Data Preprocessing

데이터 전처리

- 데이터의 질과 유용한 정보의 양은 머신러닝 알고리즘의 학습 효율성을 결정하는 중요한 요소

- 데이터를 학습 알고리즘에 입력하기 전에 확인 및 전처리 필요

- 기본 전처리 요소
  - » 결측 데이터 처리 → 제거 또는 대체
  - » 범주형 데이터를 알고리즘을 위한 형태로 변형
  - » 연관성 있는 피처 선택

결측 데이터

- 입력되지 않거나 잘못 입력된 데이터

- 파이썬에서 결측 데이터는 빈공간, NaN, Null 등으로 표시

결측 데이터 처리

- 패키지 준비

```
import pandas as pd
from io import StringIO # 문자열을 파일처럼 사용할 수 있도록 지원
```

- 데이터 준비

```
csv_data = """A,B,C,D
1.0,2.0,3.0,4.0
5.0,6.0,,8.0
9.0,10.0,11.0,
"""
df = pd.read_csv(StringIO(csv_data))
df
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	9.0	10.0	11.0	NaN

결측 데이터 처리

- 결측 여부 확인

```
# df.isnull()
# df.isna()
# df.notnull()
# df.notna()
# df.isnull().sum(axis=1) # NA 데이터 포함 여부 (NA → 0)
```

- 결측 데이터 제거

```
# NaN 제거
# df.dropna() # NaN이 포함된 모든 행 제거
# df.dropna(axis=1) # NaN이 포함된 모든 열 제거
# df.dropna(how="all") # 모든 데이터가 NaN인 행 제거 (any or all)
# df.dropna(thresh=4) # NaN이 아닌 데이터가 4개 미만인 행 제거
# df.dropna(subset=['C']) # C 컬럼에 NaN이 포함된 행 제거
```

결측 데이터 처리

- 결측 데이터를 다른 데이터로 대체
  - » 다양한 보정방법을 사용해서 데이터 세트 내의 다른 훈련 샘플들로부터 결측 값 추정

```
from sklearn.impute import SimpleImputer
import numpy as np

simr = SimpleImputer(missing_values=np.nan, strategy="constant",
fill_value=100) # strategy : mean, most_frequent, median
simr = simr.fit(df)
imputed_data = simr.transform(df.values)
imputed_data
```

```
array([[  1.,   2.,   3.,   4.],
       [  5.,   6., 100.,   8.],
       [  9.,  10.,  11., 100.]])
```

범주형 데이터

- 범주형 데이터
  - » 유한한 수의 범주 또는 고유 그룹

- 데이터 종류

종류		설명
양적 데이터	연속형 데이터	▪ 두 값 사이에 무한한 개수의 값이 있는 숫자 ▪ 숫자 또는 날짜/시간 ▪ 예)부품 길이, 대금이 결제된 날짜 및 시간
	이산형 데이터	▪ 두 값 사이에 셀 수 있는 수의 값이 있는 숫자 ▪ 항상 숫자 ▪ 예) 고객 불만 수, 결점 또는 결함의 수
범주형 데이터	순위형 데이터	▪ 정렬하거나 순위화 할 수 있는 데이터 ▪ 예) T-Shirt Size : XL > L > M
	명목형 데이터	▪ 정렬하거나 순위화 할 수 없는 데이터 ▪ 예) T-Shirt Color : Red, Green, Blue

범주형 데이터 처리

- 데이터 준비

```
df2 = pd.DataFrame([
    ['green', 'M', 10.1, 'class1'],
    ['red', 'L', 13.5, 'class2'],
    ['blue', 'XL', 15.3, 'class1'],
])
df2.columns = ['color', 'size', 'price', 'classlabel']
df2
```

	color	size	price	classlabel
0	green	M	10.1	class1
1	red	L	13.5	class2
2	blue	XL	15.3	class1

범주형 데이터 전처리

- 범주 문자열을 이산형 숫자로 변경

