강원지역혁신플랫폼

기계학습

Machine Learning

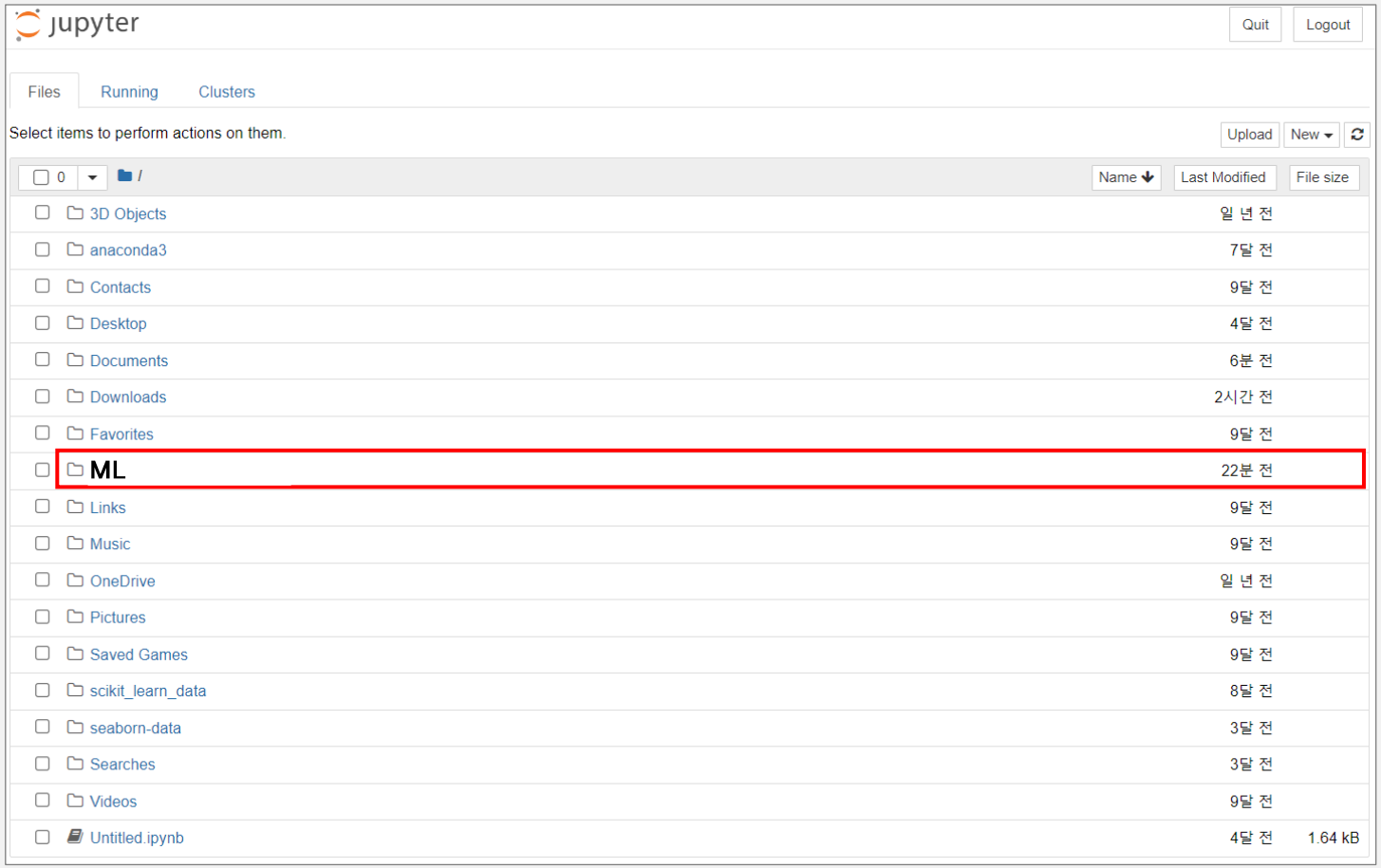
범주형 변환 변수

학습목표

- 범주형 변환 변수의 개념을 이해하고 구현할 수 있습니다.

