

Ch15. 머신러닝을 이용한 예측 분석

머신러닝(Machine Learning)은 컴퓨터 프로그램이 데이터에서 학습하고 경험을 통해 성능을 향상시키는 인공지능의 한 분야

머신러닝 모델은 머신러닝 알고리즘이 특정 작업을 수행하기 위해 학습한 결과물(판단 근거 형태: 모델)

> 모델은 데이터를 입력으로 받아 원하는 결과를 출력으로 내놓는 함수의 형태

15-1. 머신러닝 모델 알아보기

머신러닝 모델을 만들 때 사용하는 두 가지 변수

> **예측 변수(predictor variable):** 예측하는 데 활용하는 변수 또는 모델에 입력하는 값

> **타겟 변수(target variable):** 예측하고자 하는 변수 또는 모델이 출력하는 값

In []:

15-2. 소득 예측 모델 만들기

adult.csv는 미국인의 성별, 인종, 직업, 학력 등 다양한 인적 정보를 담고 있는 인구조사 데이터

In [3]:

```
## 인구조사 데이터 구축
import pandas as pd
df = pd.read_csv('adult.csv')
df
```

Out[3]:

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship
0	25	Private	226802	11th		7	Never-married	Machine-op-inspt
1	38	Private	89814	HS-grad		9	Married-civ-spouse	Farming-fishing
2	28	Local-gov	336951	Assoc-acdm		12	Married-civ-spouse	Protective-serv
3	44	Private	160323	Some-college		10	Married-civ-spouse	Machine-op-inspt
4	18	?	103497	Some-college		10	Never-married	?
...
48837	27	Private	257302	Assoc-acdm		12	Married-civ-spouse	Tech-support
48838	40	Private	154374	HS-grad		9	Married-civ-spouse	Machine-op-inspt
48839	58	Private	151910	HS-grad		9	Widowed	Adm-clerical
48840	22	Private	201490	HS-grad		9	Never-married	Adm-clerical
48841	52	Self-emp-inc	287927	HS-grad		9	Married-civ-spouse	Exec-managerial

48842 rows × 15 columns

In [4]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   age         48842 non-null   int64  
 1   workclass   48842 non-null   object  
 2   fnlwgt     48842 non-null   int64  
 3   education   48842 non-null   object  
 4   education_num 48842 non-null   int64  
 5   marital_status 48842 non-null   object  
 6   occupation   48842 non-null   object  
 7   relationship 48842 non-null   object  
 8   race        48842 non-null   object  
 9   sex         48842 non-null   object  
 10  capital_gain 48842 non-null   int64  
 11  capital_loss 48842 non-null   int64  
 12  hours_per_week 48842 non-null   int64  
 13  native_country 48842 non-null   object  
 14  income       48842 non-null   object  
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

adult 컬럼 구성

age(나이), workclass(근로 형태), fnlwgt(인구통계 가중치), education(학력), education_num(교육 기간) marital_status(결혼 상태), occupation(직종), relationship(가구주와의 관계), race(인종), sex(성별) capital_gain(자본 소득), hours_per_week(주당 근무 시간), native_country(출신 국가), income(연소득)

In []:

전처리하기

1. 타겟 변수 전처리

In [5]:
연소득 데이터에 대한 빈도 수 분포 알아보기
df['income'].value_counts(normalize = True)

Out[5]:
income
<=50K ... 0.760718
>50K ... 0.239282
Name: proportion, dtype: float64

In [6]:
연소득 데이터 보기
df['income']

Out[6]:
0 ... <=50K
1 ... <=50K
2 ... >50K
3 ... >50K
4 ... <=50K
...
48837 ... <=50K
48838 ... >50K
48839 ... <=50K
48840 ... <=50K
48841 ... >50K
Name: income, Length: 48842, dtype: object

In [7]:
연소득 데이터 값 변경
import numpy as np
df['income'] = np.where(df['income'] == '>50K', 'high', 'low')
df['income'].value_counts(normalize = True)

Out[7]:
income
low ... 0.760718
high ... 0.239282
Name: proportion, dtype: float64

2. 불필요한 변수 제거하기

In [8]:
인구 통계 가중치 컬럼 제거하기
인구 통계 특성이 같으면 fnlwgt 값이 같으므로 제거
df = df.drop(columns = 'fnlwgt')

In []:

3. 문자 타입 변수를 숫자 타입으로 바꾸기

모델에 사용되는 모든 값은 숫자 형태여야 함

원핫 인코딩하기

문자열 값을 갖는 컬럼을 대상으로 함

원핫(one-hot) 인코딩은 값의 범주에 따라 0 또는 1로 변환하는 방법

즉 인코딩 대상 컬럼 대신에 존재 할 수 있는 값에 대한 컬럼으로 변환되어 False 또는 True 값을 갖음

pandas.get_dummies() 함수 사용

기존의 컬럼 대신에 컬럼에 있는 컬럼명_값 컬럼으로 재구성 됨

재구성된 컬럼 값은 False 또는 True

```
In [9]: df['sex']
```

```
Out[9]: 0      Male
1      Male
2      Male
3      Male
4    Female
...
48837  Female
48838   Male
48839  Female
48840   Male
48841  Female
Name: sex, Length: 48842, dtype: object
```

```
In [10]: ##[연습] 성별만으로 데이터 프레임 구축
df_tmp = df[['sex']]
df_tmp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 1 columns):
 # ... Column ... Non-Null Count Dtype ...
 --- ... -- ...
 0   sex      48842 non-null  object
dtypes: object(1)
memory usage: 381.7+ KB
```

```
In [11]: df_tmp
```

```
Out[11]:
```

sex	
0	Male
1	Male
2	Male
3	Male
4	Female
...	...
48837	Female
48838	Male
48839	Female
48840	Male
48841	Female

48842 rows × 1 columns

```
In [12]: df_tmp['sex'].value_counts()
```

```
Out[12]:
```

```
sex
Male     32650
Female    16192
Name: count, dtype: int64
```

```
In [13]: ##[연습] df_tmp의 문자 타입 변수에 원핫 인코딩 적용
df_tmp = pd.get_dummies(df_tmp)
df_tmp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 2 columns):
 # ... Column ... Non-Null Count Dtype ...
 --- ... --- ...
 0 sex_Female 48842 non-null bool
 1 sex_Male   48842 non-null bool
dtypes: bool(2)
memory usage: 95.5 KB
```

```
In [14]: df_tmp
```

Out[14]:

	sex_Female	sex_Male
0	False	True
1	False	True
2	False	True
3	False	True
4	True	False
...
48837	True	False
48838	False	True
48839	True	False
48840	False	True
48841	True	False

48842 rows × 2 columns

In [15]:

```
## 연소득(income: 'high' or 'low') 컬럼만 제외하고 원핫 인코딩 변환 적용
# 원핫 인코딩은 문자열 값을 갖는 컬럼을 대상으로 함
# income 컬럼은 타겟 변수(target variable)로 사용
target = df['income'] # income 추출

df = df.drop(columns = 'income') # income 제거
df = pd.get_dummies(df) # 문자 타입 변수 원핫 인코딩

df['income'] = target # df에 target 삽입
df
```

Out[15]:

	age	education_num	capital_gain	capital_loss	hours_per_week	workclass_?	workclass_Fed
0	25	7	0	0	40	False	
1	38	9	0	0	50	False	
2	28	12	0	0	40	False	
3	44	10	7688	0	40	False	
4	18	10	0	0	30	True	
...
48837	27	12	0	0	38	False	
48838	40	9	0	0	40	False	
48839	58	9	0	0	40	False	
48840	22	9	0	0	20	False	
48841	52	9	15024	0	40	False	

48842 rows × 108 columns

In [16]:

```
## 컬럼이 100개 이상이면 컬럼 정보 출력을 안함 : df는 총 108개 컬럼
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Columns: 108 entries, age to income
dtypes: bool(102), int64(5), object(1)
memory usage: 7.0+ MB
```

```
In [17]: ## 컬럼이 100개 이상이면 컬럼 정보 출력을 안함
import numpy as np
df.info(max_cols = np.inf) #무한대로 보기
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 108 columns):
 #   Column          Non-Null Count Dtype
 --- 
 0   age             48842 non-null int64
 1   education_num  48842 non-null int64
 2   capital_gain   48842 non-null int64
 3   capital_loss   48842 non-null int64
 4   hours_per_week 48842 non-null int64
 5   workclass_?     48842 non-null bool
 6   workclass_Federal-gov 48842 non-null bool
 7   workclass_Local-gov 48842 non-null bool
 8   workclass_Never-worked 48842 non-null bool
 9   workclass_Private 48842 non-null bool
 10  workclass_Self-emp-inc 48842 non-null bool
 11  workclass_Self-emp-not-inc 48842 non-null bool
 12  workclass_State-gov 48842 non-null bool
 13  workclass_Without-pay 48842 non-null bool
 14  education_10th    48842 non-null bool
 15  education_11th    48842 non-null bool
 16  education_12th    48842 non-null bool
 17  education_1st-4th 48842 non-null bool
 18  education_5th-6th 48842 non-null bool
 19  education_7th-8th 48842 non-null bool
 20  education_9th    48842 non-null bool
 21  education_Assoc-acdm 48842 non-null bool
 22  education_Assoc-voc 48842 non-null bool
 23  education_Bachelors 48842 non-null bool
 24  education_Documentary 48842 non-null bool
 25  education_HS-grad 48842 non-null bool
 26  education_Masters 48842 non-null bool
 27  education_Preschool 48842 non-null bool
 28  education_Prof-school 48842 non-null bool
 29  education_Some-college 48842 non-null bool
 30  marital_status_Divorced 48842 non-null bool
 31  marital_status_Married-AF-spouse 48842 non-null bool
 32  marital_status_Married-civ-spouse 48842 non-null bool
 33  marital_status_Married-spouse-absent 48842 non-null bool
 34  marital_status_Never-married 48842 non-null bool
 35  marital_status_Separated 48842 non-null bool
 36  marital_status_Widowed 48842 non-null bool
 37  occupation_? 48842 non-null bool
 38  occupation_Adm-clerical 48842 non-null bool
 39  occupation_Armed-Forces 48842 non-null bool
 40  occupation_Craft-repair 48842 non-null bool
 41  occupation_Exec-managerial 48842 non-null bool
 42  occupation_Farming-fishing 48842 non-null bool
 43  occupation_Handlers-cleaners 48842 non-null bool
 44  occupation_Machine-op-inspct 48842 non-null bool
 45  occupation_Other-service 48842 non-null bool
 46  occupation_Priv-house-serv 48842 non-null bool
 47  occupation_Prof-specialty 48842 non-null bool
 48  occupation_Protective-serv 48842 non-null bool
 49  occupation_Sales 48842 non-null bool
 50  occupation_Tech-support 48842 non-null bool
 51  occupation_Transport-moving 48842 non-null bool
 52  relationship_Husband 48842 non-null bool
 53  relationship_Not-in-family 48842 non-null bool
 54  relationship_Other-relative 48842 non-null bool
 55  relationship_Own-child 48842 non-null bool
 56  relationship_Unmarried 48842 non-null bool
 57  relationship_Wife 48842 non-null bool
 58  race_Amer-Indian-Eskimo 48842 non-null bool

```

```
59 race_Asian-Pac-Islander ..... 48842 non-null bool
60 race_Black ..... 48842 non-null bool
61 race_Other ..... 48842 non-null bool
62 race_White ..... 48842 non-null bool
63 sex_Female ..... 48842 non-null bool
64 sex_Male ..... 48842 non-null bool
65 native_country_? ..... 48842 non-null bool
66 native_country_Cambodia ..... 48842 non-null bool
67 native_country_Canada ..... 48842 non-null bool
68 native_country_China ..... 48842 non-null bool
69 native_country_Columbia ..... 48842 non-null bool
70 native_country_Cuba ..... 48842 non-null bool
71 native_country_Dominican-Republic ..... 48842 non-null bool
72 native_country_Ecuador ..... 48842 non-null bool
73 native_country_El-Salvador ..... 48842 non-null bool
74 native_country_England ..... 48842 non-null bool
75 native_country_France ..... 48842 non-null bool
76 native_country_Germany ..... 48842 non-null bool
77 native_country_Greece ..... 48842 non-null bool
78 native_country_Guatemala ..... 48842 non-null bool
79 native_country_Haiti ..... 48842 non-null bool
80 native_country_Holand-Netherlands ..... 48842 non-null bool
81 native_country_Honduras ..... 48842 non-null bool
82 native_country_Hong ..... 48842 non-null bool
83 native_country_Hungary ..... 48842 non-null bool
84 native_country_India ..... 48842 non-null bool
85 native_country_Iran ..... 48842 non-null bool
86 native_country_Ireland ..... 48842 non-null bool
87 native_country_Italy ..... 48842 non-null bool
88 native_country_Jamaica ..... 48842 non-null bool
89 native_country_Japan ..... 48842 non-null bool
90 native_country_Laos ..... 48842 non-null bool
91 native_country_Mexico ..... 48842 non-null bool
92 native_country_Nicaragua ..... 48842 non-null bool
93 native_country_Outlying-US(Guam-USVI-etc) ..... 48842 non-null bool
94 native_country_Peru ..... 48842 non-null bool
95 native_country_Phippines ..... 48842 non-null bool
96 native_country_Poland ..... 48842 non-null bool
97 native_country_Portugal ..... 48842 non-null bool
98 native_country_Puerto-Rico ..... 48842 non-null bool
99 native_country_Scotland ..... 48842 non-null bool
100 native_country_South ..... 48842 non-null bool
101 native_country_Taiwan ..... 48842 non-null bool
102 native_country_Thailand ..... 48842 non-null bool
103 native_country_Trinadad&Tobago ..... 48842 non-null bool
104 native_country_United-States ..... 48842 non-null bool
105 native_country_Vietnam ..... 48842 non-null bool
106 native_country_Yugoslavia ..... 48842 non-null bool
107 income ..... 48842 non-null object
dtypes: bool(102), int64(5), object(1)
memory usage: 7.0+ MB
```

```
In [18]: ## iloc[,] > 행, 열 범위 추출
import numpy as np
df.iloc[:,0:6].info() #df의 모든 행에 대해 6개 시작 열만 검색
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   age              48842 non-null   int64  
 1   education_num    48842 non-null   int64  
 2   capital_gain     48842 non-null   int64  
 3   capital_loss     48842 non-null   int64  
 4   hours_per_week   48842 non-null   int64  
 5   workclass_?      48842 non-null   bool  
dtypes: bool(1), int64(5)
memory usage: 1.9 MB
```

```
In [19]: df.iloc[:,5:12] #df의 모든 행에 대해 6개 시작 열만 검색
```

	workclass_?	workclass_Federal-gov	workclass_Local-gov	workclass_Never-worked	workclass_Private	workclass_Other-gov
0	False	False	False	False	True	False
1	False	False	False	False	True	False
2	False	False	True	False	False	False
3	False	False	False	False	True	False
4	True	False	False	False	False	False
...
48837	False	False	False	False	True	False
48838	False	False	False	False	True	False
48839	False	False	False	False	True	False
48840	False	False	False	False	True	False
48841	False	False	False	False	False	False

48842 rows × 7 columns

4. 데이터 분할하기

- 분할 방법: 모델을 형성할 데이터 군(training set)과 모델의 성능을 확인할 데이터 군(test set)으로 분할
 - 분할 이유: 모델을 형성한 데이터로 모델의 성능을 검증하면 의미가 없음
 - 분할 정도: 모델 형성 비중이 크므로 더 크게 형성 ##### **adult** 데이터 분할하기

```
C:\Users\ADMIN\anaconda3\lib\site-packages\scipy\__init__.py:155: UserWarning: A NumPy version >=1.18.5 and <1.25.0 is required for this version of SciPy (detected version 1.25.2)
... warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

```
In [21]: ## training set : 70% rows
df_train.shape
```

```
Out[21]: (34189, 108)
```

```
In [22]: ## test set : 30% rows
df_test.shape
```

```
Out[22]: (14653, 108)
```

```
In [23]: ## 타겟 변수 income의 비율 유지 확인 : training set
df_train['income'].value_counts(normalize = True)
```

```
Out[23]: income
low .... 0.760713
high ... 0.239287
Name: proportion, dtype: float64
```

```
In [24]: ## 타겟 변수 income의 비율 유지 확인 : test set
df_test['income'].value_counts(normalize = True)
```

```
Out[24]: income
low .... 0.760732
high ... 0.239268
Name: proportion, dtype: float64
```

```
In [ ]:
```

의사결정나무 모델 만들기

모델 설정하기

```
In [25]: ## 의사결정Tree 인스턴스 만들기
from sklearn import tree
clf = tree.DecisionTreeClassifier(random_state = 1234, # 난수 고정
                                   max_depth = 3)           # 나무 깊이
```

모델 만들기

```
In [26]: ## 예측 변수와 타겟 변수로 의사결정Tree의 모델 만들기
train_x = df_train.drop(columns = 'income') # 예측 변수 추출
train_y = df_train['income']                 # 타겟 변수 추출

model = clf.fit(X = train_x, y = train_y)    # 모델 만들기
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\sklearn\utils\validation.py:623: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
... if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
```

```
In [ ]:
```

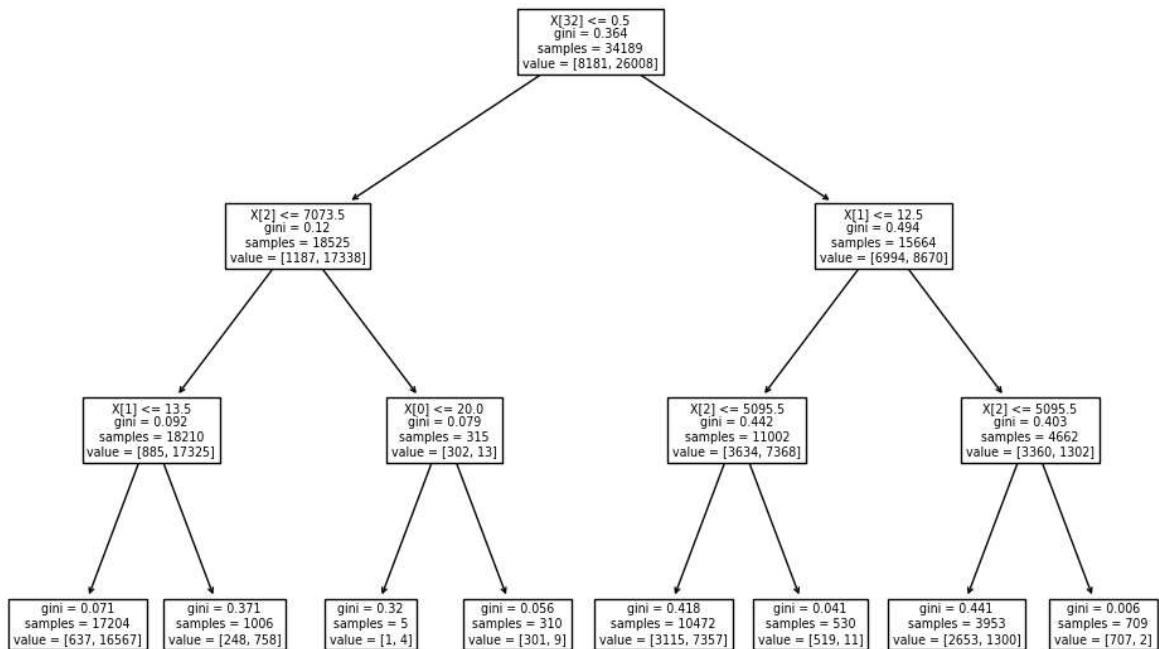
모델 구조 살펴보기

matplotlib.pyplot.tree.plot_tree()로 모델 그리기

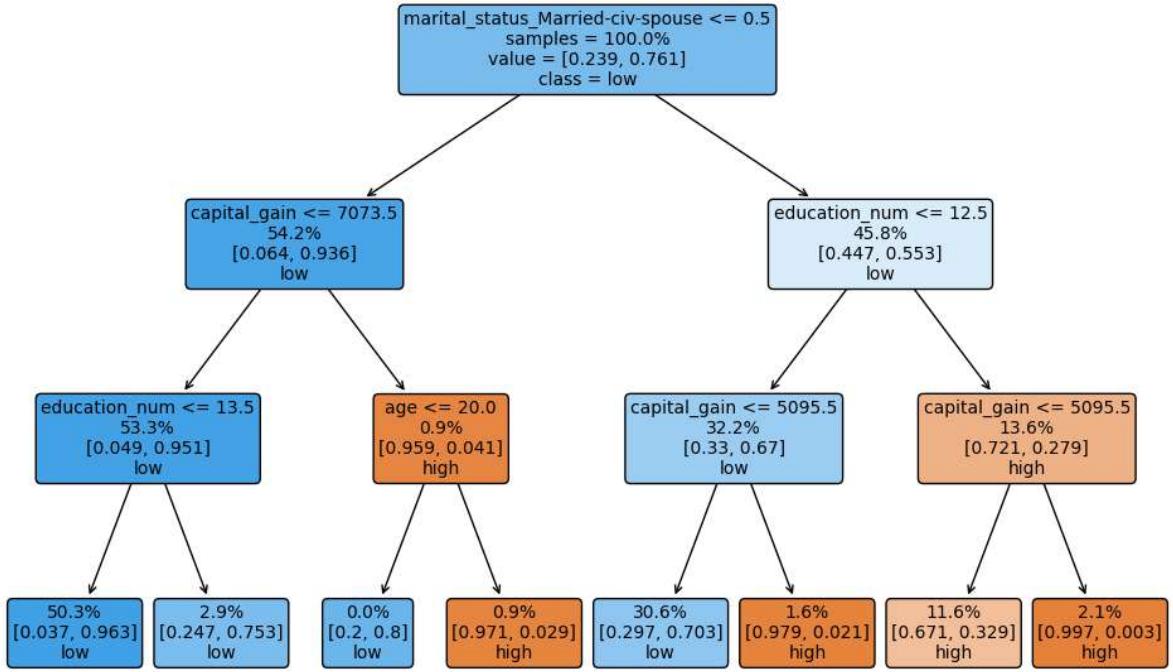
타깃 변수를 잘 예측할 수 있는 변수(좌, 우 분할 비율이 큰)를 Tree의 상위에 위치 시킴

```
In [27]: # 그래프 설정하기
import matplotlib.pyplot as plt
plt.rcParams.update({'figure.dpi' : '100',      # 그래프 크기 설정
                     'figure.figsize' : [12, 8]}) # 해상도 설정
```

```
In [28]: ## 의사결정Tree 모델 그리기
tree.plot_tree(model);                      # 그래프 출력
```



