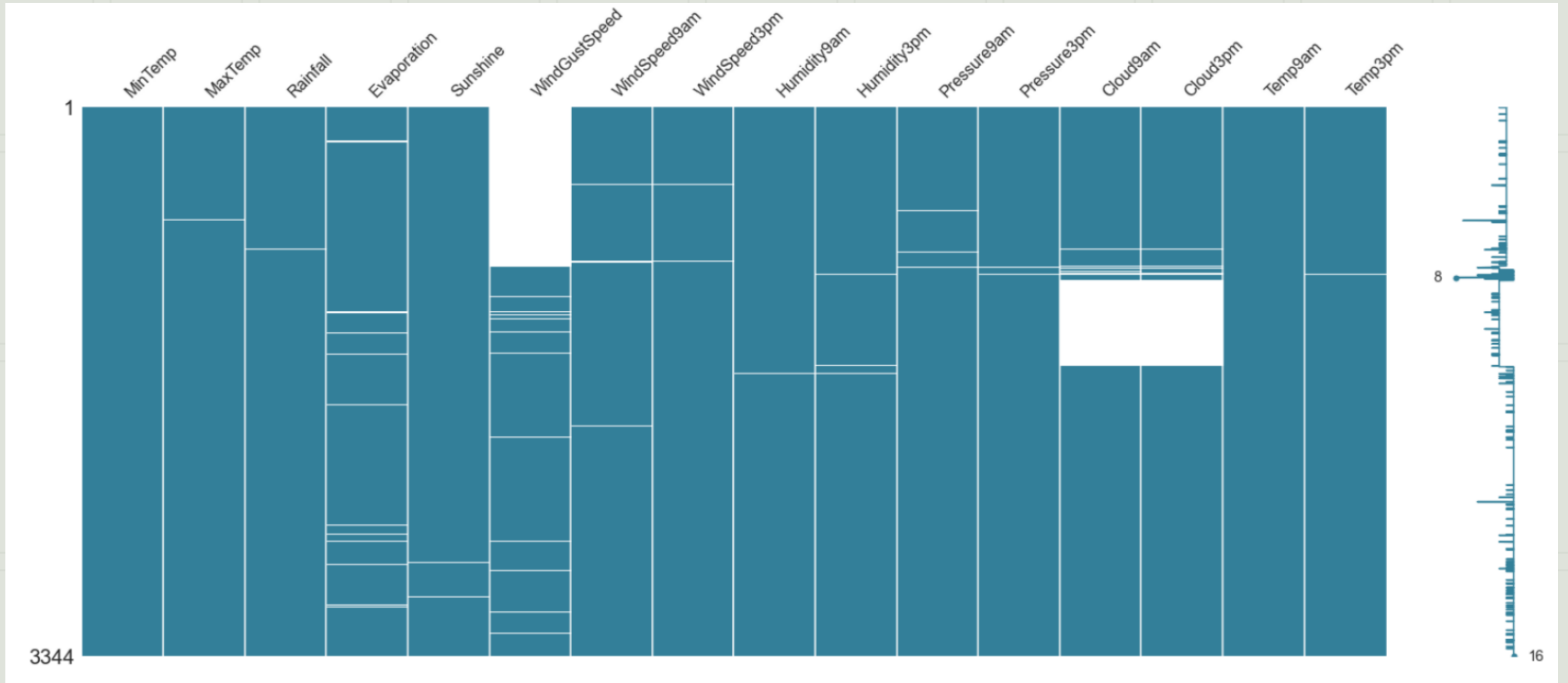
```
Date          0
Location       0
MinTemp        4
MaxTemp        2
Rainfall       7
Evaporation    51
Sunshine       16
WindGustDir    1038
WindGustSpeed  1038
WindDir9am     56
WindDir3pm     33
WindSpeed9am   26
WindSpeed3pm   25
Humidity9am    15
Humidity3pm    13
Pressure9am    21
Pressure3pm    19
Cloud9am       568
Cloud3pm       563
Temp9am        5
Temp3pm        4
RainToday      7
RainTomorrow   7
dtype: int64
```

```
# 데이터 타입이 연속형인 데이터만 추출
floats = sydney_df
for col in floats.columns:
    if floats[col].dtype != 'float64':
        floats = floats.drop(columns=col)
floats
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	Wind
30176	19.5	22.4	15.6	6.2	0.0	NaN	
30177	19.5	25.6	6.0	3.4	2.7	NaN	
30178	21.6	24.5	6.6	2.4	0.1	NaN	
30179	20.2	22.8	18.8	2.2	0.0	NaN	
30180	19.7	25.7	77.4	NaN	0.0	NaN	
...
33515	8.6	19.6	0.0	2.0	7.8	37.0	
33516	9.3	19.2	0.0	2.0	9.2	30.0	
33517	9.4	17.7	0.0	2.4	2.7	24.0	
33518	10.1	19.3	0.0	1.4	9.3	43.0	
33519	7.6	19.3	0.0	3.4	9.4	35.0	

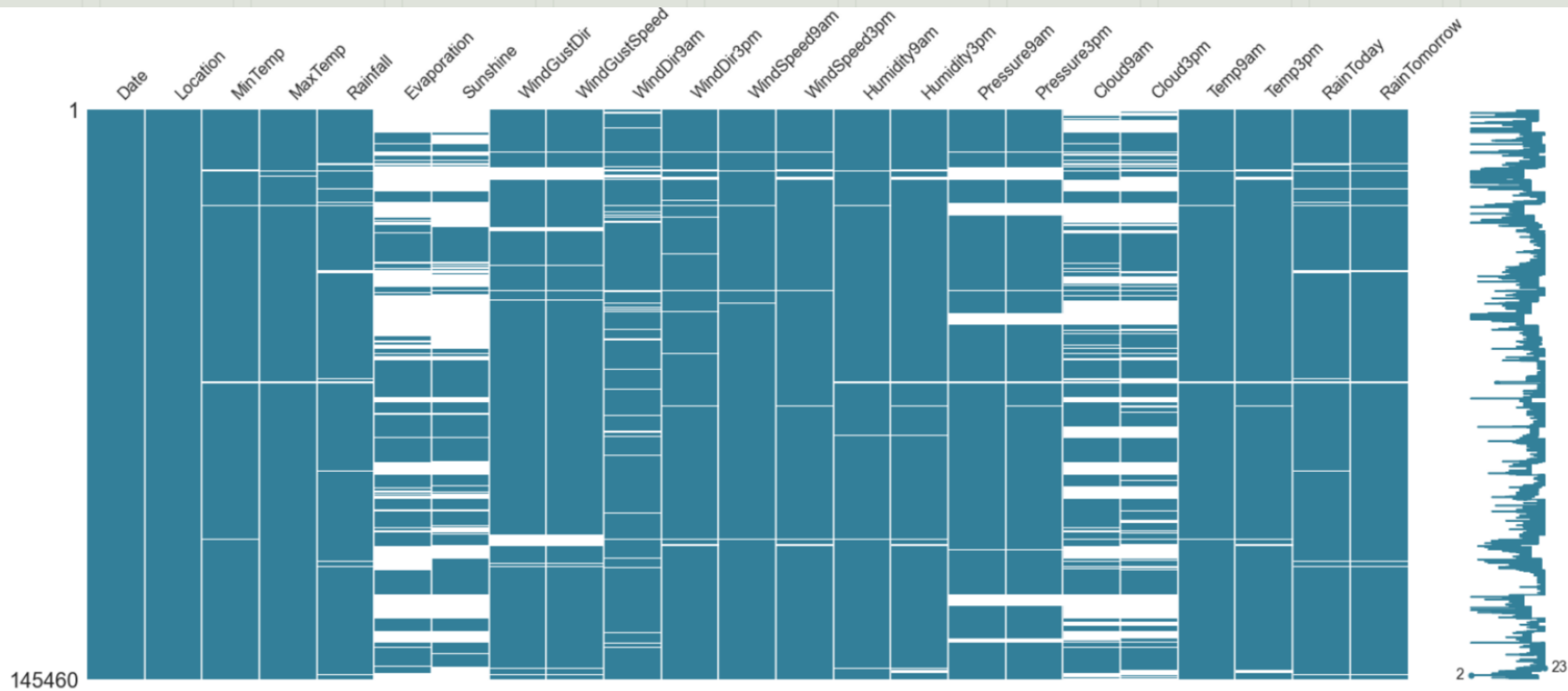
334 rows x 16 columns

결측치 시각화



살펴본 결과 결측치들은 모든 float 특징들에 존재하나,
최고 풍속과 오전, 오후 운량에서 일정기간 관측이 안됨을 알 수 있다.

결측치 시각화 (전체 데이터)



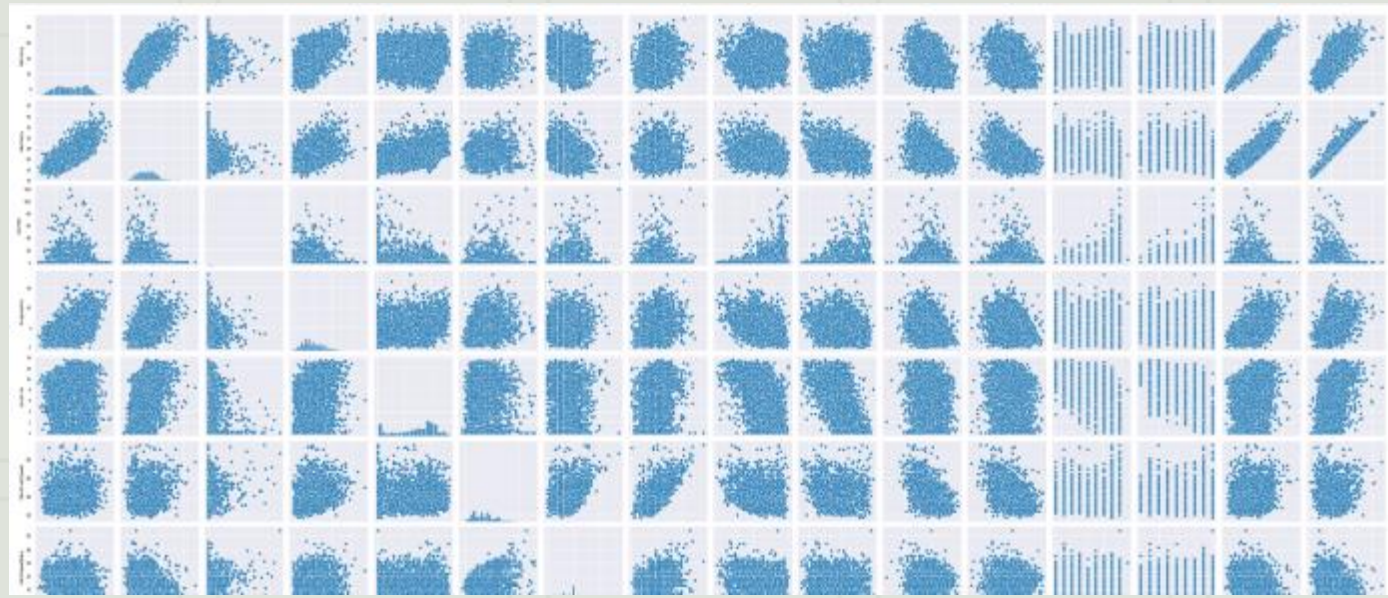
일정 기간동안 데이터가 수집되지 않은 경우...

1. 해당 특징을 사용하지 않는다.
2. 상관관계가 강한 다른 특징을 이용해 유추한다.

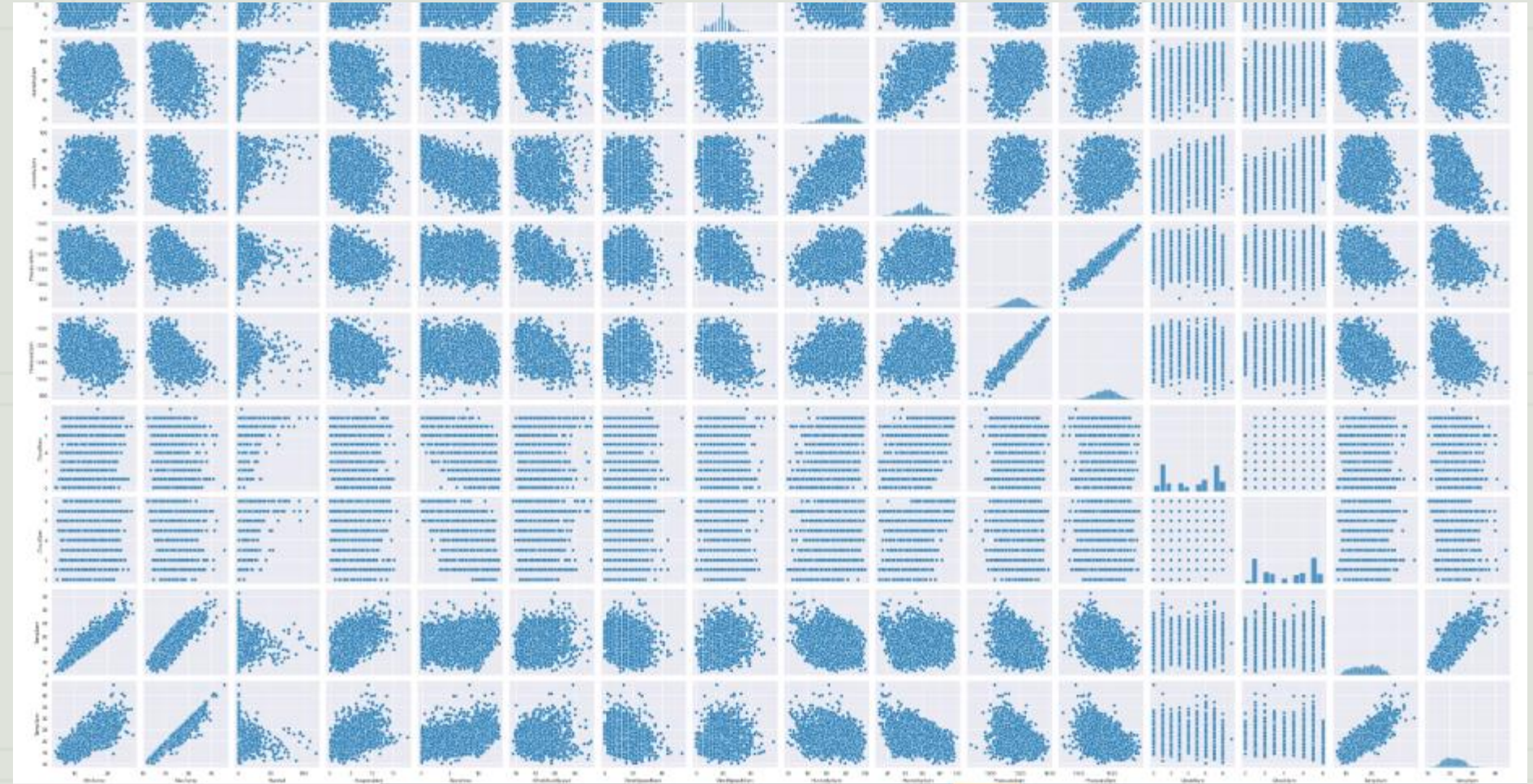
그 외에 연속적인 결측값의 빈도가 낮은 특징들은 근처 값들의 평균을 이용해 채운다.

3. 데이터 분석 / 전처리

변수들간 상관관계 파악하기



seaborn pair plot 그리기



변수들간 상관관계 파악하기

floats.corr(method='pearson')

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am
MinTemp	1.000000	0.770461	0.012735	0.571549	-0.063352	0.152936	-0.294792	0.223307	0.052596	0.271891	-0.393417
MaxTemp	0.770461	1.000000	-0.139968	0.513437	0.327355	0.088114	-0.346385	0.151797	-0.187342	-0.152548	-0.389251
