2. 데이터 전처리

데이터 전처리

데이터 전처리

- 표준화
- 정규화
- 인코딩
- 결측값 처리
- 단순 데이터 분리
- sklearn.preprocessing 패키지는 원시 데이터를 모델을 만들기 적합한 데이터로 변환하는 것을 도와준다.

표준화

표준화

- 데이터를 정해진 구간 사이의 값으로 표준화(Standardization) 한다.
- klearn.preprocessing 모듈의 함수 사용
  - scale(x) : 표준 정규분포를 사용해 표준화
  - robust_scale(x) : 중위수(median)와 사분위범위(interquartile range)를 사용하여 표준화
  - minmax_sacle(x) :최댓값과 최솟값을 사용하여 표준화. 0~1
  - maxabs_scale(x) :최대 절댓값을 사용하여 표준화. -1~1

표준화 실습

- scale()
  - 평균을 0, 표준편차를 1이 되도록 표준화 한다.

```
from sklearn.preprocessing import scale
x_scaled = scale(x)
x_scaled[:5,:]

x_scaled.mean(axis=0) # 표준화 한 데이터들의 평균

for scaled_mean in x_scaled.mean(axis=0):
    print("{:10.9f}".format(scaled_mean)) #사실상 0에 가까운 수
```

표준화

- **robust_scale()**
  - 중위수(median)와 사분위범위(interquartile range)를 사용하여 표준화 한다.
  - Ex)

```
from sklearn.preprocessing import robust_scale
iris_robust_scaled = robust_scale(x)
iris_robust_scaled[:5,:]
```

- **minmax_scale()**
  - 최솟값을 0, 최댓값을 1에 매핑시켜 표준화 한다.
  - Ex)

```
from sklearn.preprocessing import minmax_scale
iris_minmax_scaled = minmax_scale(x)
iris_minmax_scaled[:5,:]
```

- **maxabs_scale()**
  - 절댓값이 가장 큰 값을 1로 정하면서 0부터 1사이의 값에 매핑시키지만 음수는 부호를 그대로 유지하는 것이 minmax_scale() 함수와 다른 점이다.
  - Ex)

```
from sklearn.preprocessing import maxabs_scale
iris_maxabs_scaled = maxabs_scale(x)
iris_maxabs_scaled[:5,:]
```

표준화

- **스케일과 스케일 백**
  - **표준화 메서드들을 이용해서 표준화를 할 수도 있지만 StandardScaler, MinMaxScaler, MaxAbsScaler 클래스를 이용해서 표준화를 할 수 있다.**
  - **이들 클래스를 이용하면 표준화 후 표준화한 값을 원래의 값 범위로 되돌릴 수 있다**
  - **이들 클래스의 표준화 방법은 scale(), minmax_scale(), maxabs_scale() 함수와 같다.**

메서드	설명
fit(X[, y])	스케일링에 사용될 평균 및 표준 편차를 계산합니다.
fit_transform(X[, y])	평균과 표준편차를 계산하고 표준화를 수행해서 데이터를 변환합니다.
get_params([deep])	파라미터를 가져옵니다.
inverse_transform(X[, copy])	데이터를 원래 표현으로 스케일 백 합니다.
partial_fit(X[, y])	스케일링을 위해 X에 대한 평균 및 표준편차를 계산합니다.(RobustScaler 클래스에는 이 메서드가 없습니다.)
set_params(**params)	매개변수를 설정합니다.
transform(X[, y, copy])	표준화를 수행해서 데이터를 변환합니다.

표준화

- StandardScaler
  - StandardScaler 객체를 이용해 표준화 한다.
  - Ex)

```
import seaborn as sns
iris = sns.load_dataset("iris")
iris.iloc[:5,:-1]

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
iris_scaled = sc.fit_transform(iris.iloc[:,:-1])
iris_scaled[:5,:]

iris_origin = sc.inverse_transform(iris_scaled)
iris_origin[:5,:]
```

정규화(Normalization)

□ 정규화(Normalization)

- 정의
  - 정규화(Normalization)는 개별 샘플을 단위 표준으로 조정 하는 절차이다.
  - 정규화는 과적합을 예방하고 일반화 성능을 높이는 데 도움을 주며 L1 정규화, L2 정규화 등이 있다.
- L1 정규화
  - 라소(Lasso) 정규화라고 한다.
  - 1차 항인 절댓값 항을 제약하기 때문에 L1 정규화라고 부른다
  - L1 정규화된 가중치 행렬은 중요한 특징(feature)에 연결된 가중치 값을 제외하고 작은 가중치 값을 가지도록 제약하는 경향이 있다.
  - Lasso 정규화 강도를 점차 늘리면 가중치 값이 분류 문제 해결에 덜 중요한 feature 순서대로 0이 된다.
  - 각 feature에 대응하는 가중치 값이 0으로 수렴하는 속도를 보여주는 이 플롯을 Lasso path라고 하는데 선형 분류기나 회귀분석에서는 Lasso path를 사용해서 feature 중요도를 판단하기도 한다.
- L2 정규화
  - 릿지(Ridge) 정규화라고 한다.
  - 가중치 행렬 W의 모든 성분의 제곱 합을 제약하는 정규화 방법이다.
  - 제곱값(2차항)을 제약하기 때문에 L2 정규화라고 부른다.
  - 이 정규화는 튀는 값을 좋아하지 않는 Ridge의 특성상 덜 공격적인 가중치 값을 얻을 수 있다.

