데이터 가공 및 시각화

- 1. 전처리
  - 결측값 처리: 단순대치, 평균 대치, 단순확률 대치 (Hot-deck, nearest neighbor), 다중 대치, knnImputation, centralimputation
  - 클래스불균형: 업샘플링 (SMOTE, Boaderline SMOTE, Adasyn), 다운샘플링
  - 이상값 처리: 극단값 절단, 조정
  - 변수 변환, 스케일링: 수치형 변수 변환(로그변환, 제곱근변환, 지수변환, 제곱변환, Box-cox 변환, 표준화, 정규화), 범주형 변수 변환(범주형 변수 인코딩, 대규모 범주형 변수처리), 날짜 및 변수 변환, 피쳐스케일링
  - 원핫인코딩(더미변수), 컬럼 트랜스퍼, 구간분할, 이산화, 피쳐선택
- 1. 표본 추출: 단순랜덤 추출법, 계통추출법, 집락추출법, 층화추출법
- 1. **데이터 분할: 구축/검정/시험용, 홀드아웃방법, 교차확인방법 (10 fold 교차분석), 부트스트랩**
- 1. 그래프 그리기:
  - 산점도, 막대그래프, 선그래프, 히트맵, 서브플롯, 트리맵, 도넛차트, 버블차트, 히스토그램, 체르노프 페이스, 스타차트, 다차원척도법, 평행좌표계
  - 도식화와 시각화

파이썬 데이터 리샘플링, 분할 방법들

부트스트랩

모집단에서 추출한 표본에 대해서 또 다시 재표본을 여러번 추출하여 모델을 평가하거나 데이터의 분포를 파악하는 재표본 추출방법이다. 샘플링을 할 때는 단순랜덤 복원 추출법을 사용하여 동일한 크기의 표본을 여러개 생성하므로, 특정 데이터가 여러 샘플에 포함될 수도 있고 혹은 어떠한 샘플에도 포함되지 않을수도 있다.

부트스트랩을 통해 100개의 샘플을 추출했을 때 샘플에 한번도 선택되지 않는 원데이터가 발생할 확률은 36.8%이다. 이러한 데이터를 OOB(out of bag)데이터라고 하며, OOB 데이터의 실제값과 예측값 사이의 오차로 정의되는 값을 OOB-error라고 한다. ensenble 기법들(Bagging, Boosting)에서 파라미터로 Bootstrap=True/False 적용하여, 재표본 추출할지 여부를 정할 수 있다.

단순랜덤복원추출법으로 동일한 크기의 여러 표본을 생성하려면, 지난번 포스팅한 단순랜덤 추출법을 참고하면 된다.

일반적인 데이터 분할 및 홀드아웃방법

일반적으로 데이터 분할은 train data(70%), test data(30%)로 비율에 따라 랜덤으로 데이터를 분할한다. 여기서 홀드아웃방법(Hold-out: 데이터를 랜덤하게 두 분류로 분리하여 교차 검정을 실시, 하나는 훈련용 데이터, 하나는 검증용 데이터로 사용 데이터를 랜덤하게 두 분류로 분리하여 교차 검정을 실시)은 train data(50%), test data(50%)로 데이터를 분할해서 교차검정을 실시하면 된다.

파이썬으로 데이터 분할을 구현하기 위해서는 데이터 셋을 설명변수(X)와 타겟변수(y)로 나눈 후, Scikit Learn의 train_test_split 함수를 사용하면 구할 수 있다. 홀드아웃의 경우 test_size 파라미터를 0.5로 지정하면 된다.

In [3]:
```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('/Users/benny/Desktop/datascience/heart.csv', na_values=['',
df.info()
X = df.drop('HeartDisease', axis=1)
y = df['HeartDisease']
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         916 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

In [5]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stra

print(len(X_train), len(X_test))
print(len(y_train), len(y_test))
print(len(X_train)/len(X), len(X_test)/len(X))
```

```
642 276
642 276
0.6993464052287581 0.3006535947712418
```

K-fold 교차분석 (k-fold cross validation)

데이터를 k개의 집단으로 나눈 뒤 k-1개의 집단으로 분류기를 학습시키고, 나머지 1개의 집단으로 분류기의 성능을 테스트하는 방법이다. 이 과정을 k번 반복하여 모든 데이터가 학습과 검증에 사용될 수 있도록 하고, 최종적으로 k번의 테스트를 통해 얻은 MSE(평균제곱오차)값들의 평균을 해당 모델의 MSE값으로 사용한다.

train set과 test set의 비율은 fold를 몇번으로 하느냐에 따라 달라진다. fold를 5로 가져가면, train set 4/5, test set 1/5의 비율이 되고, fold를 10으로 가져가면, train set 9/10, test set 1/10의 비율이 된다.

K-fold 교차분석은 Scikit Learn의 KFold 함수로 구현할 수 있다. 이 함수는 train과 test set으로 나뉘어진 인덱스만 반환한다. 그래서 이 반환된 인덱스를 원래의 데이터셋에 적용해서 그 인덱스에 해당하는 데이터값들을 받아와야 한다. KFold의 파라미터 n_splits는 fold의 횟수를 나타내고, shuffle은 데이터 분할 전에 데이터를 섞을지 여부를 나타낸다.