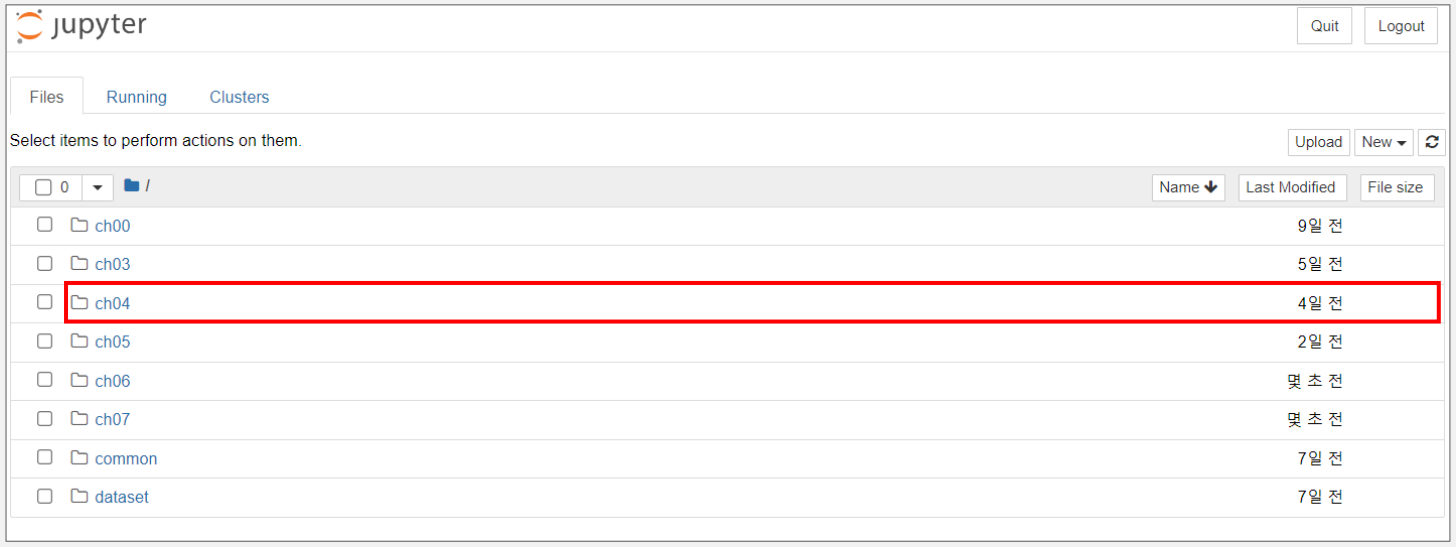
01 | ML 폴더

ML 폴더를 클릭하기



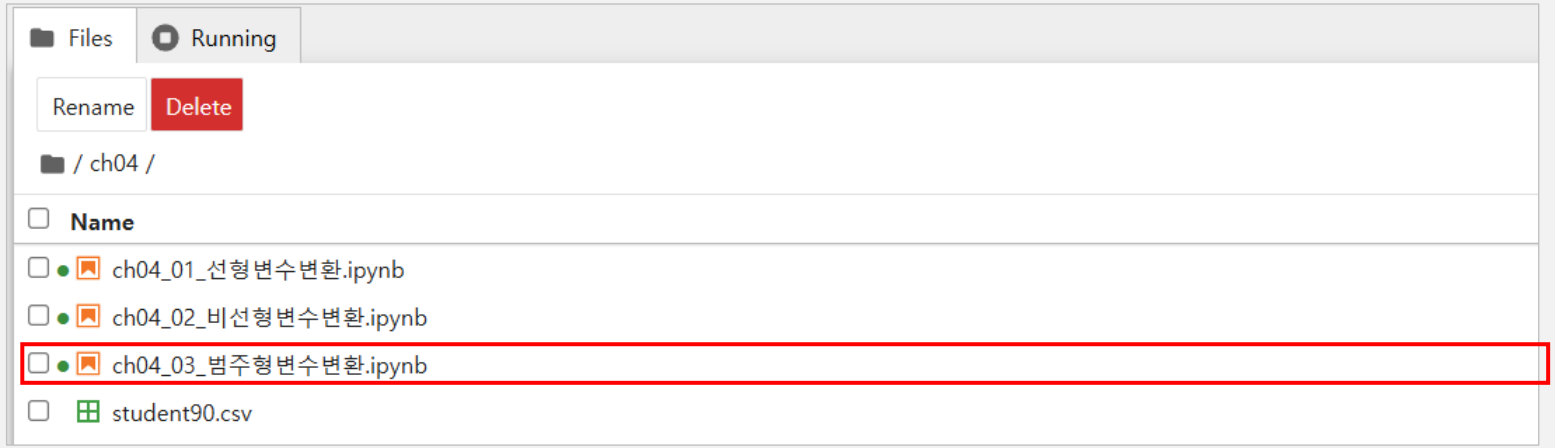
02 | ch04 폴더

ch04 폴더 클릭하기



03 | ch04_03_범주형변수변환.ipynb

✦ ch04_03_범주형변수변환.ipynb 파일 클릭하기



04 | 범주형 변수 변환

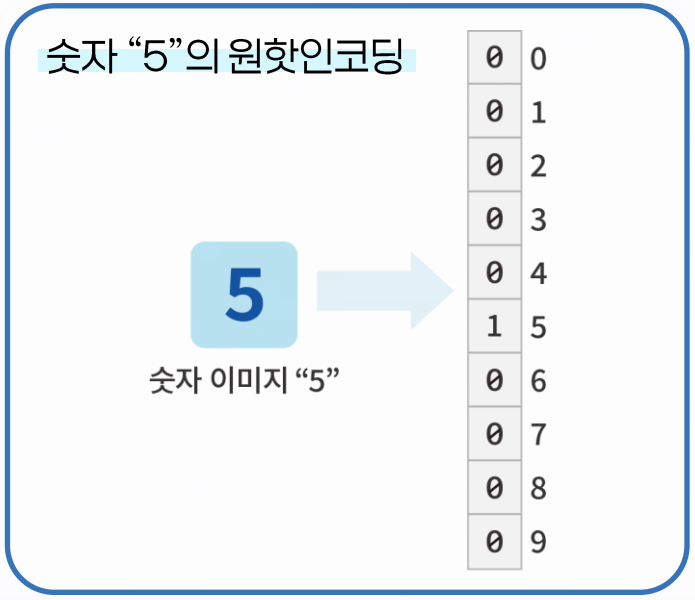
범주형 변수 변환(Categorical Feature Scaling)

- 특정 애플리케이션에 가장 적합한 데이터 표현을 찾는 것을 특성 공학(feature engineering)이라고 함
- 올바른 데이터 표현은 지도 학습 모델에서 적절한 매개변수를 선택하는 것보다 성능에 더 큰 영향을 미침
- 범주형 특성 혹은 이산형 특성(discrete feature)을 변환하는 방법은 다음과 같음
  - 원핫인코딩(One-hot-encoding), 더미코딩(dummy coding), 이펙트코딩(effect coding), 숫자로 표현된 범주형 특성, 레이블인코딩(label encoding), 특징 해싱(feature hashing), 빈도인코딩(frequency encoding)
    - 원핫인코딩이 가장 전통적인 방식임