```
In [29]: ## 의사결정Tree 모델 그리기 : 매개변수 적용
tree.plot_tree(model,
               feature_names = train_x.columns,    # 예측 변수명
               class_names = ['high', 'low'],       # 타겟 변수 클래스, 알파벳순
               proportion = True,                 # 비율 표기
               filled = True,                   # 색칠
               rounded = True,                  # 둥근 테두리
               impurity = False,                # 불순도 표시
               label = 'root',                  # label 표시 위치
               fontsize = 10);                 # 글자 크기
```



의사결정Tree 해석

Node의 값 표기

marital_status_Married-civ-spouse <= 0.5 : 하위 노드를 분리할 때 사용할 기준
(기혼은 1(True), 비혼은 0)\ samples = 100.0% : 현재 노드에서의 좌우 분류 비율\ value = [0.239, 0.761] : 타깃 변수의 클래스별 비율 (알파벳 순)\ class = low : 우세한 타깃 변수의 클래스\ 우세에 따라 노드 색상 및 농도 구분

Node의 값 해석

Level-0 : [예측변수 선택] 결혼 여부가 'Married-civ-spouse'이 True와 False를 반씩 추출 > 수입이 low인 비율이 76.1%로 우세 예측\ Level-1 : 결혼 여부가 'Married-civ-spouse'이 True인 대상(54.2%)은 > 수입이 low인 비율이 93.6%로 우세 예측\ Level-1 : 결혼 여부가 'Married-civ-spouse'이 False인 대상(45.8%)은 > 수입이 low인 비율이 55.3%로 우세 예측\ Level-2 : 'Married-civ-spouse'이 True이고 자본소득이 <= 7073.5인 대상(53.3%)은 > 수입이 low인 비율이 95.1%로 우세 예측\ Level-2 : 'Married-civ-spouse'이 True이고 자본소득이 > 7073.5인 대상(0.9%)은 > 수입이 high인 비율이 95.9%로 우세 예측\ Level-2 : 'Married-civ-spouse'이 False이고 교육 기간이 <= 12.5인 대상(32.2%)은 > 수입이 low인 비율이 67.0%로 우세 예측\ Level-2 : 'Married-civ-spouse'이 False이고 교육 기간이 > 12.5인 대상(13.6%)은 > 수입이 high인 비율이 72.1%로 우세 예측

[예측] 결론적으로 비혼이면서 자본소득이 적으면, 교육 기간에 관계없이 소득이 적을 確率이 높다.\ [예측] 반대로 기혼으로 교육 기간이 길면 자본 소득과 관계없이 소득이 높다.

In []:

모델을 이용해 예측하기

test set 데이터로 예측 검사

```
In [30]: ## test set 데이터를 예측 변수와 타겟 변수로 구분  
test_x = df_test.drop(columns = 'income') # 예측 변수 추출  
test_y = df_test['income'] # 타겟 변수 추출
```

```
In [31]: ## 예측값 구하기 : 예측 변수 사용  
df_test['pred'] = model.predict(test_x)  
df_test
```

C:\Users\ADMIN\anaconda3\lib\site-packages\sklearn\utils\validation.py:623: FutureWarning: is_sparse is deprecated and will be removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
... if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():

```
Out[31]:
```

	age	education_num	capital_gain	capital_loss	hours_per_week	workclass_?	workclass_Fed
11712	58		10	0	0	60	False
24768	39		10	0	0	40	False
26758	31		4	0	0	20	False
14295	23		9	0	0	40	False
3683	24		9	0	0	40	False
...
11985	24		13	0	0	30	False
48445	35		13	10520	0	45	False
19639	41		9	0	0	40	False
21606	29		4	0	0	30	False
3822	31		13	0	0	40	False

14653 rows × 109 columns

실제 데이터와 예측 결과 비교

예측 결과 pred 값과 테스트 입력 income 값이 같으면 예측이 맞은 것

[실습] pred 값과 income 값이 일치하는 비율을 구하시오.

pred 값과 income 값이 일치하는 비율을 구하기