아래는 n_splits를 4로 했기 때문에 train set과 test set의 비율이 각각 0.75(=3/4), 0.25(=1/4)로 나타났다.

In [9]:

```
from sklearn.model_selection import KFold
kf = KFold(n_splits=10, shuffle=False) # n_splits=fold 개수, shuffle= 데이터 분할
kf.get_n_splits(X)
```

```
print(kf)

for train_index, test_index in kf.split(X):
    X_train, X_test, y_train, y_test = X.loc[train_index], X.loc[test_index],

print(len(X_train), len(X_test))
print(len(y_train), len(y_test))
print(len(X_train)/len(X), len(X_test)/len(X))
```

```
KFold(n_splits=10, random_state=None, shuffle=False)
827 91
827 91
0.900871459694989 0.09912854030501089
```

층화 K-fold 교차분석 (Stratified k-fold cross validation)

타겟 변수값이 랜덤으로 여러번 fold하는 과정에서 어떤 분할 데이터셋에서는 타겟 변수의 level 중 일부가 누락되거나 level의 비율이 현저히 다를 수 있다. 이것을 방지하기 위해, 모든 분할 데이터셋의 타겟 변수 level의 비율이 원본과 동일한 비율로 나누어지도록 하는 k-fold 교차분석 방법이다.

Scikit Learn의 StratifiedKFold 함수로 구현할 수 있으며, 이 역시 train과 test set의 인덱스를 반환해 준다. 아래는 반복문으로 데이터셋을 분할하고 있는데 실제로 모델을 fitting하는 식을 그 아래에 넣고, 모델링 결과를 모으는 코드를 그 아래에 넣을 수 있다.

아래 결과를 보면, 원본 타겟변수의 level 비율과 타겟변수 y의 훈련세트의 level 비율과 y의 test세트의 level 비율이 동일한 것을 확인 할 수 있다. 그와 동시에 train - test set의 비율은 k-fold에 따른 비율을 맞춘다.

In [13]:

```
from sklearn.model_selection import StratifiedKFold
skf = StratifiedKFold(n_splits=10)
skf.get_n_splits(X, y)

for train_index, test_index in kf.split(X):
    X_train, X_test, y_train, y_test = X.loc[train_index], X.loc[test_index],

print(len(X_train), len(X_test))
print(len(y_train), len(y_test))
print(len(X_train)/len(X), len(X_test)/len(X))

print(y.value_counts().values[0]/len(y))
print(y.value_counts().values[1]/len(y))

print(y_train.value_counts().values[0]/len(y_train))
print(y_train.value_counts().values[1]/len(y_train))

print(y_test.value_counts().values[0]/len(y_test))
print(y_test.value_counts().values[1]/len(y_test))
```

```
827 91
827 91
0.900871459694989 0.09912854030501089
0.5533769063180828
0.4466230936819172
0.5610640870616687
0.43893591293833134
0.5164835164835165
0.4835164835164835
```

Group K-fold 교차분석

매 데이터 분할 시마다 각 group의 데이터들 중 한 그룹의 데이터를 test 데이터로 적용하고 이를 돌아가면서 사용하여 k-fold를 진행한다. 이때문에 group의 개수는 fold의 개수와 같거나 fold 개수보다 커야 한다. group의 개수와 fold의 개수가 같을 경우, 모든 group들이 한번씩 test 데이터로 적용될 것이고, group의 개수보다 fold의 개수가 적을 경우 일부 group들은 test 데이터로 적용이 되지 않고 교차 분석이 완료될 것이다.

In [14]:
```
df.shape
```

Out[14]:
```
(918, 12)
```

In [15]:
```
df2 = df.copy()
df2['group']=1
df2['group'][:100]=0
df2['group'][500:]=2
df2['group'][800:]=3
```

```
/var/folders/02/0vw5fnqn5wb7bdg5tfs4cz4r0000gn/T/ipykernel_64107/240141084.py:
3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['group'][:100]=0
/var/folders/02/0vw5fnqn5wb7bdg5tfs4cz4r0000gn/T/ipykernel_64107/240141084.py:
4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['group'][500:]=2
/var/folders/02/0vw5fnqn5wb7bdg5tfs4cz4r0000gn/T/ipykernel_64107/240141084.py:
5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['group'][800:]=3
```

In [18]:
```
from sklearn.model_selection import GroupKFold
X = df2.drop('HeartDisease', axis=1)
y = df2['HeartDisease']
group = np.array(df2['group'])
gkf = GroupKFold(n_splits=3)

for train_index, test_index in gkf.split(X, y, group):
    X_train, X_test, y_train, y_test, G_train, G_test = X.loc[train_index], X
```

In [21]:
```
print(pd.DataFrame(G_train).value_counts())
print(pd.DataFrame(G_test).value_counts())
print(pd.DataFrame(group).value_counts())
```

```
1    400
2    300
dtype: int64
3    118
0    100
dtype: int64
1    400
```

```
2    300
3    118
0    100
dtype: int64
```

In [ ]: