

# 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 [2]:

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

In [3]:

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

	Product
0	양말
1	여우
2	양말
3	상자

In [4]:

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

Out[4]:

	Product_상자	Product_양말	Product_여우
0	0	1	0
1	0	0	1
2	0	1	0
3	1	0	0

In [5]:

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

Out[5]:

	Product	Product_상자	Product_양말	Product_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 [6]:

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

C:\Users\WWJ\Anaconda3\envs\Wtf2xWlib\site-packages\mglearn\data\adult.data

In [7]:

```
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 [8]:

```
data.columns
```

Out[8]:

```
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 [9]:



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

Out[9]:

	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 [10]:



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

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

## pandas에서 get\_dummies 함수를 이용하여 인코딩

In [11]:



```
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', 'income']
```

get\_dummies 후 특성 :

```
['age', 'hours-per-week', 'workclass_?', 'workclass_Federal-gov', 'workclass_Local-gov', 'workclass_Never-worked', 'workclass_Private', 'workclass_Self-emp-inc', 'workclass_Self-emp-not-inc', 'workclass_State-gov', 'workclass_Without-pay', 'education_10th', 'education_11th', 'education_12th', 'education_1st-4th', 'education_5th-6th', 'education_7th-8th', 'education_9th', 'education_Assoc-acdm', 'education_Assoc-voc', 'education_Bachelors', 'education_Doctorate', 'education_HS-grad', 'education_Masters', 'education_Preschool', 'education_Prof-school', 'education_Some-college', 'gender_Female', 'gender_Male', 'occupation_?', 'occupation_Adm-clerical', 'occupation_Armed-Forces', 'occupation_Craft-repair', 'occupation_Exec-managerial', 'occupation_Farming-fishing', 'occupation_Handlers-cleaners', 'occupation_Machine-op-inspct', 'occupation_Other-service', 'occupation_Priv-house-serv', 'occupation_Prof-specialty', 'occupation_Protective-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 [12]:



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

In [13]:



```
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:\Users\WWJ\Anaconda3\lib\site-packages\sklearn\linear\_model\\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html> (<https://scikit-learn.org/stable/modules/preprocessing.html>)

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression) ([https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression))

n\_iter\_i = \_check\_optimize\_result(