    - 신경망의 경우에는 임베딩 계층을 변수별로 구성하는게 조금 번거롭지만 유효함

04 | 범주형 변수 변환: 원핫인코딩

원핫인코딩(One-hot-encoding) with get_dummies, OneHotEncoder, ColumnTransformer

- 원핫인코드는 One-out-of-N encoding, 가변수(dummy variable)라고도 부름
  - ✦ 범주형 변수를 0 또는 1 값을 가진 하나 이상의 새로운 특성으로 바꾼 것임
    - › 0과 1로 표현된 변수는 선형 이진 분류 공식에 적용할 수 있어서 개수에 상관없이 범주마다 하나의 특성으로 표현함
      - 아래의 그림은 숫자 이미지 "5"를 원핫인코딩으로 표현한 것임



04 | 범주형 변수 변환: 원핫인코딩

- 특징의 개수가 범주형 변수의 **레벨 개수**에 **따라 증가**하기 때문에
  정보가 적은 특징이 **대량 생성**돼서 **학습**에 필요한 **계산 시간**이나 **메모리**가 **급증**할 수 있음

  - 따라서 범주형 변수의 **레벨**이 **너무 많을 때**는 아래의 방법을 검토하여야 함

    - **다른 인코딩** 방법
    - 범주형 변수의 **레벨 개수**를 **줄임**
    - **빈도**가 **낮은 범주**를 **기타 범주**로 모아 정리하는 방법

04 | 범주형 변수 변환: 원핫인코딩

- 원핫인코딩은 다음의 함수로 구현할 수 있음
  - 판다스(pandas)의 get_dummies(데이터) 함수
    - 판다스의 get_dummies(drop_first=True) 함수에서 파라미터 drop_first=True를 설정해서 구현할 수 있음
      - 범주형 변수의 레벨이 n개일 때 해당 레벨 개수만큼 가변수를 만들면 다중공선성이 생기기 때문에 이를 방지하기 위해 n-1개의 가변수를 만드는 방법을 쓰는 것이 더미코딩임
  - 사이킷런(scikit learn)의 OneHotEncoder 혹은 ColumnTransformer 함수
    - OneHotEncoder 함수는 모든 특성을 범주형이라고 가정하여 수치형 열을 포함한 모든 열에 인코딩을 수행함
    - ColumnTransformer 함수는 범주형 변수에는 원핫인코딩, 수치형 변수에는 스케일링을 적용함

04 | 범주형 변수 변환: 원핫인코딩

- 다음은 대학생 90명의 키(cm)와 몸무게(kg) 데이터셋으로 원핫인코딩을 수행함
  - 아래의 표는 데이터셋의 데이터 구조임
    - 대학생 데이터셋의 관측치는 90개, 속성은 4개로 구성됨

no	sex	weight_kg	height_cm
1	M	98	198
2	F	77	170
3	M	70	170
4	M	90	198
5	F	71	170

04 | 범주형 변수 변환: 원핫인코딩

다음은 대학생 90명의 데이터 세트를 읽어오는 코드이다.

- 실행 결과 데이터의 형상이 (90, 4)인 것을 알 수 있음
  - 즉, 관측치의 데이터는 90개, 속성은 4개인 것을 알 수 있음

```
# 데이터 읽어오기
data = pd.read_csv(os.getcwd()+'/student90.csv')
print(data.shape)   # (90, 4)
data[:10]           # 10행 출력
```

(90, 4)

	no	sex	weight_kg	height_cm
0	1	M	98	198
1	2	M	77	170
2	3	M	70	170
3	4	M	90	198
4	5	M	71	170
5	6	M	70	165
6	7	M	73	193
7	8	M	59	142
8	9	M	68	137
9	10	M	86	155

04 | 범주형 변수 변환: 원핫인코딩-OneHotEncoder

다음은 대학생 90명 데이터셋으로 OneHotEncoder 함수를 이용해 원핫인코딩을 구현하는 코드이다.

✦ 실행 결과 모든 특성을 범주형이라고 가정하여 수치형 열을 포함한 모든 열에 인코딩을 수행한 것을 볼 수 있음

```
ohe = OneHotEncoder(sparse_output=False)      # 준비
data_ohe = ohe.fit_transform(data)            # 인코딩
feature_names = ohe.get_feature_names_out()   # feature names 추출
print(data_ohe.shape)                         # 형상 출력
print(data_ohe[:1,:])                         # 1행 출력
print(feature_names)                          # feature names 출력
```

```
(90, 154)
[[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. 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. 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. 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.]]
['no_1' 'no_2' 'no_3' 'no_4' 'no_5' 'no_6' 'no_7' 'no_8' 'no_9' 'no_10'
 'no_11' 'no_12' 'no_13' 'no_14' 'no_15' 'no_16' 'no_17' 'no_18' 'no_19'
 'no_20' 'no_21' 'no_22' 'no_23' 'no_24' 'no_25' 'no_26' 'no_27' 'no_28'
 'no_29' 'no_30' 'no_31' 'no_32' 'no_35' 'no_36' 'no_37' 'no_38' 'no_39'
 'no_40' 'no_41' 'no_42' 'no_43' 'no_44' 'no_45' 'no_46' 'no_47' 'no_48'
 'no_49' 'no_50' 'no_51' 'no_52' 'no_53' 'no_54' 'no_55' 'no_56' 'no_57'
 'no_58' 'no_59' 'no_60' 'no_61' 'no_62' 'no_63' 'no_64' 'no_65' 'no_66'
 'no_67' 'no_68' 'no_69' 'no_70' 'no_71' 'no_72' 'no_73' 'no_74' 'no_75'
 'no_76' 'no_77' 'no_78' 'no_79' 'no_80' 'no_81' 'no_82' 'no_83' 'no_84'
 'no_85' 'no_86' 'no_87' 'no_88' 'no_89' 'no_90' 'no_91' 'no_92' 'sex_f'
 'sex_m' 'weight_kg_43' 'weight_kg_46' 'weight_kg_49' 'weight_kg_50'
 'weight_kg_51' 'weight_kg_52' 'weight_kg_53' 'weight_kg_54'
 'weight_kg_55' 'weight_kg_56' 'weight_kg_57' 'weight_kg_59'
 'weight_kg_60' 'weight_kg_61' 'weight_kg_62' 'weight_kg_63'
 'weight_kg_64' 'weight_kg_66' 'weight_kg_67' 'weight_kg_68'
 'weight_kg_69' 'weight_kg_70' 'weight_kg_71' 'weight_kg_73'
 'weight_kg_74' 'weight_kg_75' 'weight_kg_77' 'weight_kg_79'
 'weight_kg_80' 'weight_kg_82' 'weight_kg_84' 'weight_kg_86'
 'weight_kg_88' 'weight_kg_89' 'weight_kg_90' 'weight_kg_98'
 'weight_kg_100' 'weight_kg_101' 'weight_kg_110' 'height_cm_132'
 'height_cm_135' 'height_cm_137' 'height_cm_140' 'height_cm_142'
 'height_cm_147' 'height_cm_150' 'height_cm_152' 'height_cm_155'
 'height_cm_156' 'height_cm_160' 'height_cm_161' 'height_cm_162'
 'height_cm_163' 'height_cm_165' 'height_cm_166' 'height_cm_168'
 'height_cm_170' 'height_cm_171' 'height_cm_180' 'height_cm_190'
 'height_cm_193' 'height_cm_198']
```