```
In [32]: ## pred 값과 income 값이 일치하는 행 추출하기  
df_test.query('income == pred')
```

Out[32]:

	age	education_num	capital_gain	capital_loss	hours_per_week	workclass_?	workclass_Fed
11712	58	10	0	0	60	False	
24768	39	10	0	0	40	False	
26758	31	4	0	0	20	False	
14295	23	9	0	0	40	False	
3683	24	9	0	0	40	False	
...
39437	33	10	0	0	40	False	
11985	24	13	0	0	30	False	
48445	35	13	10520	0	45	False	
21606	29	4	0	0	30	False	
3822	31	13	0	0	40	False	

12366 rows × 109 columns

In [33]: `## ## pred 값과 income 값에 따른 행의 수 데이터 프레임 구성하기
df_pred = df_test.groupby(['income', 'pred']).count()['age'].to_frame()
df_pred`

Out[33]:

age		
income	pred	
high	high	1801
	low	1705
low	high	582
	low	10565

In [34]: `## 일치율 구하기
eq_rate = (df_pred.iloc[0] + df_pred.iloc[3]) / (df_pred.iloc[0] + df_pred.iloc[1])
eq_rate`

Out[34]: age ... 0.843923
dtype: float64

In []:

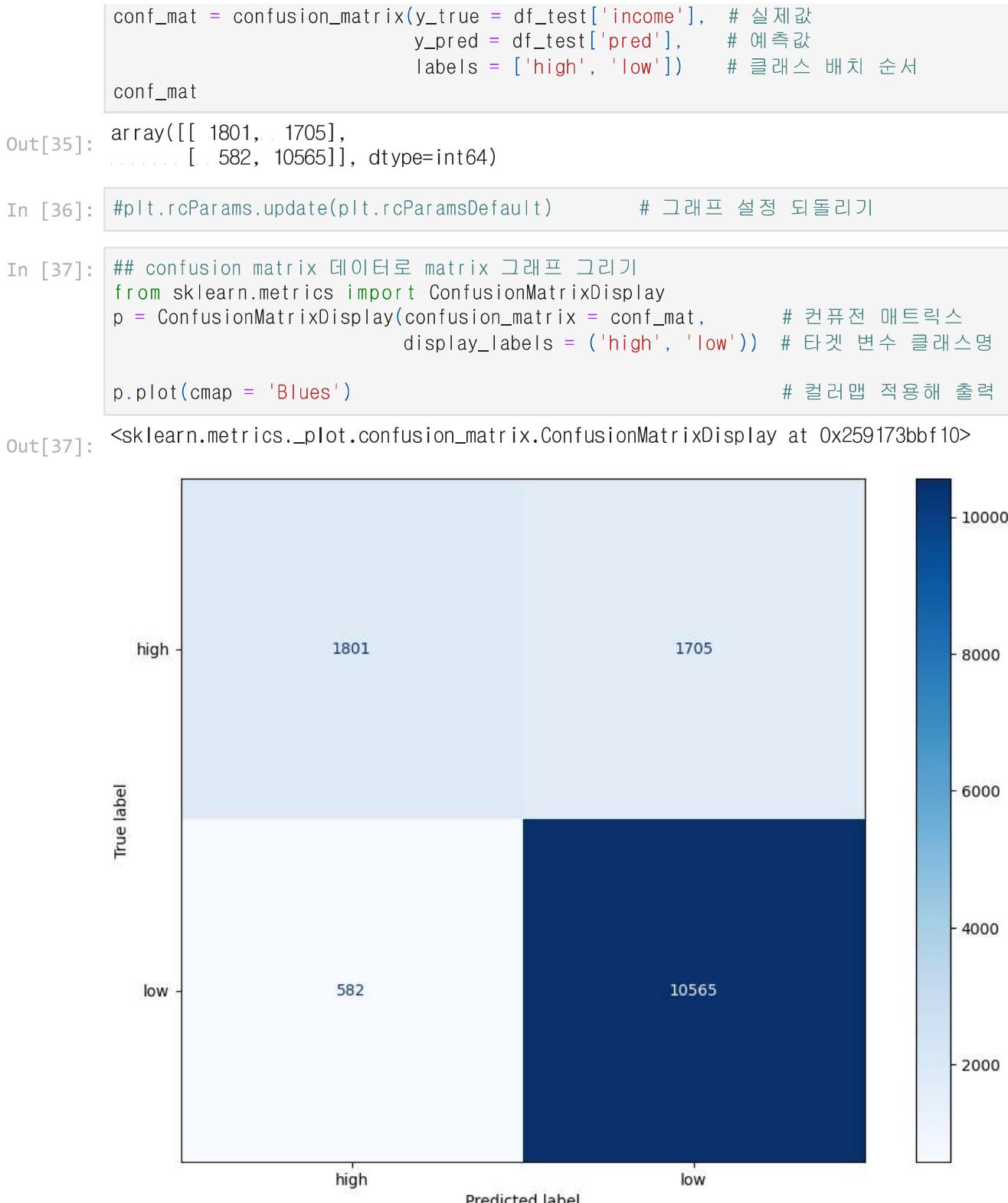
성능 평가하기

예측 변수 값으로 도출된 값과 실제 값(test set의 타깃 변수 값)과 비교 평가

confusion matrix 만들기

모델이 예측한 값 중에 맞은 경우와 틀린 경우의 빈도를 나타낸 표

In [35]: `## confusion matrix 데이터 만들기 : type은 numpy.ndarray
from sklearn.metrics import confusion_matrix`



confusion matrix 그래프 해석

y-축은 실제(Actual) 값의 예측 방향: 상위 행이 Positive, 하위 행이 Negative

x-축은 예측된(predicted) 값의 예측 방향: 왼쪽 열이 Positive, 오른 쪽 열이 Negative

모델의 목적이 고소득자(income=high)를 찾는 것이었으므로

high이면 Piositive, low이면 Negative

x, y-축 정답 일치 여부

- True Positive (TP) : Positive-Positive로 일치(**True**) > 4/4 분면
- True Negative (TN) : Negative-Negative로 일치(**True**) > 2/4 분면
- False Positive (FP) : Positive-Negative로 불일치(**False**) > 3/4 분면
- False Negative (FN) : Negative-Positive로 불일치(**False**) > 1/4 분면

In []:

성능 평가 지표 구하기

accuracy

정확도는 예측한대로 맞은 비율

- > 클래스를 예측(Positive: TP+FP 또는 Negative: TN+FN)해서, 맞춘(True: Tp + TN) 비율
- > 예측(TP + FP + TN + FN)해서 일치(TP + TN)함
- > 일반적인 성능 지표이므로 항상 살펴봐야 할 항목

$$\text{Accuracy} = (TP + TN) / ((TP + FP) + (TN + FN))$$

precision

정밀도는 관심(positive) 클래스를 예측(Positive: TP+FP)해서, 관심 클래스를 맞춘(True: Tp) 비율

- > Positive(TP + FP)로 예측해서 Positive(TP)로 일치함
- > 관심 클래스가 분명할 때 사용

$$\text{precision} = (TP) / (TP + FP)$$

recall

재현율은 클래스((positive + Negative)를 예측 해 맞춘(Positive: TP, Negative: FN) 것 중에 예측한데로 일치한(True: Tp) 비율

- > 일치한 것(TP + FN) 중에 Positive로 일치(TP)한 비율
- > 관심 클래스를 최대한 많이 찾아내야 할 때 사용

$$\text{recall} = (TP) / (TP + FN)$$

F1 score

F1 score는 정밀도(Precision)와 재현율(Recall)의 조화 평균

- > 이 두 지표를 하나로 결합하여 모델의 성능을 평가할 때 사용
- > precision과 recall 이 모두 중요할 때 사용

$$F1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

