원핫 인코딩 실습

학습 목표

```
01 원핫 인코딩 실습
02 adult.data 셋을 활용한 onehot encoding 실습
03 'hello world'를 원핫인코딩하기
```

01 원핫 인코딩 실습

- 데이터 셋을 불러와 원핫 인코딩 실습
- hello world 원핫 인코딩 실습

In [1]:

```
import mglearn
import pandas as pd
import os
```

In [2]:

```
demo_df = pd.DataFrame({"범주형_feature":['양말', '여우', '양말', '상자']})
display(demo_df)
```

	범주형_feature
0	양말
1	여우
2	양말
3	상자

In [3]:

```
onehot = pd.get_dummies(demo_df)
onehot
```

Out[3]:

	범주형_feature_상자	범주형_feature_양말	범주형_feature_여우
0	0	1	0
1	0	0	1
2	0	1	0
3	1	0	0

In [4]:

```
df = pd.concat([demo_df, onehot], axis=1)
df
```

Out[4]:

	범주형_feature	범주형_feature_상자	범주형_feature_양말	범주형_feature_여우
0	양말	0	1	0
1	여우	0	0	1
2	양말	0	1	0
3	상자	1	0	0

02. adult.data 셋을 활용한 onehot encoding 실습

In [5]:

```
path = os.path.join(mglearn.datasets.DATA_PATH, 'adult.data')
print(path)
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩mglearn₩data₩adult.data
```

In [6]:

```
data = pd.read_csv(path,
                  header=None,
                  index_col=False,
     names=['age', 'workclass', 'fnlwgt', 'education',
            'education-num', 'marital-status', 'occupation', 'relationship',
            'race', 'gender', 'capital-gain', 'capital-loss',
            'hours-per-week', 'native-country', 'income'])
```

In [7]:

```
data.columns
```

Out[7]:

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'gender',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'income'],
      dtype='object')
```

일부 변수 선택 후, 진행

In [11]:

```
sel = ['age', 'workclass','education','gender','hours-per-week',
       'occupation','income']
data = data[sel]
data.head()
```

Out[11]:

	age	workclass	education	gender	hours-per-week	occupation	income
0	39	State-gov	Bachelors	Male	40	Adm-clerical	<=50K
1	50	Self-emp-not-inc	Bachelors	Male	13	Exec-managerial	<=50K
2	38	Private	HS-grad	Male	40	Handlers-cleaners	<=50K
3	53	Private	11th	Male	40	Handlers-cleaners	<=50K
4	28	Private	Bachelors	Female	40	Prof-specialty	<=50K

의미 있는 범주형 데이터 있는지 확인

In [12]:

```
print(data.gender.value_counts())
```

```
 Male      21790
 Female    10771
Name: gender, dtype: int64
```

pandas에서 get_dummies 함수를 이용하여 인코딩

In [13]:

```
print("원본 특성 :\n", list(data.columns), "\n")
data_dummies = pd.get_dummies(data)
print("get_dummies 후 특성 : \n", list(data_dummies.columns))
```

```
원본 특성 :
 ['age', 'workclass', 'education', 'gender', 'hours-per-week', 'occupation', 'inco
me']

get_dummies 후 특성 :
 ['age', 'hours-per-week', 'workclass_ ?', 'workclass_ Federal-gov', 'workclass_ L
ocal-gov', 'workclass_ Never-worked', 'workclass_ Private', 'workclass_ Self-emp-i
nc', 'workclass_ Self-emp-not-inc', 'workclass_ State-gov', 'workclass_ Without-pa
y', 'education_ 10th', 'education_ 11th', 'education_ 12th', 'education_ 1st-4th',
'education_ 5th-6th', 'education_ 7th-8th', 'education_ 9th', 'education_ Assoc-ac
dm', 'education_ Assoc-voc', 'education_ Bachelors', 'education_ Doctorate', 'educ
ation_ HS-grad', 'education_ Masters', 'education_ Preschool', 'education_ Prof-sc
hool', 'education_ Some-college', 'gender_ Female', 'gender_ Male', 'occupation_
?', 'occupation_ Adm-clerical', 'occupation_ Armed-Forces', 'occupation_ Craft-rep
air', 'occupation_ Exec-managerial', 'occupation_ Farming-fishing', 'occupation_ H
andlers-cleaners', 'occupation_ Machine-op-inspct', 'occupation_ Other-service',
'occupation_ Priv-house-serv', 'occupation_ Prof-specialty', 'occupation_ Protecti
ve-serv', 'occupation_ Sales', 'occupation_ Tech-support', 'occupation_ Transport-
moving', 'income_ <=50K', 'income_ >50K']
```

- age와 hours-per-week는 그대로이지만 범주형 특성은 새로운 특성으로 확장

특성을 포함한 열 'age'~'occupation_ Transport-moving' 모두 추출

In [14]:

```
features = data_dummies.loc[:, "age":"occupation_ Transport-moving"]
X = features.values
y = data_dummies['income_ >50K'].values
```

In [15]:

```
print("X.shape : {}, y.shape : {}".format(X.shape, y.shape))
```

```
X.shape : (32561, 44), y.shape : (32561,)
```

실습 1

- 로지스틱 모델을 만들어보기
  - (1) 데이터를 나누어준다.
  - (2) 모델을 만든다.
  - (3) 모델을 학습한다.(학습 데이터를 이용해서)
  - (4) score를 확인(테스트 데이터를 이용해서)

로지스틱 모델 사용해 보기

In [16]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
print("테스트 점수 {:.2f}".format(logreg.score(X_test, y_test)))
```

```
테스트 점수 0.81

C:₩ProgramData₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.py:433: F
utureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
```

숫자로 표현된 범주형 특성을 원핫 인코딩하기

In [17]:

```
demo_df = pd.DataFrame({"숫자_feature":[0,1,2,1],
                        "범주형_feature":['양말', '여우', '양말', '상자']})
display(demo_df)
```

	숫자_feature	범주형_feature
0	0	양말
1	1	여우
2	2	양말
3	1	상자

In [18]:

```
display(pd.get_dummies(demo_df))
```

	숫자_feature	범주형_feature_상자	범주형_feature_양말	범주형_feature_여우
0	0	0	1	0
1	1	0	0	1
2	2	0	1	0
3	1	1	0	0

숫자도 원핫해보기

In [19]:

```
demo_df['숫자_feature']=demo_df['숫자_feature'].astype(str)
display(pd.get_dummies(demo_df, columns=['숫자_feature', '범주형_feature']))
```

	숫자_feature_0	숫자_feature_1	숫자_feature_2	범주형_feature_상자	범주형_feature_양말	범주형_feature_여우
0	1	0	0	0	1	0
1	0	1	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	1	0	0

03. 'hello world'를 원핫인코딩하기

In [20]:

```
from numpy import argmax
# define input string
data = 'hello world'
print(data)
```

```
hello world
```

In [21]:

```
# define universe of possible input values
alphabet = 'abcdefghijklmnopqrstuvwxyz '
# define a mapping of chars to integers
char_to_int = dict((c, i) for i, c in enumerate(alphabet))
int_to_char = dict((i, c) for i, c in enumerate(alphabet))

print("char_to_int : ", char_to_int)
print()
print("int_to_char : ", char_to_int)
```

```
char_to_int :  {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7,
'i': 8, 'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16,
'r': 17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25, '
': 26}

int_to_char :  {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7,
'i': 8, 'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16,
'r': 17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25, '
': 26}
```

In [22]:

```
# integer encode input data
integer_encoded = [char_to_int[char] for char in data]
print(integer_encoded)
```

```
[7, 4, 11, 11, 14, 26, 22, 14, 17, 11, 3]
```

In [23]:

```
# one hot encode
onehot_encoded = list()
for value in integer_encoded:
    letter = [0 for _ in range(len(alphabet))]
    letter[value] = 1
    onehot_encoded.append(letter)
print(onehot_encoded)
```

```
[[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0], [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0], [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0]]
```

In [24]:

```
# invert encoding
inverted = int_to_char[argmax(onehot_encoded[0])]
print(inverted)
```

```
h
```