04 | 범주형 변수 변환: 원핫인코딩-ColumnTransformer

다음은 대학생 90명 데이터셋으로 ColumnTransformer 함수를 이용해 원핫인코딩을 구현하는 코드이다.

- 실행 결과 범주형 변수에는 원핫인코딩, 수치형 변수에는 스케일링을 적용하여 인코딩된 것을 볼 수 있음

```
# 준비
ct = ColumnTransformer([
                ('scaling', StandardScaler(), ['weight_kg',          'height_cm']),
                ('onehot', OneHotEncoder(sparse_output=False), ['sex'])
])
# 인코딩
data_ct = ct.fit_transform(data)
feature_names = ct.get_feature_names_out()  # feature names 추출
print(data_ct.shape)                        # 형상 출력
print(data_ct[:1,:])                        # 1행 출력
print(feature_names)                        # feature names 출력
```

```
(90, 4)
[[1.99325723 1.75590438 0.         1.        ]]
['scaling__weight_kg' 'scaling__height_cm' 'onehot__sex_f' 'onehot__sex_m']
```

04 | 범주형 변수 변환: 원핫인코딩-get_dummies( )

다음은 대학생 90명 데이터셋으로 get_dummies() 함수를 이용해 원핫인코딩을 구현하는 코드이다.

- 실행 결과 범주형 변수에는 원핫인코딩이 적용되어 인코딩된 것을 볼 수 있음

```
# 원핫인코딩
dummies = pd.get_dummies(data)
print(list(data.columns), data.shape)              # 원래 데이터
print(list(dummies.columns), dummies.shape)  # 원핫인코딩된 데이터
print(dummies[:5])                                        # 원핫인코딩된 데이터 5행 출력
```

```
['no', 'sex', 'weight_kg', 'height_cm'] (90, 4)
['no', 'weight_kg', 'height_cm', 'sex_f', 'sex_m'] (90, 5)
   no  weight_kg  height_cm  sex_f  sex_m
0   1         98        198      0      1
1   2         77        170      0      1
2   3         70        170      0      1
3   4         90        198      0      1
4   5         71        170      0      1
```

04 | 범주형 변수 변환: 더미코딩-get_dummies( )

다음은 대학생 90명 데이터셋으로 get_dummies() 함수를 이용해 더미코딩을 구현하는 코드이다.