```
# np.unique(df2['classlabel'])
# class_mapping = { 'class1' : 0, 'class2' : 1, 'class3' : 2}
class_mapping = {
    label : idx for idx, label
        in enumerate(np.unique(df2['classlabel']))
}

df2['classlabel'] = df2['classlabel'].map(class_mapping)
df2
```

	color	size	price	classlabel
0	green	M	10.1	0
1	red	L	13.5	1
2	blue	XL	15.3	0

범주형 데이터 전처리

- 이산형 숫자를 범주 문자열로 변경

```
inv_class_mapping = { v : k for k, v in class_mapping.items() }
# print(inv_class_mapping)
df2['classlabel'] = df2['classlabel'].map(inv_class_mapping)
df2
```

	color	size	price	classlabel
0	green	M	10.1	class1
1	red	L	13.5	class2
2	blue	XL	15.3	class1

범주형 데이터 전처리

- 범주 문자열을 이산형 숫자로 변경

```
from sklearn.preprocessing import LabelEncoder

class_le = LabelEncoder()
y = class_le.fit_transform(df2['classlabel'].values)
y
```

```
array([0, 1, 0], dtype=int64)
```

```
df2['classlabel'] = class_le.fit_transform(df2['classlabel'].values)
df2
```

	color	size	price	classlabel
0	green	M	10.1	0
1	red	L	13.5	1
2	blue	XL	15.3	0

범주형 데이터 전처리

- 이산형 숫자를 범주 문자열로 변경

```
df2['classlabel'] = class_le.inverse_transform(df2['classlabel'].values)
df2
```

	color	size	price	classlabel
0	green	M	10.1	class1
1	red	L	13.5	class2
2	blue	XL	15.3	class1

범주형 데이터 전처리

- 원 핫 인코딩

```
from sklearn.preprocessing import OneHotEncoder

X = df2[['color', 'price']].values
# print(X)
color_le = LabelEncoder()
X[:, 0] = color_le.fit_transform(X[:, 0])
print(X)

ohe = OneHotEncoder(categorical_features=[0]) # 0번째 컬럼 Encoding
encoded = ohe.fit_transform(X).toarray()
print(encoded)
```

```
[[1 10.1]
 [2 13.5]
 [0 15.3]]
[[ 0.   1.   0.  10.1]
 [ 0.   0.   1.  13.5]
 [ 1.   0.   0.  15.3]]
```

범주형 데이터 전처리

- 원 핫 인코딩

```
pd.get_dummies(df2[['price', 'color']])
```

	price	color_blue	color_green	color_red
0	10.1	0	1	0
1	13.5	0	0	1
2	15.3	1	0	0

테스트 데이터 / 훈련 데이터 분할

- 데이터 준비

```
df_wine = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-
databases/wine/wine.data',
header=None)

df_wine.columns = ['Class label', 'Alcohol', 'Malic acid', 'Ash',
'Alcalinity of ash', 'Magnesium', 'Total phenols',
'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins',
'Color intensity', 'Hue', 'OD280/OD315 of diluted wines', 'Proline']

print('Class labels', np.unique(df_wine['Class label']))
df_wine.head()
```

	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflava phe
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	

테스트 데이터 / 훈련 데이터 분할

- 데이터 분할

```
from sklearn.model_selection import train_test_split

X, y = df_wine.iloc[:, 1:].values, df_wine.iloc[:, 0].values
X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=0.3, random_state=0)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((124, 13), (54, 13), (124,), (54,))
```

데이터 스케일링

- 0 ~ 1 사이의 데이터로 변경

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

```
from sklearn.preprocessing import MinMaxScaler

X = df_wine.iloc[:, 1:].values # 1열을 제외한 나머지 모든 데이터
mms = MinMaxScaler()
X2 = mms.fit_transform(X)
print(X[:5, :])
print("*" * 50)
print(X2[:5, :])
```

결과 생략

데이터 스케일링

- 평균과 표준편차를 기준으로 변경
  - » 평균 0, 표준편차 1인 정규분포로 변경
  - » 변경후에도 이상치에 대한 유용한 정보 보존

$$z = \frac{x_i - \mu}{\sigma}$$

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
X2 = ss.fit_transform(X)
print(X[:5, :])
print("*" * 50)
print(X2[:5, :])
```

결과 생략