```
Out[28]: 0.8439227461953184
```

```
In [29]: ## precision 값 구하기  
metrics.precision_score(y_true = df_test['income'], # 실제값  
                        y_pred = df_test['pred'], # 예측값  
                        pos_label = 'high') # 관심 클래스
```

Out[29]: 0.7557700377675199

```
In [30]: ## recall 값 구하기  
metrics.recall_score(y_true = df_test['income'], # 실제값  
                      y_pred = df_test['pred'], # 예측값  
                      pos_label = 'high') # 관심 클래스
```

```
Out[30]: 0.5136908157444381
```

```
In [31]: ## F1 score 값 구하기  
metrics.f1_score(y_true = df_test['income'], # 실제값  
                  y_pred = df_test['pred'], # 예측값  
                  pos_label = 'high') # 관심 클래스
```

```
Out[31]: 0.6116488368143997
```

In []:

정리하기

```

## 2. 의사결정나무 모델 만들기

# 모델 설정하기
from sklearn import tree
clf = tree.DecisionTreeClassifier(random_state = 1234, # 난수 고정
                                   max_depth = 3)           # 나무 깊이

# 모델 만들기
train_x = df_train.drop(columns = 'income')          # 예측 변수 추출
train_y = df_train['income']                         # 타겟 변수 추출
model = clf.fit(X = train_x, y = train_y)           # 모델 만들기

# 모델 구조 살펴보기
import matplotlib.pyplot as plt
tree.plot_tree(model,
               feature_names = train_x.columns,      # 예측 변수명
               class_names = ['high', 'low'],        # 타겟 변수 클래스, 알파벳순
               proportion = True,                  # 비율 표기
               filled = True,                     # 색칠
               rounded = True,                   # 둥근 테두리
               impurity = False,                 # 불순도 표시
               label = 'root',                  # label 표시 위치
               fontsize = 12)                    # 글자 크기

## 3. 모델을 이용해 예측하기

# 예측하기
test_x = df_test.drop(columns = 'income')          # 예측 변수 추출
test_y = df_test['income']                         # 타겟 변수 추출
df_test['pred'] = model.predict(test_x)            # 예측값 구하기

## 4. 성능 평가하기

# confusion matrix 만들기
from sklearn import metrics
conf_mat = confusion_matrix(y_true = df_test['income'], # 실제값
                            y_pred = df_test['pred'],   # 예측값
                            labels = ['high', 'low']) # 클래스 배치 순서

# confusion matrix 시각화
from sklearn.metrics import ConfusionMatrixDisplay
p = ConfusionMatrixDisplay(confusion_matrix = conf_mat,    # 컨퓨전 매트릭스
                           display_labels = ('high', 'low')) # 타겟 변수 클래스명
p.plot(cmap = 'Blues')                                # 컬러맵 적용해 출력

# accuracy
metrics.accuracy_score(y_true = df_test['income'],     # 실제값
                       y_pred = df_test['pred'])       # 예측값

# precision
metrics.precision_score(y_true = df_test['income'],    # 실제값
                        y_pred = df_test['pred'],   # 예측값
                        pos_label = 'high')        # 관심 클래스

# recall
metrics.recall_score(y_true = df_test['income'],      # 실제값
                      y_pred = df_test['pred'],   # 예측값
                      pos_label = 'high')        # 관심 클래스

# F1 score
metrics.f1_score(y_true = df_test['income'],          # 실제값
                  y_pred = df_test['pred'],     # 예측값
                  pos_label = 'high')

```

```
y_pred = df_test['pred'],           # 예측값  
pos_label = 'high')                # 관심 클래스
```