- 범주형 변수의 레벨이 n개일 때 해당 레벨 개수만큼 가변수를 만들면 다중공선성이 생기기 때문에 이를 방지하기 위해 n-1개의 가변수를 만드는 방법을 쓰는 것이 더미코딩
  - 실행 결과 범주형 변수의 레벨이 2개이므로, 1개의 가변수가 생성된 것을 볼 수 있음

```
# 더미코딩
dummies2 = pd.get_dummies(data, drop_first=True)
print(list(dummies2.columns), dummies2.shape)  # 더미코딩된 데이터
print(dummies2[:5])                            # 더미코딩된 데이터 5행 출력
```

```
['no', 'weight_kg', 'height_cm', 'sex_m'] (90, 4)
   no  weight_kg  height_cm  sex_m
0   1         98        198      1
1   2         77        170      1
2   3         70        170      1
3   4         90        198      1
4   5         71        170      1
```

04 | 범주형 변수 변환: 오디널인코딩

오디널 인코딩(Ordinal encoding) with OrdinalEncoder

- 오디널(Ordinal)한 데이터에 특화된 인코딩기법임
  - 주의해야 될 점은 오디널(Ordinal)데이터를 함부로 숫자화할 수 없다는 점임
    - 실제로 대다수 데이터는 상하관계 혹은 피처(feature)간의 차이가 샘플마다 다른 경우가 대다수임
      - 예를 들어 A에게 상과 중의 차이가 B가 느끼는 상과 중의 차이와 다를 수 있는 것임
      - 오디널 인코딩을 실제로 사용하는 일은 굉장히 드묾

04 | 범주형 변수 변환: 레이블인코딩

레이블 인코딩(Label encoding) with LabelEncoder

- 각 레벨을 단순히 정수로 변환하는 방법임
  - 예를 들어 5개의 레벨이 있는 범주형 변수는 각 레벨이 0~4까지의 수치로 바꿈
    - 사전 순으로 나열했을 때의 인덱스 수치는 대부분 본질적인 의미가 없음
    - 결정 트리 모델에 기반을 둔 방법이 아닐 경우 레이블 인코딩으로 변환한 특징을 학습에 직접 이용하는 건 그다지 적절하지 않음
      - 결정트리에서는 범주형 변수의 특정 레벨만 목적 변수에 영향을 줄 때도 분기를 반복함으로써 예측값에 반영할 수 있으므로 학습에 활용할 수 있음

변수
A
B
C
D

레이블인코딩
0
1
2
3

4개의 레벨의 범주형 변수는 각 레벨이 0~3까지의 수치로 바뀜

04 | 범주형 변수 변환: 레이블인코딩

다음은 레이블인코딩에 이용할 데이터 세트를 생성하는 코드이다.

- 아래와 같이 2개의 속성(grade, class)과 4개 행 데이터가 생성된 것을 볼 수 있음

```
std_card = pd.DataFrame({
                'grade': ['A', 'B', 'C', 'D'],
                'class': ['H', 'M', 'L', 'H']
})
std_card
```

	grade	Class
0	A	H
1	B	M
2	C	L
3	D	H

04 | 범주형 변수 변환: 레이블인코딩

다음은 'grade' 변수에 레이블인코딩을 적용하는 코드이다.

- 실행 결과 'A' → 0, 'B' → 1, 'C' → 2, 'D' → 3 으로 레이블인코딩된 것을 알 수 있음

```
le = LabelEncoder()
grade_le = le.fit_transform(std_card['grade']) # 레이블인코딩
print(grade_le)
print(le.classes_)                              # 변환된 label classes 확인
print(le.inverse_transform(grade_le))           # 역변환
```

```
[0 1 2 3]
['A' 'B' 'C' 'D']
['A' 'B' 'C' 'D']
```

04 | 범주형 변수 변환: 레이블인코딩

- 다음은 'class' 변수에 레이블인코딩을 적용하는 코드이다.
  - 실행 결과 'H' → 0, 'L' → 1, 'M' → 2 로 레이블인코딩된 것을 알 수 있음

```
class_le = le.fit_transform(std_card['class']) # 레이블인코딩
print(class_le)
print(le.classes_)                    # 변환된 label classes 확인
print(le.inverse_transform(class_le))  # 역변환
```

```
[0 2 1 0]
['H' 'L' 'M']
['H' 'M' 'L' 'H']
```

05 | 수치형 변수 → 범주형 변수 변환: 구간분할

구간분할(Binning) with cut, KBinsDiscretizer

- 구간분할 혹은 이산화는 수치형 변수를 구간별로 나누어 범주형 변수로 변환하는 방법임
  - 구간분할을 하면 순서가 있는 범주형 변수가 되므로 순서를 그대로 수치화할 수도 있음
    - 또한, 범주형 변수로서 원핫인코딩을 적용할 수도 있음
    - 구간의 범주마다 다른 변수값을 집계할 수 있는 범주형 변수로 사용할 수 있음
    - 구간 분할하는 방법은 다음과 같음
      - 같은 간격으로 분할하는 방법
      - 분위점을 이용하여 분할하는 방법
      - 구간 구분을 지정하여 분할하는 방법 등