정규화(Normalization)

- **sklearn.preprocessing.normalize()**
  - **sklearn.preprocessing.normalize() 함수는 정규화를 제공한다.**

메서드	설명
fit(X[, y])	아무것도 하지 않고 변경되지 않은 값을 반환합니다.
fit_transform(X[, y])	정규화를 수행해서 데이터를 변환합니다.
get_params([deep])	파라미터를 가져옵니다.
set_params(**params)	매개변수를 설정합니다.
transform(X[, y, copy])	X를 정규화 합니다.

  - **Ex)**

```
from sklearn.preprocessing import normalize
x = [[ 1., -1., 2.],
[ 2., 0., 0.],
[ 0., 1., -1.]]
x_normalized_l1 = normalize(x, norm='l1')
x_normalized_l1

x_normalized_l2 = normalize(x, norm='l2')
x_normalized_l2
```

인코딩

인코딩

레이블 인코딩(Label Encoding)

- 실제 값에 상관없이 0~K-1까지의 정수로 변환하는 것이다.
- 지역 또는 성별 등 문자로 되어있는 데이터를 숫자로 변환해야 하며 preprocessing 모듈의 LabelEncoder 클래스(sklearn.preprocessing.LabelEncoder())를 이용한다.

메서드	설명
fit(y)	레이블 인코더 모델을 생성합니다.
fit_transform(y)	레이블 인코더 모델을 생성하고 인코드된 레이블을 반환합니다.
get_params([deep])	매개변수를 반환합니다.
inverse_transform(y)	레이블을 원래 인코딩으로 다시 변환합니다.
set_params(**params)	매개변수를 설정합니다.
transform(y)	라벨을 표준화 된 인코딩으로 변환합니다.

- Ex)

```
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
iris = sns.load_dataset("iris")
y = iris.species
le = LabelEncoder()
le.fit(y)
species = le.transform(y)
species
```

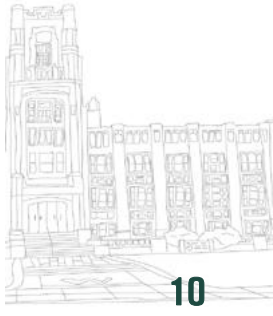
인코딩

- 원-핫 인코딩
  - 클래스의 수만큼 0과 유일한 1을 이용해 표현하는데 각 클래스마다 유일한 1의 위치하도록 인코딩 하는 것을 의미한다.
  - 만일 0부터 9까지 10개의 클래스가 있을 경우 0은 [1,0,0,0,0,0,0,0,0,0]으로 인코딩 되며 6은 [0,0,0,0,0,0,1,0,0,0]으로 인코딩 된다.
  - Ex)

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
enc.fit(species.reshape(-1,1))

iris_onehot = enc.transform(species.reshape(-1,1))
iris_onehot

iris_onehot.toarray()
```



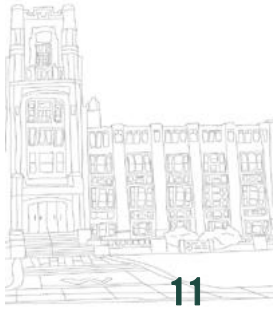
결측값 처리

결측값 처리

- 데이터에 누락된 정보(결측값, Missing Value)가 있을 경우 이 값을 다른 값으로 채워야 한다.
- 결측값 처리는 preprocessing 모듈의 Imputer 클래스를 이용하면 데이터의 평균, 중앙값 또는 최빈값 중 하나로 채울 수 있다.
- sklearn.preprocessing.Imputer(missing_values=nan, strategy='mean', verbose=0, copy=True)구문에서 strategy 에서 설정한다.
  - strategy : 결측값을 채울 방법을 지정하며 mean, median, most_frequent가 있다.
- Ex)

```
import seaborn as sns
iris = sns.load_dataset("iris")
x = iris.iloc[:,:-1]
x

import random
for col in range(4) : # 결측값 생성
    x.iloc[[random.sample(range(len(iris)), 10)],col] = float('nan')
x.head()
```



결측값 처리

```
x.mean(axis=0) # 평균값 확인

# 평균으로 채우기
from sklearn.impute import SimpleImputer
import numpy as np

imp_mean = SimpleImputer(missing_values=np.nan, strategy='mean')
imp_mean.fit_transform(x)[0:5,:]

x.median(axis=0) # 중앙값 확인

imp_mean = SimpleImputer(missing_values=np.nan, strategy='median')
imp_mean.fit_transform(x)[0:5,:]

x.mode(axis=0) # 최빈값 확인

imp_mean = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
imp_mean.fit_transform(x)[0:5,:]
```

단순 데이터 분리

단순 데이터 분리

- sklearn.model_selection.train_test_split()
  - 학습용 데이터와 검증용 데이터를 쉽게 나누도록 도와준다.
- Ex)

```
import seaborn as sns
from sklearn.model_selection import train_test_split

iris = sns.load_dataset("iris")

x_train, x_test, y_train, y_test = train_test_split(iris.iloc[:,0:4], iris.species, test_size=0.3)
x_train.shape
```