

위스콘신 유방암 데이터의 기본 모델 만들기

학습 목표

- 보팅 방식의 앙상블 기법을 실습을 통해 알아본다.
- 배깅 방식의 앙상블 기법을 실습을 통해 알아본다.

학습 내용

- 위스콘신 유방암 데이터 세트에 대한 기본 모델 구현하기
- 보팅 방식과 배깅 방식의 앙상블 기법 구현해 보기

목차

[01 데이터 로드 및 전처리](#)

[02 모델 학습 및 평가 - LogisticRegression](#)

[03 배깅 방식의 앙상블 기법 활용 - 랜덤 포레스트](#)

01 데이터 로드 및 전처리

[목차로 이동하기](#)

데이터 로드 및 전처리

In [14]:

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib

from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_breast_cancer
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score
```



In [4]:

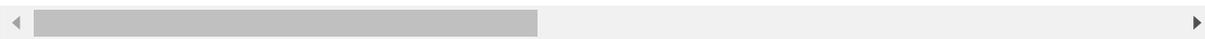


```
cancer = load_breast_cancer()
cancer_df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
cancer_df.head()
```

Out[4]:

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry |
|---|-------------|--------------|----------------|-----------|-----------------|------------------|----------------|---------------------|---------------|
| 0 | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.3001 | 0.14710 | 0.2419 |
| 1 | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.0869 | 0.07017 | 0.1812 |
| 2 | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.1974 | 0.12790 | 0.2069 |
| 3 | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.2414 | 0.10520 | 0.2597 |
| 4 | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.1980 | 0.10430 | 0.1809 |

5 rows × 30 columns



In [5]:



```
cancer_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
#   Column                               Non-Null Count  Dtype
---  -
0   mean radius                           569 non-null    float64
1   mean texture                           569 non-null    float64
2   mean perimeter                         569 non-null    float64
3   mean area                              569 non-null    float64
4   mean smoothness                        569 non-null    float64
5   mean compactness                       569 non-null    float64
6   mean concavity                          569 non-null    float64
7   mean concave points                    569 non-null    float64
8   mean symmetry                          569 non-null    float64
9   mean fractal dimension                 569 non-null    float64
10  radius error                           569 non-null    float64
11  texture error                           569 non-null    float64
12  perimeter error                         569 non-null    float64
13  area error                              569 non-null    float64
14  smoothness error                       569 non-null    float64
15  compactness error                      569 non-null    float64
16  concavity error                         569 non-null    float64
17  concave points error                   569 non-null    float64
18  symmetry error                          569 non-null    float64
19  fractal dimension error                 569 non-null    float64
20  worst radius                           569 non-null    float64
21  worst texture                           569 non-null    float64
22  worst perimeter                         569 non-null    float64
23  worst area                              569 non-null    float64
24  worst smoothness                       569 non-null    float64
25  worst compactness                      569 non-null    float64
26  worst concavity                         569 non-null    float64
27  worst concave points                    569 non-null    float64
28  worst symmetry                          569 non-null    float64
29  worst fractal dimension                 569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
```

In [6]:



```
print( cancer_df.shape)
```

```
(569, 30)
```

데이터 설명

- 위스콘신 유방암 데이터 세트는 유방암의 악성 종양, 양성 종양 여부를 결정하는 이진 분류
- 종양의 크기, 모양 등의 형태와 관련한 많은 피처를 가지고 있음.
- 569개의 행과, 30개의 피처로 이루어진 데이터
- null 값이 없음. 값들은 실수로 되어 있음.

데이터 나누기

In [10]:

```
# 피처와 레이블을 지정.  
X = cancer_df[:]  
y = cancer.target  
  
X.shape, y.shape
```

Out[10]:

```
((569, 30), (569,))
```

In [11]:

```
from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(X, y,  
                                                    test_size=0.2, random_state=0)  
  
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[11]:

```
((455, 30), (114, 30), (455,), (114,))
```

02 모델 학습 및 평가 - LogisticRegression

[목차로 이동하기](#)

In [13]:

```
# 모델 선택
model_log = LogisticRegression()
# 학습
model_log.fit(X_train, y_train)
# 예측
pred = model_log.predict(X_test)
pred[:15]
```

C:\Users\WwithJesus\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-learn.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 (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
n_iter_i = _check_optimize_result(

Out[13]:

```
array([0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1])
```

In [16]:

```
print("LogisticRegression 분류기 정확도 : {0:.4f}".format(accuracy_score(y_test, pred) ) )
```