05 | 수치형 변수 → 범주형 변수 변환: 구간분할

- 구간분할은 다음의 함수로 구현할 수 있음
  - 판다스(pandas)의 cut 함수
    - 판다스의 cut 함수로 지정한 bins 수만큼 균일한 범위로 데이터를 분할 하거나 지정한 범위대로 데이터를 분할 할 수 있음
  - 사이킷런(scikit learn)의 KBinsDiscretizer 함수
    - KBinsDiscretizer 함수를 이용하면, 지정한 bins 수대로 데이터를 분할한 후 각 구간 별로 원핫인코딩을 적용할 수 있음
      - 구간마다 하나의 새로운 특성이 생기므로 희소 행렬이 만들어짐

05 | 수치형 변수 → 범주형 변수 변환: 구간분할

다음은 대학생 90명의 몸무게로 5개 구간으로 나누어 범주형 변수로 변환하는 코드이다.

- 실행 결과 아래와 같이 5개 구간으로 나눈 것을 알 수 있음

```
binned_weight = cut(data['weight_kg'], 5)  # bin수 지정
binned_weight.value_counts()
```

```
(56.4, 69.8]       32
(69.8, 83.2]       29
(42.933, 56.4]     14
(83.2, 96.6]        9
(96.6, 110.0]       6
Name: weight_kg, dtype: int64
```

05 | 수치형 변수 → 범주형 변수 변환: 구간분할

다음은 대학생 90명의 몸무게로 5개 구간으로 나누어 범주형 변수로 변환하는 코드이다.

- 여기서는 5개 구간으로 나누고 각 구간을 “A”, “B”, “C”, “D”, “E” 범주형 코드로 변환함
  - 실행 결과 아래와 같이 각 구간이 “A”, “B”, “C”, “D”, “E” 범주형 코드로 변환된 것을 볼 수 있음

```
binned_weight = cut(data['weight_kg'], 5, labels=["A","B","C","D","E"])  # bin수 지정
binned_weight.value_counts()
```

```
B    32
C    29
A    14
D     9
E     6
Name: weight_kg, dtype: int64
```

05 | 수치형 변수 → 범주형 변수 변환: 구간분할

- 다음은 대학생 90명의 몸무게로 bin의 범위를 지정하여 범주형 변수로 변환하는 코드이다.
  - 여기서는 bin의 범위를 지정하여 각 구간을 1, 2, 3, 4, 5, 6, 7 범주형 코드로 변환함
    - 실행 결과 아래와 같이 각 구간이 1, 2, 3, 4, 5, 6, 7 범주형 코드로 변환된 것을 볼 수 있음

```
bin_edges = [0, 50, 60, 70, 80, 90, 100, float('inf')]
# 0초과~50이하, 50초과~60이하, ..., 100초과

binned_weight2 = cut(data['weight_kg'], bin_edges, labels=[1,2,3,4,5,6,7])
binned_weight2.value_counts()
```

```
3    32
2    19
4    17
5    12
1     4
6     3
7     3
Name: weight_kg, dtype: int64
```

05 | 수치형 변수 → 범주형 변수 변환: 구간분할

- 다음은 대학생 90명의 몸무게로 5개 구간으로 나누어 범주형 변수로 변환하는 코드이다.
  - 여기서는 KBinsDiscretizer 함수로 5개 구간으로 분할 후 각 구간 별로 원핫인코딩을 적용함
    - 실행 결과 아래와 같이 각 구간이 원핫인코딩으로 변환된 것을 볼 수 있음

```
kb = KBinsDiscretizer(n_bins=5, strategy='uniform')
kb_weight = kb.fit_transform(np.array(data['weight_kg']).reshape(-1,1))
kb_weight = kb_weight.toarray()
print(kb_weight.shape)     # 형상 출력
print(kb_weight[:5,:])     # 5행 출력
```

```
(90, 5)
[[0. 0. 0. 0. 1.]
 [0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 1. 0. 0.]]
```