Rainfall	0.012735	-0.139968	1.000000	-0.126332	-0.307399	0.172323	0.154055	0.039840	0.331066	0.301590	-0.036914
Evaporation	0.571549	0.513437	-0.126332	1.000000	0.177359	0.299394	-0.076591	0.299602	-0.398800	-0.102956	-0.349273
Sunshine	-0.063352	0.327355	-0.307399	0.177359	1.000000	-0.036461	-0.059561	0.177092	-0.490316	-0.585325	-0.047863
WindGustSpeed	0.152936	0.088114	0.172323	0.299394	-0.036461	1.000000	0.311453	0.608910	-0.271754	-0.117879	-0.487213
WindSpeed9am	-0.294792	-0.346385	0.154055	-0.076591	-0.059561	0.311453	1.000000	0.198502	-0.192183	-0.152051	0.002851
WindSpeed3pm	0.223307	0.151797	0.039840	0.299602	0.177092	0.608910	0.198502	1.000000	-0.290423	-0.135652	-0.330808
Humidity9am	0.052596	-0.187342	0.331066	-0.398800	-0.490316	-0.271754	-0.192183	-0.290423	1.000000	0.659305	0.263610
Humidity3pm	0.271891	-0.152548	0.301590	-0.102956	-0.585325	-0.117879	-0.152051	-0.135652	0.659305	1.000000	0.154757
Pressure9am	-0.393417	-0.389251	-0.036914	-0.349273	-0.047863	-0.487213	0.002851	-0.330808	0.263610	0.154757	1.000000
Pressure3pm	-0.355855	-0.425367	0.003328	-0.303301	-0.095963	-0.419857	0.091525	-0.275674	0.251942	0.183305	0.963163
Cloud9am	0.259959	-0.096941	0.277358	-0.041986	-0.734493	0.035586	-0.070941	-0.061411	0.459622	0.498215	-0.011775
Cloud3pm	0.180220	-0.108205	0.229394	-0.040464	-0.751219	0.069810	-0.018180	-0.100485	0.359511	0.517362	-0.043864
Temp9am	0.939044	0.859225	-0.056187	0.609325	0.115385	0.191401	-0.330320	0.276907	-0.148065	0.128969	-0.419369
Temp3pm	0.752707	0.960453	-0.143620	0.477929	0.345615	0.043393	-0.348675	0.132725	-0.157518	-0.213913	-0.346935

세 변수들 모두 0.7 이상의 상관계수를 가지는 변수들이 존재하지 않으나,
WindGustSpeed는 WindSpeed9am, WindSpeed3am에 0.6 이상의 상관관계를 가지고 있다.
Cloud9am과 Cloud3pm는 Sunshine과 0.6이상의 상관관계를 가지고 있다.

결측치 처리

- 방법 1(특징 제거)

#결측치 특징 제거

```
floats_1 = floats.drop(columns = ['WindGustSpeed', 'Cloud9am', 'Cloud3pm'])
```

#결측치 보간으로 처리

```
for col in floats_1.columns:  
    floats_1[col].interpolate(inplace=True)  
floats_1.isnull().sum()
```

```
MinTemp      0  
MaxTemp      0  
Rainfall     0  
Evaporation  0  
Sunshine     0  
WindSpeed9am 0  
WindSpeed3pm 0  
Humidity9am   0  
Humidity3pm   0  
Pressure9am   0  
Pressure3pm   0  
Temp9am       0  
Temp3pm       0  
dtype: int64
```

3. 데이터 분석 / 전처리

```
# Feature와 Target값을 하나의 데이터 프레임으로 만든다.
```

```
floats_1['RainTomorrow'] = sydney_df['RainTomorrow']
```

```
floats_1.set_index(sydney_df['Date'], inplace=True)
```

```
sydney_df1 = floats_1
```

```
# Target 인코딩
```

```
sydney_df1.loc[sydney_df1['RainTomorrow'] == 'Yes', 'RainTomorrow'] = 1
```

```
sydney_df1.loc[sydney_df1['RainTomorrow'] == 'No', 'RainTomorrow'] = 0
```

```
# RainfallT 추가
```

```
sydney_df1['RainfallT'] = sydney_df1['Rainfall'].shift(-1)
```

```
# sydney_df1.drop(columns='RainfallT', inplace=True)
```

```
sydney_df1['RainfallT'].fillna(0, inplace=True)
```

```
# Rainfall 결측치 처리
```

```
sydney_df1['RainTomorrow'].isnull()
```

```
sydney_df1[(sydney_df1['RainTomorrow'].isnull()) & (sydney_df1['Rainfall'] < 1)]['RainTomorrow'].fillna(0)
```

```
Date
```

```
2010-06-18    0
```

```
2010-10-09    0
```

```
2010-11-12    0
```

```
2014-09-16    0
```

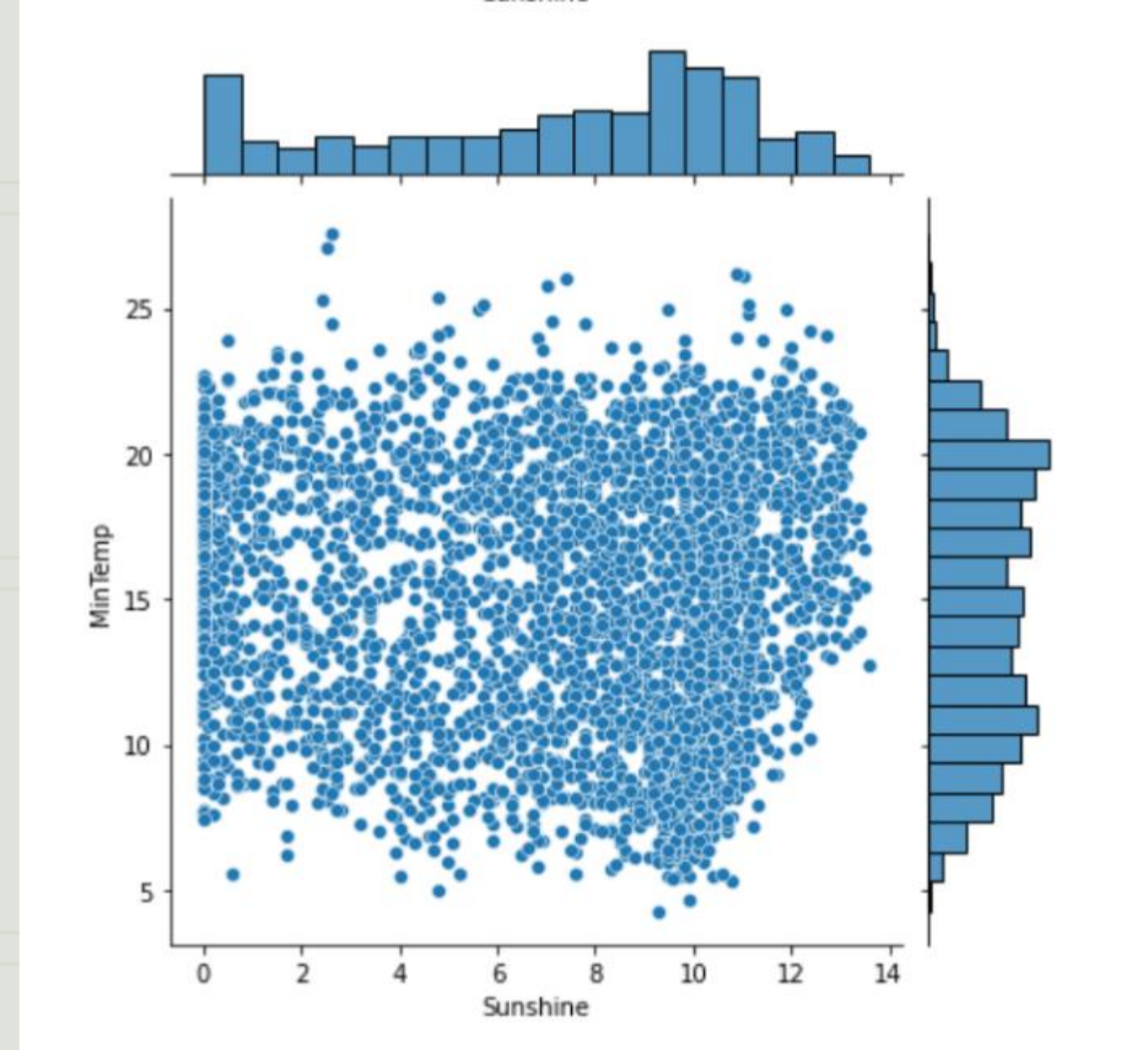
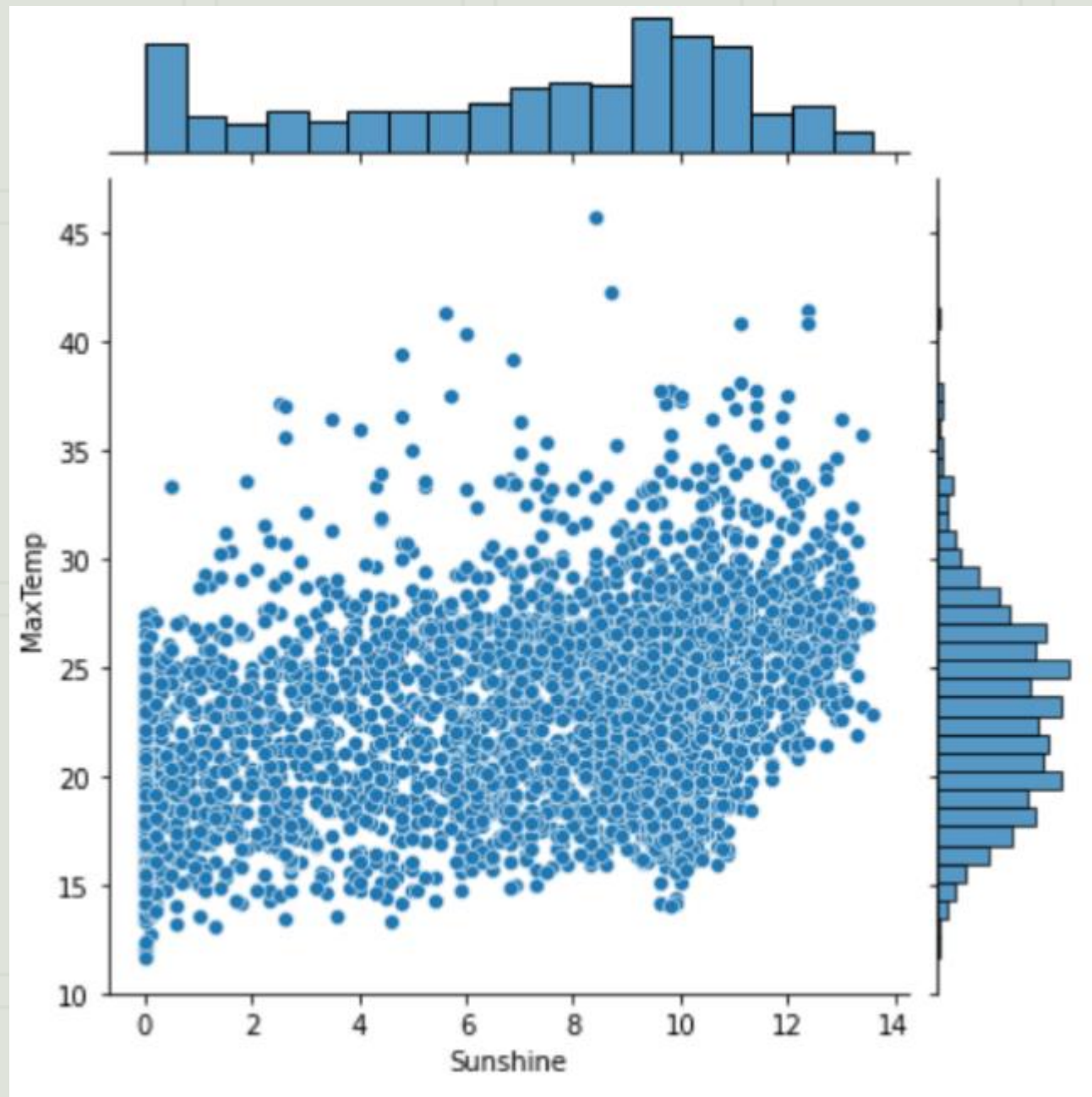
```
2014-10-31    0
```

```
2014-11-27    0
```

```
2014-11-28    0
```

```
Name: RainTomorrow, dtype: int64
```

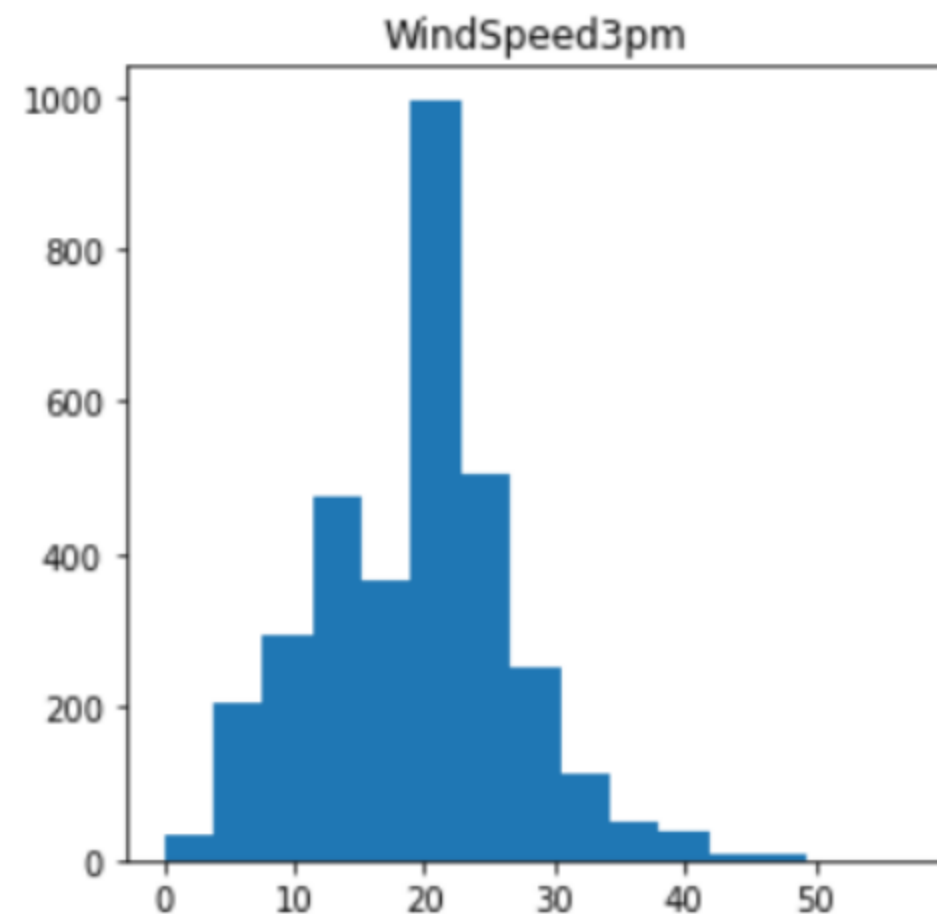
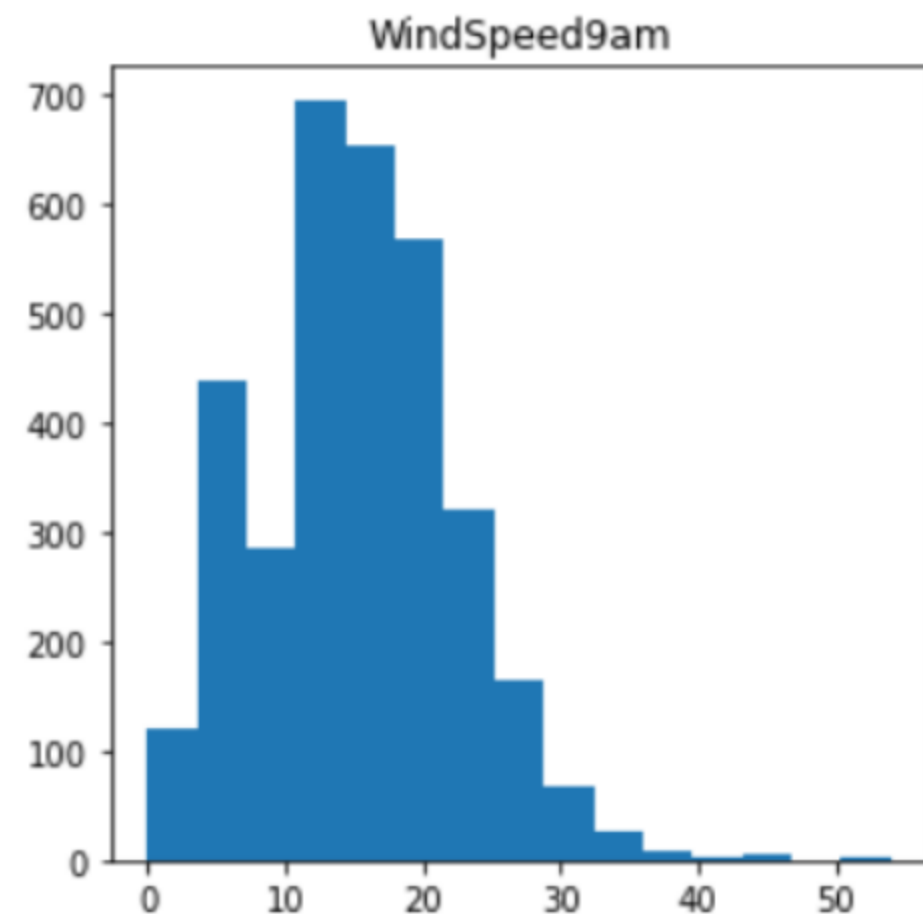
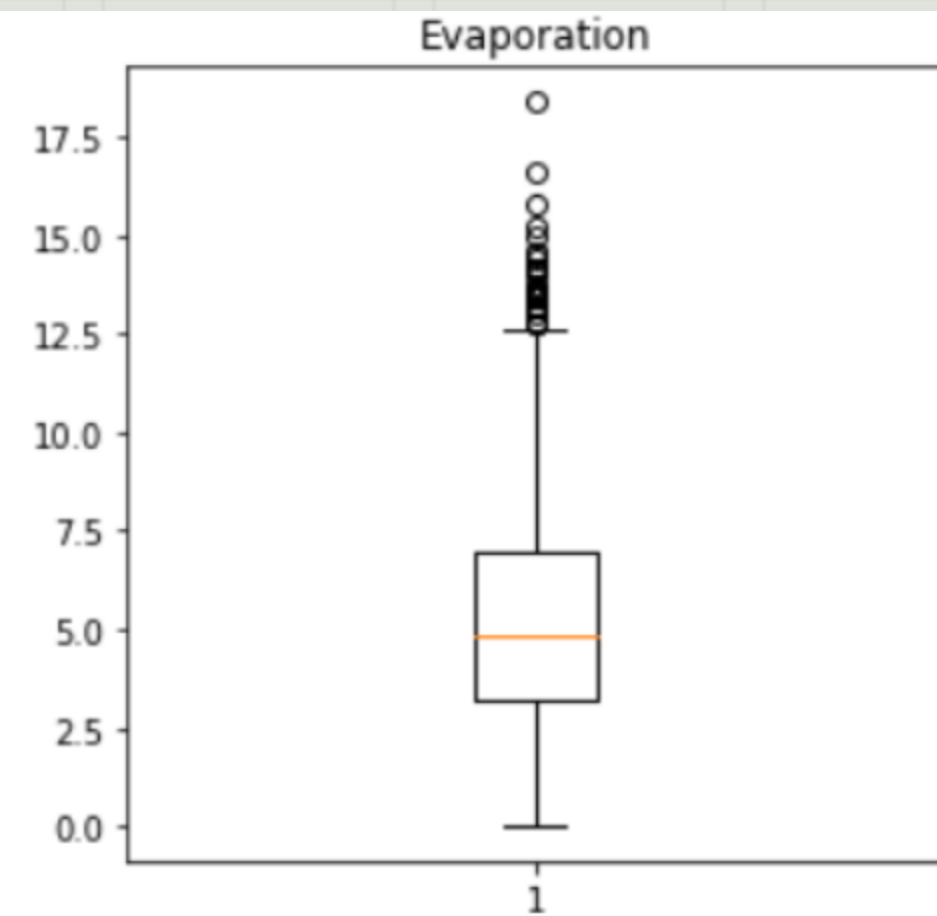
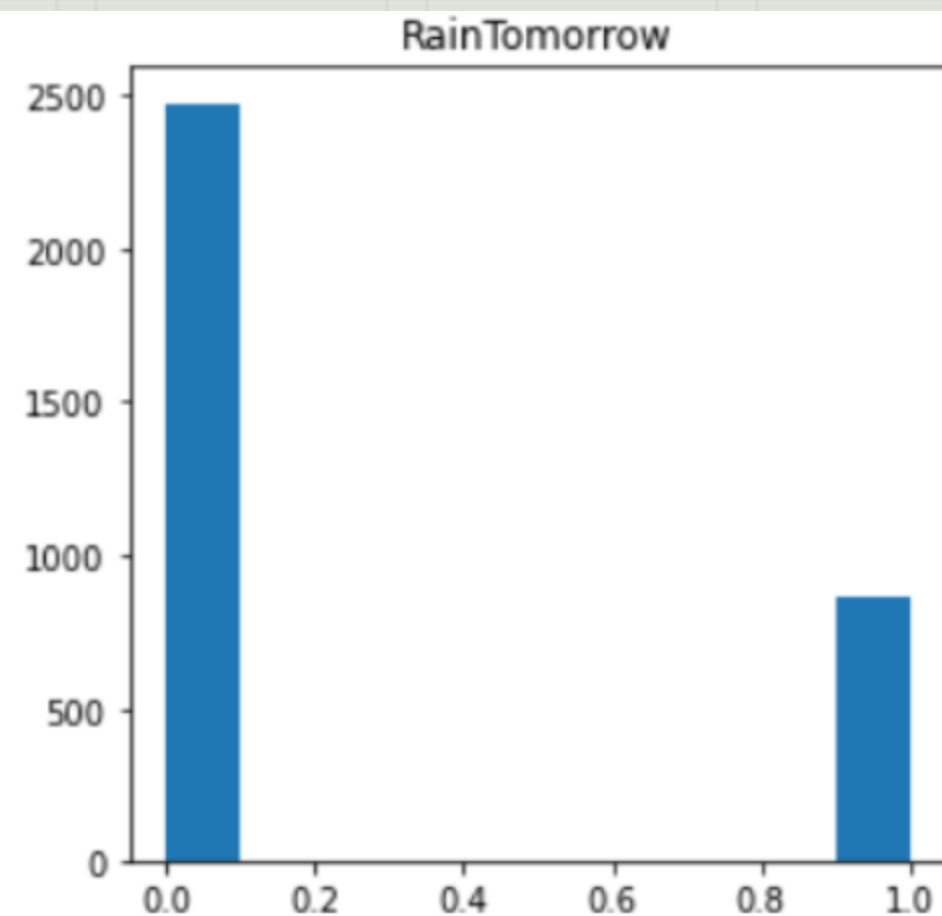
4. 데이터 시각화



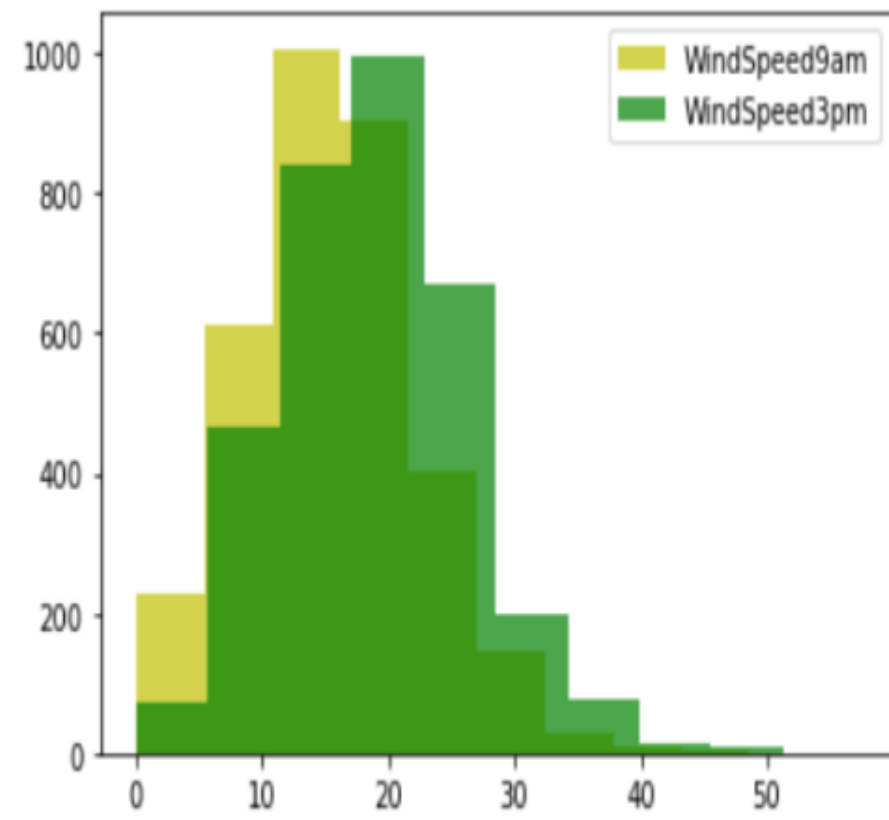
최고 온도와 일사량, 최저 온도와 일사량의 관계를 시각화
많은 상관성을 보이진 않음.

4. 데이터 시각화

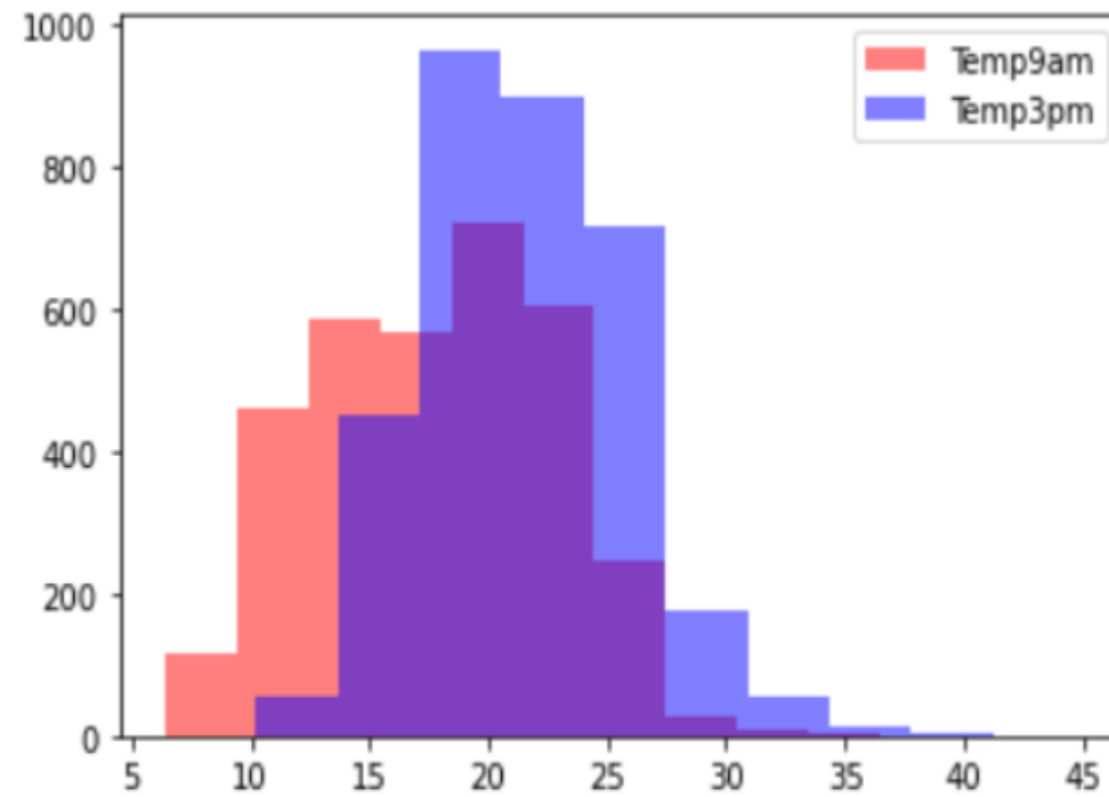
1. 전체적으로 비가 안온날이 더 많다.
2. 증발량은 5 정도의 중앙값을 가지고 있으며 날에 따라 편차가 큰 편
3. 바람은 오후가 대체로 강하게 분다.



4. 데이터 시각화



- 오전과 오후의 풍속의 변화를 나타내었다. 전반적으로 오후에 바람이 더 강하게 부는 경향이 있음을 알 수 있다.



- 오전과 오후의 온도 변화를 나타낸다. 전반적으로 오후의 온도가 더 높음을 알 수 있다.

5. 모델링

모델 성능에 영향을 줄 수 있는 요인들 :

- 1 . 어떤 특징(feature)들을 사용 할 것인가? (추가 or 제거)
- 2 . 결측치를 어떻게 다룰 것인가?
- 3 . 어떤 알고리즘을 이용해 학습시킬 것인가?
- 4 . 모델의 파라미터들이 최적인가?

5. 모델링

사용해 볼 모델 알고리즘 : KNN, 의사결정트리, 랜덤 포레스트

각각의 모델을 생성한 후 테스트 셋과의 정확도를 비교, 파라미터를 조정해 본다.

```
# X, y 설정하기
sel = ['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Temp9am', 'Temp3pm']
X = sydney_df1[sel]
y = sydney_df1['RainTomorrow']
# train, test 나누기
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

```
# KNN 모델 생성
model = KNeighborsClassifier(n_neighbors=8)
model.fit(X_train, y_train)
# 모델 예측
knn_pred = model.predict(X_test)
print("정확도 : {:.2f}".format(np.mean(knn_pred == y_test)))
```

정확도 : 0.82

5. 모델링

```
# k값에 따른 정확도 확인
tr_acc = []
test_acc = []
k_nums = range(1, 22)# 1,3,5~21

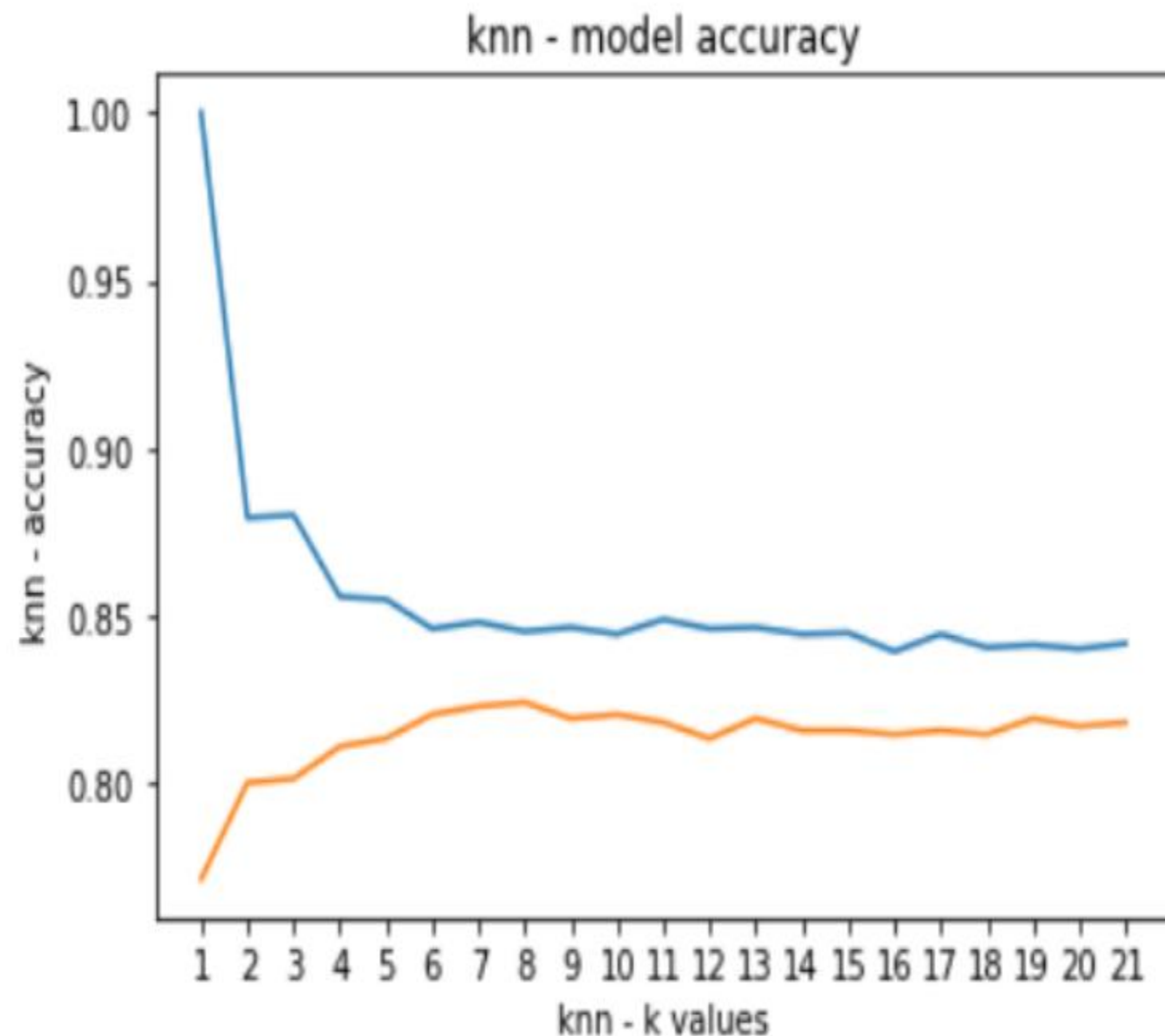
for n in k_nums:
    # 모델 선택 및 학습
    model = KNeighborsClassifier(n_neighbors=n)
    model.fit(X_train, y_train)

    # 정확도 구하기
    acc_tr = model.score(X_train, y_train)
    acc_test = model.score(X_test, y_test)

    # 정확도 값 저장.
    tr_acc.append(acc_tr)
    test_acc.append(acc_test)

print("k : ", n)
print("train 정확도 {:.3f}".format(acc_tr) )
print("test 정확도 {:.3f}".format(acc_test) )
```

```
train 정확도 0.855
test 정확도 0.813
k : 6
train 정확도 0.846
test 정확도 0.821
k : 7
train 정확도 0.848
test 정확도 0.823
k : 8
train 정확도 0.845
test 정확도 0.824
k : 9
train 정확도 0.846
test 정확도 0.819
k : 10
train 정확도 0.844
test 정확도 0.821
k : 11
train 정확도 0.849
test 정확도 0.818
k : 12
train 정확도 0.846
test 정확도 0.813
k : 13
train 정확도 0.846
test 정확도 0.819
k : 14
train 정확도 0.844
test 정확도 0.816
k : 15
train 정확도 0.845
test 정확도 0.816
k : 16
train 정확도 0.839
test 정확도 0.815
k : 17
train 정확도 0.844
test 정확도 0.816
```



5. 모델링

```
# 의사결정트리
# 모델 생성 및 학습
model = DecisionTreeClassifier(max_depth=3, criterion="entropy", random_state=0).fit(X_train, y_train)
# 예측
tree_pred = model.predict(X_test)
from sklearn import metrics
# Model Accuracy, 얼마나 정확한가? 정확도
print("Accuracy:", metrics.accuracy_score(y_test, tree_pred))
```

Accuracy: 0.8133971291866029

5. 모델링

```
# max_depth값에 따른 정확도 확인
tr_acc = []
test_acc = []
max_nums = range(1, 22)# 1,3,5~21

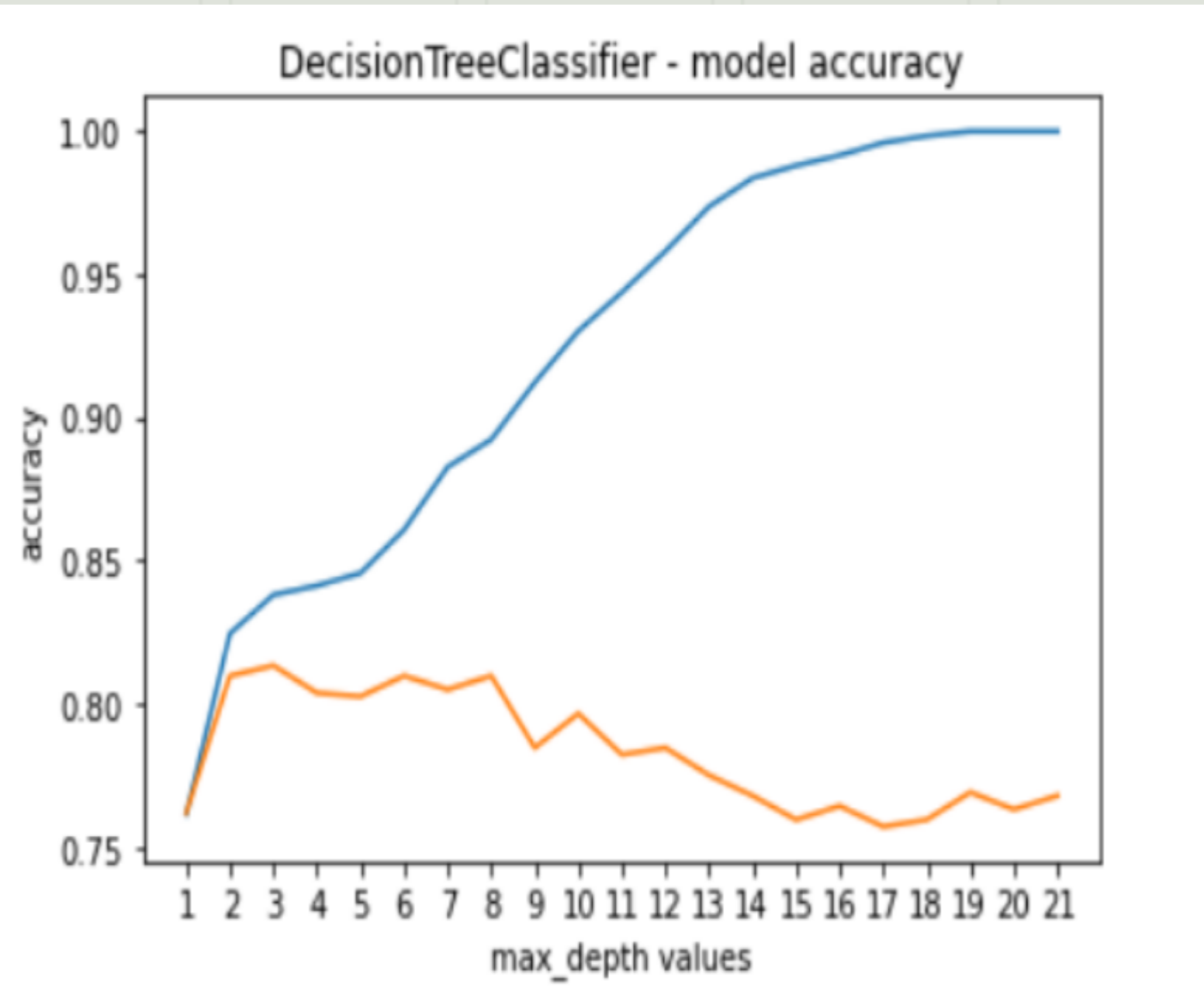
for n in max_nums:
    # 모델 선택 및 학습
    model = DecisionTreeClassifier(max_depth=n, criterion="entropy")
    model.fit(X_train, y_train)

    # 정확도 구하기
    acc_tr = model.score(X_train, y_train)
    acc_test = model.score(X_test, y_test)

    # 정확도 값 저장.
    tr_acc.append(acc_tr)
    test_acc.append(acc_test)

    print("max_depth : ", n)
    print("train 정확도 {:.3f}".format(acc_tr) )
    print("test 정확도 {:.3f}".format(acc_test) )
```

```
max_depth : 1
train 정확도 0.762
test 정확도 0.762
max_depth : 2
train 정확도 0.825
test 정확도 0.810
max_depth : 3
train 정확도 0.838
test 정확도 0.813
max_depth : 4
train 정확도 0.841
test 정확도 0.804
max_depth : 5
train 정확도 0.846
test 정확도 0.803
max_depth : 6
train 정확도 0.861
test 정확도 0.810
max_depth : 7
train 정확도 0.883
test 정확도 0.805
max_depth : 8
train 정확도 0.892
test 정확도 0.810
max_depth : 9
train 정확도 0.912
test 정확도 0.785
max_depth : 10
train 정확도 0.930
test 정확도 0.797
max_depth : 11
train 정확도 0.944
test 정확도 0.782
```



5. 모델링

랜덤 포레스트

```
model = RandomForestClassifier(n_estimators = 25) # 모델 만들기
model.fit(X_train, y_train) # 모델 훈련시키기 model.fit(입력, 출력)
r_pred = model.predict(X_test) # 학습된 모델로 예측하기
### model.score()를 이용해서 구하기
print( model.score(X_train, y_train) )
print( model.score(X_test, y_test) )
```

0.9992025518341308

0.8409090909090909

n_estimators값에 따른 정확도 확인

tr_acc = []

test_acc = []

n_nums = range(1, 60, 3) # 1, 3, 5~21

for n in n_nums:

모델 선택 및 학습

model = RandomForestClassifier(n_estimators=n)

model.fit(X_train, y_train)

정확도 구하기

acc_tr = model.score(X_train, y_train)

acc_test = model.score(X_test, y_test)

정확도 값 저장.

tr_acc.append(acc_tr)

test_acc.append(acc_test)

print("n_estimators : ", n)

print("train 정확도 {:.3f}".format(acc_tr))

print("test 정확도 {:.3f}".format(acc_test))

train 정확도 0.992

test 정확도 0.822

n_estimators : 19

train 정확도 0.996

test 정확도 0.825

n_estimators : 22

train 정확도 0.996

test 정확도 0.835

n_estimators : 25

train 정확도 0.997

test 정확도 0.827

n_estimators : 28

train 정확도 0.999

test 정확도 0.833

n_estimators : 31

train 정확도 1.000

test 정확도 0.841

n_estimators : 34

train 정확도 0.998

test 정확도 0.833

n_estimators : 37

train 정확도 1.000

test 정확도 0.825

n_estimators : 40

train 정확도 0.999

test 정확도 0.834

n_estimators : 43

train 정확도 0.999

test 정확도 0.842

n_estimators : 46

train 정확도 1.000

test 정확도 0.847

n_estimators : 49

train 정확도 1.000

test 정확도 0.833

n_estimators : 52

train 정확도 1.000

test 정확도 0.834

5. 모델링

랜덤 포레스트

```
model = RandomForestClassifier(n_estimators = 25) # 모델 만들기
model.fit(X_train, y_train) # 모델 훈련시키기 model.fit(입력, 출력)
r_pred = model.predict(X_test) # 학습된 모델로 예측하기
### model.score()를 이용해서 구하기
print( model.score(X_train, y_train) )
print( model.score(X_test, y_test) )
```