LogisticRegression 분류기 정확도 : 0.9474

모델 학습 및 평가 - LogisticRegression

In [17]:

```
# 모델 선택
model_knn = KNeighborsClassifier()
# 학습
model_knn.fit(X_train, y_train)
# 예측
pred = model_knn.predict(X_test)
pred[:15]
```

Out[17]:

```
array([0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0])
```

In [18]:

```
print("KNeighborsClassifier 분류기 정확도 : {0:.4f}".format(accuracy_score(y_test, pred) ) )
```

LogisticRegression 분류기 정확도 : 0.9386

LogisticRegression의 정확도는 94.74%, Knn 모델의 정확도는 93.86%로 현재 결과로 LogisticRegression 우수하다.

보팅 분류기(Voting Classifier)를 활용한 앙상블 학습

- 앙상블 학습의 유형은 크게; 보팅(Voting), 배깅(Bagging), 부스팅(Boostin)의 세가지로 구분
 - 보팅(Voting) : 여러개의 머신러닝 알고리즘 활용하여 최종 예측 결과 결정
 - 배깅(Bagging) : 하나의 머신러닝 알고리즘 활용. 샘플 데이터를 다르게 하며 학습을 수행.
- 보팅 유형 - 하드 보팅(Hard Voting)과 소프트 보팅(Soft Voting)
 - 하드 보팅 : 다수의 분류기가 결정한 예측값을 최종 결과값으로 선정
 - 소프트 보팅 : 분류기들의 레이블 값 결정 확률을 더하고 이를 평균하여 확률이 가장 높은 레이블 값을 최종 결과값으로 선정.
 - 보통 소프트 보팅이 보팅 방법으로 적용됩니다.
- 사이킷 런은 보팅 방식의 앙상블을 Voting Classifier 클래스로 제공하고 있음.

In [35]:

```
from sklearn.ensemble import VotingClassifier
```

In [36]:

```
# 모델 선택
model_log = LogisticRegression()
model_knn = KNeighborsClassifier(n_neighbors=8)

# 개별 모델을 소프트 보팅 기반의 앙상블 모델로 구현
vo_clf = VotingClassifier( estimators=[("LR", model_log) ,
                                       ("KNN", model_knn)] , voting='soft') # 기본값 voting=hard
```

In [37]:

```
X_train , X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2, random_state=0)

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

Out[37]:

```
((455, 30), (114, 30), (455,), (114,))
```

In [38]:

```
### VotingClassifier 학습/예측 평가
vo_clf.fit(X_train, y_train)
pred = vo_clf.predict(X_test)
print("Voting 분류기 정확도 : {0:.4f}".format(accuracy_score(y_test, pred)))
```

Voting 분류기 정확도 : 0.9561

03 배깅 방식의 앙상블 기법 활용 - 랜덤 포레스트

[목차로 이동하기](#)

배깅 방식의 앙상블 기법 활용 - 랜덤 포레스트

- 랜덤 포레스트는 여러개의 결정 트리 분류기가 각각의 데이터 샘플링으로 개별적으로 학습 수행 후, 최종적으로 모든 분류기가 보팅을 통해 예측 결정하게 됨.

In [39]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

In [40]:

```
X_train , X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2, random_state=0)

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

Out[40]:

((455, 30), (114, 30), (455,), (114,))

In [42]:

```
# 모델 선택
model_rf = RandomForestClassifier()
# 학습
model_rf.fit(X_train, y_train)
# 예측
pred = model_rf.predict(X_test)
print( pred[:15] )
print("RandomForestClassifier 분류기 정확도 : {0:.4f}".format(accuracy_score(y_test, pred) ) )
```

[0 1 1 1 1 1 1 1 1 1 0 1 1 0 0]
RandomForestClassifier 분류기 정확도 : 0.9561

정리

- 앙상블 기법에는 보팅 방식, 배깅 방식, 부스팅 방식의 3가지 종류가 있다.
- VotingClassifier 는 보팅 방식의 앙상블 기법이다.
- 보팅 방식은 또 2가지 - 하드 보팅, 소프트 보팅 방식으로 나뉘어진다.

참조

- VotingClassifier : <https://clay-atlas.com/us/blog/2021/10/30/python-scikit-learn-convergence-warning/>
(<https://clay-atlas.com/us/blog/2021/10/30/python-scikit-learn-convergence-warning/>).

In []:

