t2_room_classify_recap

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

```
```

```
train=pd.read_csv('data/room_occupacy/datatraining.csv')
test=pd.read_csv('data/room_occupacy/datatest.csv')
```

```
train.head(2)
```

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
0	2015-02-04 17:51:00	23.18	27.2720	426.0	721.25	0.004793	1
1	2015-02-04 17:51:00	23.15	27.2675	429.5	714.00	0.004783	1

```
test.head(2)
```

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
0	2015-02-02 14:19:00	23.700	26.272	585.2	749.2	0.004764	1
1	2015-02-02 14:19:00	23.718	26.290	578.4	760.4	0.004773	1

```
train.tail(2)
```

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
8141	2015-02-10 09:32:00	21.1	36.26	433.0	820.333333	0.005621	1
8142	2015-02-10 09:33:00	21.1	36.20	447.0	821.000000	0.005612	1

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8143 entries, 0 to 8142
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           8143 non-null   object 
 1   Temperature    8143 non-null   float64
 2   Humidity       8143 non-null   float64
 3   Light          8143 non-null   float64
 4   CO2            8143 non-null   float64
 5   HumidityRatio  8143 non-null   float64
 6   Occupancy      8143 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 445.4+ KB
```

```
train.isnull().sum()
```

```
date             0
Temperature      0
Humidity         0
Light            0
CO2              0
HumidityRatio    0
Occupancy        0
dtype: int64
```

```
train['date']=pd.to_datetime(train['date'])
```

```
test['date']=pd.to_datetime(test['date'])
```

```
# help(pd.to_datetime)
```

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8143 entries, 0 to 8142
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           8143 non-null   datetime64[ns]
 1   Temperature    8143 non-null   float64
 2   Humidity       8143 non-null   float64
 3   Light          8143 non-null   float64
 4   CO2            8143 non-null   float64
 5   HumidityRatio  8143 non-null   float64
 6   Occupancy      8143 non-null   int64
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 445.4 KB
```

```
train['hour']=train['date'].dt.hour
```

```
train['min']=train['date'].dt.minute
```

```
train['day']=train['date'].dt.day
```

```
test['hour']=test['date'].dt.hour
test['min']=test['date'].dt.minute
test['day']=test['date'].dt.day
```

```
train.drop(['date'],axis=1,inplace=True)
test.drop(['date'],axis=1,inplace=True)
```

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8143 entries, 0 to 8142
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Temperature    8143 non-null   float64
 1   Humidity       8143 non-null   float64
 2   Light          8143 non-null   float64
 3   CO2            8143 non-null   float64
 4   HumidityRatio  8143 non-null   float64
 5   Occupancy      8143 non-null   int64  
 6   hour           8143 non-null   int32  
 7   min            8143 non-null   int32  
 8   day            8143 non-null   int32  
dtypes: float64(5), int32(3), int64(1)
memory usage: 477.3 KB
```

```
train.shape,test.shape
```

```
((8143, 9), (2665, 9))
```

```
target= train.pop('Occupancy')
test_target= test.pop('Occupancy')
```

```
train.shape,test.shape
```

```
((8143, 8), (2665, 8))
```

```
# 원핫인코딩
train= pd.get_dummies(train)
test= pd.get_dummies(test)
```

```
train.shape,test.shape
```

```
((8143, 8), (2665, 8))
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
```

```
rf=RandomForestClassifier(random_state=42, max_depth=7, n_estimators=500)
```

```
score= cross_val_score(rf,train,target,cv=2)
```

```
score
```

```
array([0.94965619, 0.98943748])
```

```
cross_val_score(rf,train,target,cv=2)
```

```
array([0.94965619, 0.98943748])
```

```
score.mean()
```

```
0.9695468366263074
```

```
# 모델 학습 및 평가
```

```
rf.fit(train,target)
```

```
RandomForestClassifier(max_depth=7, n_estimators=500, random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
pred=rf.predict(test)
```

```
from sklearn.metrics import roc_auc_score, accuracy_score,f1_score,recall_score
```

```
import sklearn.metrics
# dir(sklearn.metrics)
```

```
scores= roc_auc_score(test_target,pred)
```

```
scores
```

```
0.9566457988473478
```

```
accuracy_score(test_target,pred)
```

```
0.9602251407129456
```

```
f1_score(test_target,pred)
```

```
0.945360824742268
```

```
recall_score(test_target,pred)
```

```
0.9434156378600823
```

평가지표	설명	함수명
Accuracy	전체 중 맞춘 비율	`accuracy_score(y_true, y_pred)`
F1 Score	정밀도와 재현율의 조화 평균	`f1_score(y_true, y_pred)`
Precision	양성 예측 중 실제 양성 비율	`precision_score(y_true, y_pred)`
Recall	실제 양성 중 모델이 맞춘 비율	`recall_score(y_true, y_pred)`
AUC	ROC 커브 아래 면적	`roc_auc_score(y_true, y_pred_proba)`

평가지표	설명	함수명 (with `average`)
Accuracy	전체 중 맞춘 비율	`accuracy_score(y_true, y_pred)`
F1 Macro	클래스별 F1 평균	`f1_score(y_true, y_pred, average='macro')`
Precision Macro	클래스별 정밀도 평균	`precision_score(y_true, y_pred, average='macro')`
Recall Macro	클래스별 재현율 평균	`recall_score(y_true, y_pred, average='macro')`

평가지표	설명	함수명
R^2 Score	설명력 (1에 가까울수록 좋음)	`r2_score(y_true, y_pred)`
MSE	평균 제곱 오차	`mean_squared_error(y_true, y_pred)`
RMSE	평균 제곱근 오차	`mean_squared_error(..., squared=False)`
MAE	평균 절대 오차	`mean_absolute_error(y_true, y_pred)`