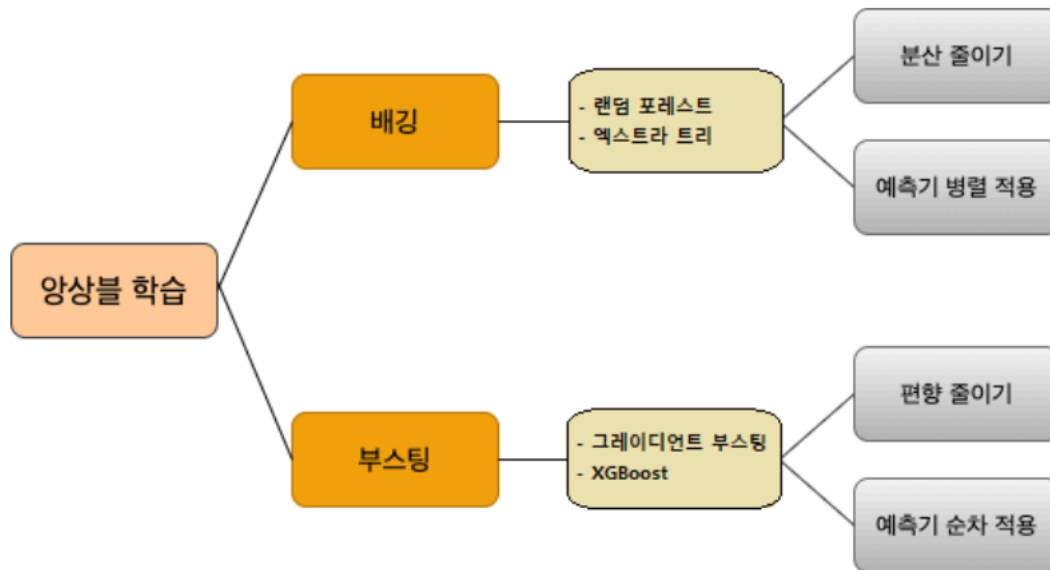


✓ 앙상블(Ensemble)

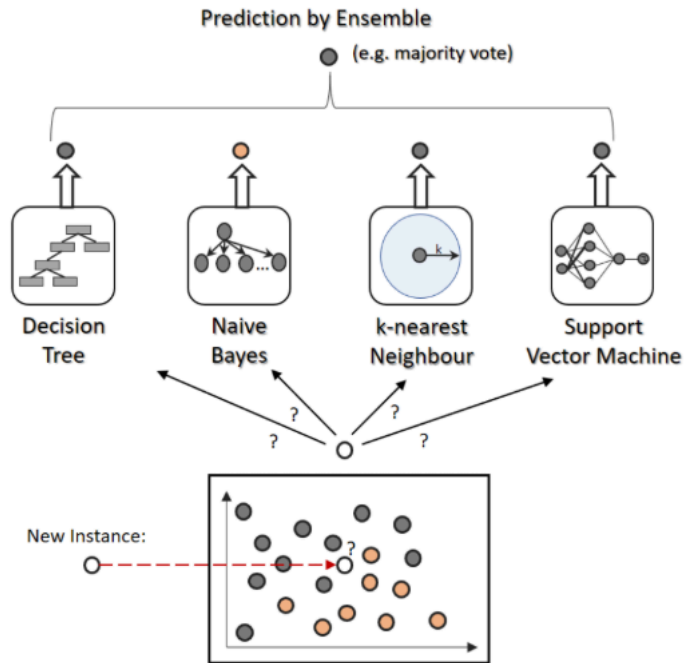
✓ 앙상블의 정의

- 여러 약한 분류기들을 결합하여 강 분류기로 만드는 것
 - 여러 개의 분류기(Classifier)를 생성하고 그 예측을 결합함으로써 보다 정확한 최종 예측을 도출하는 기법
- 앙상블 학습의 유형은 전통적으로 보팅(Voting), 배깅(Bagging), 부스팅(Boosting)의 세가지로 나눌 수 있으며, 이외에도 스택킹을 포함한 다양한 앙상블 기법이 있음



✓ 보팅(Voting)

- 서로 다른 알고리즘을 가진 분류기를 결합하는 것

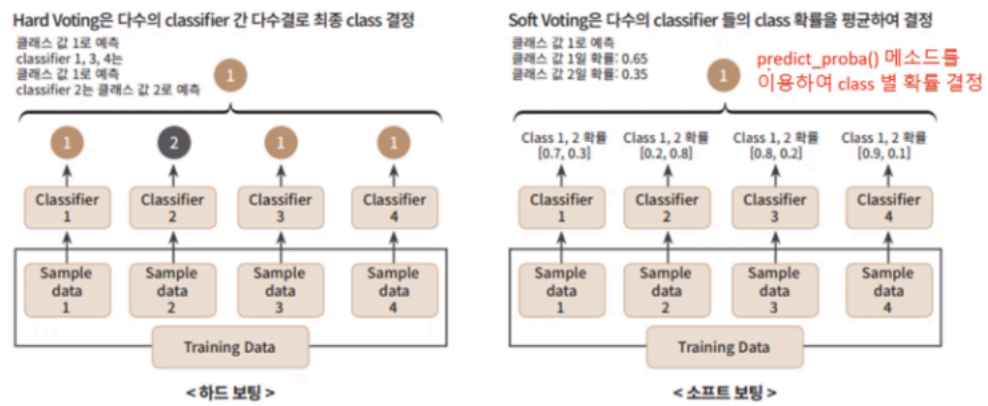


1) 하드 보팅

- 하드 보팅을 이용한 분류는 다수결 원칙과 비슷
- 다수의 분류기가 결정한 예측값을 최종 보팅 결과값으로 선정하는 것

2) 소프트 보팅

- 분류기들의 레이블 값 결정 확률을 모두 더하고 평균하여 가장 높은 레이블 값을 최종 보팅 결과값으로 선정
- 일반적으로 소프트 보팅이 많이 쓰임



```
1 #보팅을 한다고해서 항상 성능이 올라가진 않지만, 단일 ML 알고리즘보다 뛰어난 예측 성능을 가지는 경우가 많음
2 import pandas as pd
3 from sklearn.ensemble import VotingClassifier
4 from sklearn.linear_model import LogisticRegression
5 from sklearn.neighbors import KNeighborsClassifier
6 from sklearn.datasets import load_breast_cancer
7 from sklearn.model_selection import train_test_split
8 from sklearn.metrics import accuracy_score
9 #경고 메시지 무시
10 import warnings
11 warnings.filterwarnings('ignore')
12 cancer=load_breast_cancer()
13 data_df=pd.DataFrame(cancer.data,columns=cancer.feature_names)
14 data_df.head()
15 #데이터는 총 30개의 변수로 구성
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	sy
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	


5 rows × 30 columns

```
1 #보팅에 사용할 개별 분류 모델은 로지스틱회귀(Logistic Regression)와 KNN(K Nearest Neighbor)를 선택
2 #먼저 알아볼 방법은 소프트 보팅 방식
3 #이를 통해, 보팅 vs 로지스틱회귀(단일모델) vs KNN(단일모델)의 분류 성능을 비교
4 #개별 모델은 로지스틱 회귀와 KNN임
5 lr_clf=LogisticRegression()
6 knn_clf=KNeighborsClassifier(n_neighbors=8)
7
```

```

8 #개별 모델을 소프트 보팅 기반의 앙상블 모델로 구현한 분류기
9 vo_clf=VotingClassifier(estimators=[('LR',lr_clf),('KNN',knn_clf)],voting='soft')
10 X_train,X_test,y_train, y_test=train_test_split(cancer.data,cancer.target, test_size=0.2,random_state=156)
11
12 #VotingClassifier 학습/예측/평가
13 vo_clf.fit(X_train,y_train)
14 pred=vo_clf.predict(X_test)
15 print('Voting 분류기 정확도: {:.4f}'.format(accuracy_score(y_test,pred)))
16
17 #개별 모델의 학습/예측/평가.
18 classifiers=[lr_clf,knn_clf]
19 for classifier in classifiers:
20     classifier.fit(X_train, y_train)
21     pred=classifier.predict(X_test)
22     class_name=classifier.__class__.__name__
23     print('{0} 정확도: {:.4f}'.format(class_name,accuracy_score(y_test,pred)))
24
25 #결과는 Soft Voting 분류기가 단일 분류기들의 성과를 능가


```

 Voting 분류기 정확도: 0.9474
 LogisticRegression 정확도: 0.9386
 KNeighborsClassifier 정확도: 0.9386

```

1 #하드보팅 코드
2 # 개별 모델을 하드 보팅 기반의 앙상블 모델로 구현한 분류기
3 vo_clf=VotingClassifier(estimators=[('LR',lr_clf),('KNN',knn_clf)],voting='hard')
4
5 X_train,X_test,y_train,y_test=train_test_split(cancer.data,cancer.target,test_size=0.2,random_state=156)
6
7 # VotingClassifier 학습/예측/평가.
8 vo_clf.fit(X_train,y_train)
9 pred=vo_clf.predict(X_test)
10 print('Voting 분류기 정확도: {:.4f}'.format(accuracy_score(y_test,pred)))
11
12 # 개별 모델의 학습/예측/평가.
13 classifiers=[lr_clf,knn_clf]
14 for classifier in classifiers:
15     classifier.fit(X_train,y_train)
16     pred=classifier.predict(X_test)
17     class_name=classifier.__class__.__name__
18     print('{0} 정확도: {:.4f}'.format(class_name,accuracy_score(y_test,pred)))
19 #하드보팅 분류기는 단일 모델 분류기들의 정확도와 동일한 결과를 얻게 되었음

```

 Voting 분류기 정확도: 0.9386
 LogisticRegression 정확도: 0.9386
 KNeighborsClassifier 정확도: 0.9386

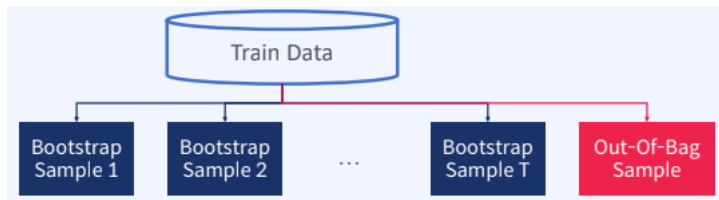
배깅(Bagging)

- Bagging = Bootstrap + Aggregation(즉, 약한 분류기들을 결합하여 강 분류기로 만드는 것)

- a) 부트스트랩(Bootstrap) 방법으로 원본 데이터에서 랜덤성과 중복을 허용하여 학습 샘플을 추출
- b) 정해진 모델로 각각의 샘플에 대해서 학습한 결과를 Aggregating(종합, 집합)해서 최적의 결과를 도출하는 방법
- 대표적으로 랜덤 포레스트가 있음
- Bagging의 작동방식
 - 1) 데이터의 부분 집합(Bootstrap 샘플)을 무작위로 생성
 - 원본 데이터에서 중복을 허용하고 랜덤하게 선택한 샘플로 구성
 - 2) 각 부분 집합에 대해 동일한 학습 알고리즘(예: 결정 트리)을 적용하여 여러 개의 모델을 생성
 - 3) 생성된 모델들의 예측을 평균화(회귀) 또는 다수결 투표(분류)를 통해 최종 예측을 수행 - 이때 보팅(투표)은 소프트 보팅 방식을 적용

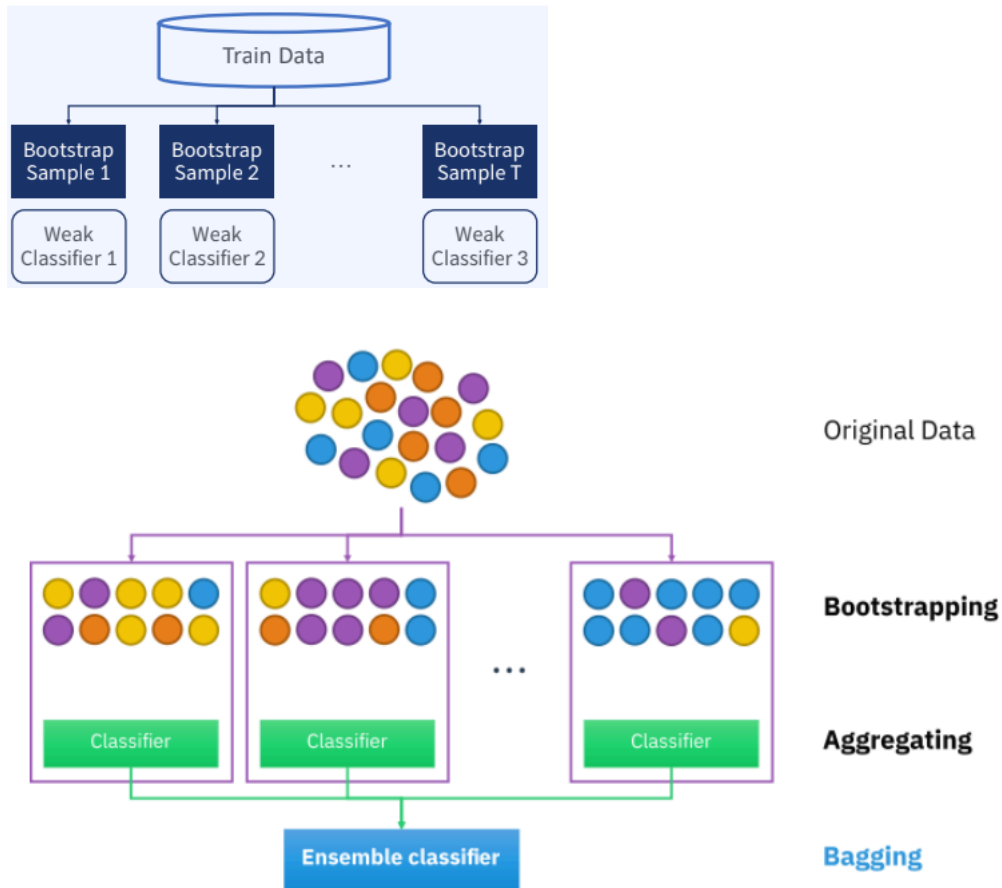
✓ 1) Bootstrap

- Train Data에서 여러 번 복원 추출하는 Random Sampling 기법(데이터 일부가 중첩됨) - 추출된 샘플들을 부트스트랩 샘플이라고 부름
- 이론적으로 36.8%의 샘플이 뽑히지 않게 됨(Out-Of-Bag 샘플)
- 추출되지 않는 샘플들을 이용해 교차 검증에서 Valid 데이터로 사용할 수 있음



✓ 2) Aggregation

- 추출된 부트스트랩 샘플마다 약 분류기를 학습
- 생성된 약 분류기들의 예측 결과를 Voting을 통해 결합함



• 배깅의 가장 큰 장점

- a) 과적합을 줄여 모델의 안정성을 향상시킴
- b) 병렬화가 가능하기 때문에 계산 성능이 향상됨

✓ <<<참조자료 사이트>>>

1. 앙상블(Ensemble)

2. 앙상블(Ensemble) 학습 (1) / 보팅(Voting)

3. [앙상블\(Ensemble\) 학습 \(3\) / 부스팅\(Boosting\) / GBM](#)