원핫 인코딩 실습

학습 목표

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

목차

01. 원핫 인코딩 실습

목차로 이동하기

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

In [6]:

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

In [7]:

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

	Product
0	양말
1	여우
2	양말
3	상자

In [8]:

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

Out[8]:

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

In [9]:

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

Out[9]:

	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

목차로 이동하기

In [11]:

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

C:₩Users₩totofriend₩anaconda3₩lib₩site-packages₩mglearn₩data₩adult.data

In [12]:

```
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'])
```

데이터 정보

age : 나이
workclass : 고용 형태
fnlwgt : 사람 대표성을 나타내는 가중치 (final weight의 약자)
education : 교육 수준 (최종 학력)
education_num : 교육 수준 수치
marital_status: 결혼 상태
occupation : 업종
relationship : 가족 관계
race : 인종
sex : 성별
capital_gain : 양도 소득
capital_loss : 양도 손실
hours_per_week : 주당 근무 시간
native_country : 국적
income : 연소득 (예측해야 하는 값, target variable) - 50K - $50,000

In [13]:

```
data.columns
```

Out[13]:

```
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 [14]:

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

Out[14]:

	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 [15]:

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

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

pandas에서 get_dummies 함수를 이용하여 인코딩

In [16]:

```
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', 'incom
e']

get_dummies 후 특성 :
 ['age', 'hours-per-week', 'workclass_ ?', 'workclass_ Federal-gov', 'workclass_ Loc
al-gov', 'workclass_ Never-worked', 'workclass_ Private', 'workclass_ Self-emp-inc',
'workclass_ Self-emp-not-inc', 'workclass_ State-gov', 'workclass_ Without-pay', 'ed
ucation_ 10th', 'education_ 11th', 'education_ 12th', 'education_ 1st-4th', 'educati
on_ 5th-6th', 'education_ 7th-8th', 'education_ 9th', 'education_ Assoc-acdm', 'educ
ation_ Assoc-voc', 'education_ Bachelors', 'education_ Doctorate', 'education_ HS-gr
ad', 'education_ Masters', 'education_ Preschool', 'education_ Prof-school', 'educat
ion_ 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', 'o
ccupation_ Machine-op-inspct', 'occupation_ Other-service', 'occupation_ Priv-house-
serv', 'occupation_ Prof-specialty', 'occupation_ Protective-serv', 'occupation_ Sal
es', 'occupation_ Tech-support', 'occupation_ Transport-moving', 'income_ <=50K', 'i
ncome_ >50K']
```

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

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

In [17]:

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

In [18]:

```
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 [21]:

```
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)
```

```
C:₩Users₩totofriend₩anaconda3₩lib₩site-packages₩sklearn₩linear_model₩_logistic.py:76
2: 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-lear
n.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 (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[21]:

```
LogisticRegression()
```

In [22]:

```
print("학습용 점수 {:.2f}".format(logreg.score(X_train, y_train)))
print("테스트 점수 {:.2f}".format(logreg.score(X_test, y_test)))
```

```
학습용 점수 0.81
테스트 점수 0.81
```

In [23]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

In [24]:

```
model = RandomForestClassifier().fit(X_train, y_train)

print("학습용 점수 {:.2f}".format(model.score(X_train, y_train)))
print("테스트 점수 {:.2f}".format(model.score(X_test, y_test)))
```

학습용 점수 0.94
테스트 점수 0.79

In [25]:

```
model = KNeighborsClassifier().fit(X_train, y_train)

print("학습용 점수 {:.2f}".format(model.score(X_train, y_train)))
print("테스트 점수 {:.2f}".format(model.score(X_test, y_test)))
```

학습용 점수 0.85
테스트 점수 0.78