0.9992025518341308

0.8409090909090909

n_estimators값에 따른 정확도 확인

tr_acc = []

test_acc = []

n_nums = range(1, 60, 3) # 1, 3, 5 ~ 21

for n in n_nums:

모델 선택 및 학습

model = RandomForestClassifier(n_estimators=n)

model.fit(X_train, y_train)

정확도 구하기

acc_tr = model.score(X_train, y_train)

acc_test = model.score(X_test, y_test)

정확도 값 저장.

tr_acc.append(acc_tr)

test_acc.append(acc_test)

print("n_estimators : ", n)

print("train 정확도 {:.3f}".format(acc_tr))

print("test 정확도 {:.3f}".format(acc_test))

train 정확도 0.992

test 정확도 0.822

n_estimators : 19

train 정확도 0.996

test 정확도 0.825

n_estimators : 22

train 정확도 0.996

test 정확도 0.835

n_estimators : 25

train 정확도 0.997

test 정확도 0.827

n_estimators : 28

train 정확도 0.999

test 정확도 0.833

n_estimators : 31

train 정확도 1.000

test 정확도 0.841

n_estimators : 34

train 정확도 0.998

test 정확도 0.833

n_estimators : 37

train 정확도 1.000

test 정확도 0.825

n_estimators : 40

train 정확도 0.999

test 정확도 0.834

n_estimators : 43

train 정확도 0.999

test 정확도 0.842

n_estimators : 46

train 정확도 1.000

test 정확도 0.847

n_estimators : 49

train 정확도 1.000

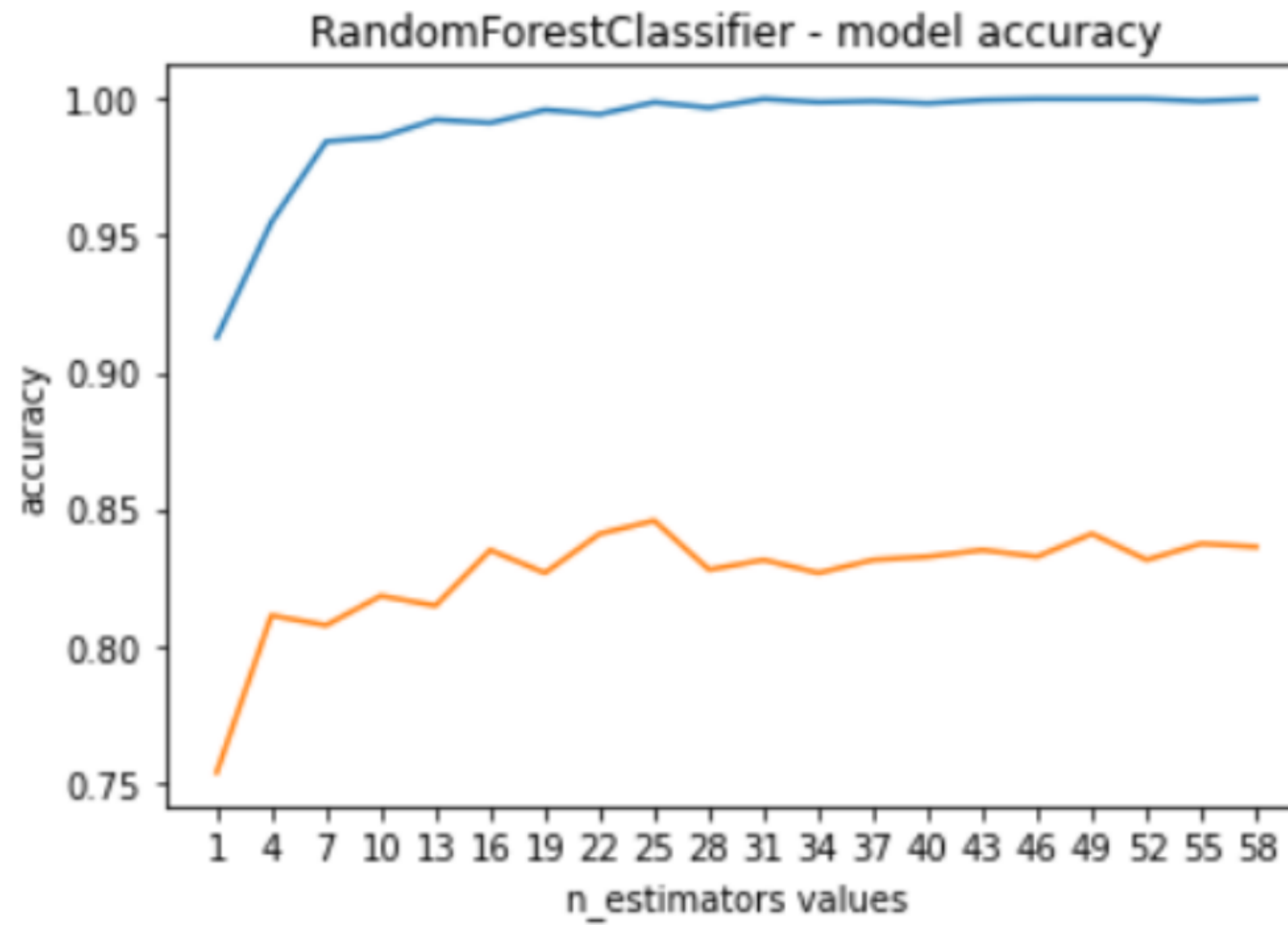
test 정확도 0.833

n_estimators : 52

train 정확도 1.000

test 정확도 0.834

5. 모델링



F1 - score 를 사용해 모델을 평가

각 모델 별 점수

```
# 모델 평가
#F1-score를 활용
from sklearn.metrics import f1_score
print("KNN 모델의 f1 score: {:.2f}".format(f1_score(y_test, knn_pred)))
print("트리 모델의 f1 score: {:.2f}".format(f1_score(y_test, tree_pred)))
print("랜덤포레스트 모델의 f1 score: {:.2f}".format(f1_score(y_test, r_pred)))
```

KNN 모델의 f1 score: 0.58

트리 모델의 f1 score: 0.66

랜덤포레스트 모델의 f1 score: 0.68

6. 평가

```
# KNN f1-score
from sklearn.metrics import classification_report
print(classification_report(y_test, knn_pred,
                           target_names=["no rain", "yes rain"]))
```

	precision	recall	f1-score	support
no rain	0.82	0.96	0.89	
yes rain	0.81	0.45	0.58	
accuracy			0.82	
macro avg	0.82	0.71	0.73	
weighted avg	0.82	0.82	0.80	

```
# DecisionTree f1-score
from sklearn.metrics import classification_report
print(classification_report(y_test, tree_pred,
                           target_names=["no rain", "yes rain"]))
```

	precision	recall	f1-score	support
no rain	0.86	0.92	0.89	607
yes rain	0.74	0.59	0.66	229
accuracy			0.83	836
macro avg	0.80	0.75	0.77	836
weighted avg	0.82	0.83	0.82	836

```
# RandomForest f1-score
from sklearn.metrics import classification_report
print(classification_report(y_test, r_pred,
                           target_names=["no rain", "yes rain"]))
```

	precision	recall	f1-score	support
no rain	0.86	0.95	0.90	607
yes rain	0.80	0.59	0.68	229
accuracy			0.85	836
macro avg	0.83	0.77	0.79	836
weighted avg	0.84	0.85	0.84	836

KNN < Decision Tree < Random Forest

호주 강우 예측을 위해 데이터를 선택하고 각각의 모델을 만들어 평가해 보았다.

랜덤포레스트 모델로 f1-score 0.68의 예측 모델을 만들 수 있었다.

추가로 고려해 볼 사항들

1. sydney 외의 다른 지역들의 모델은 어떨까?
2. Categorical 변수들도 사용한다면? 그 방법은?
3. 빈도가 높은 결측치를 보완하면 성능이 향상 될 것인가?
4. 다른 알고리즘들은 어떨까?

References

Kaggle Data : <https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package>

Observations were drawn from numerous weather stations. The daily observations are available from <http://www.bom.gov.au/climate/data>.

An example of latest weather observations in Canberra: <http://www.bom.gov.au/climate/dwo/IDCJDW2801.latest.shtml> Definitions adapted from <http://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml> Data source: <http://www.bom.gov.au/climate/dwo/> and <http://www.bom.gov.au/climate/data>.

Copyright Commonwealth of Australia 2010, Bureau of Meteorology.

Thank
you

지금까지 발표를 들어주셔서 감사합니다!