앙상블 Ensemble 3.

- 앙상블 심화
- 스태킹 앙상블

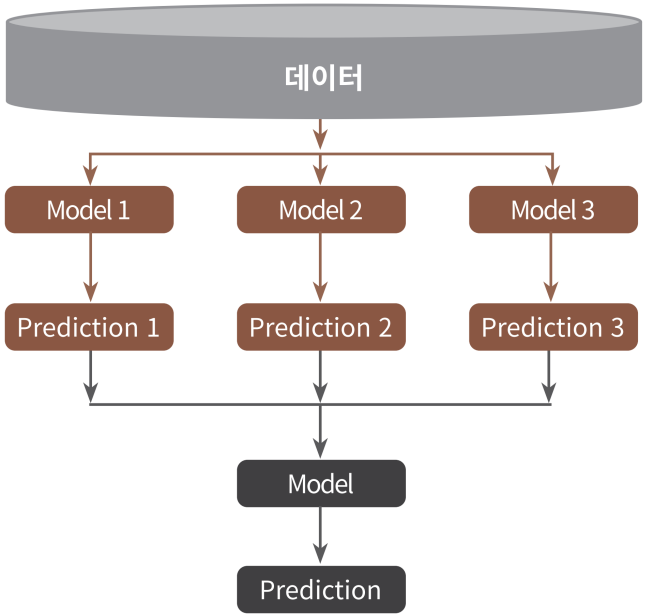
스태킹 앙상블(Stacking Ensemble)

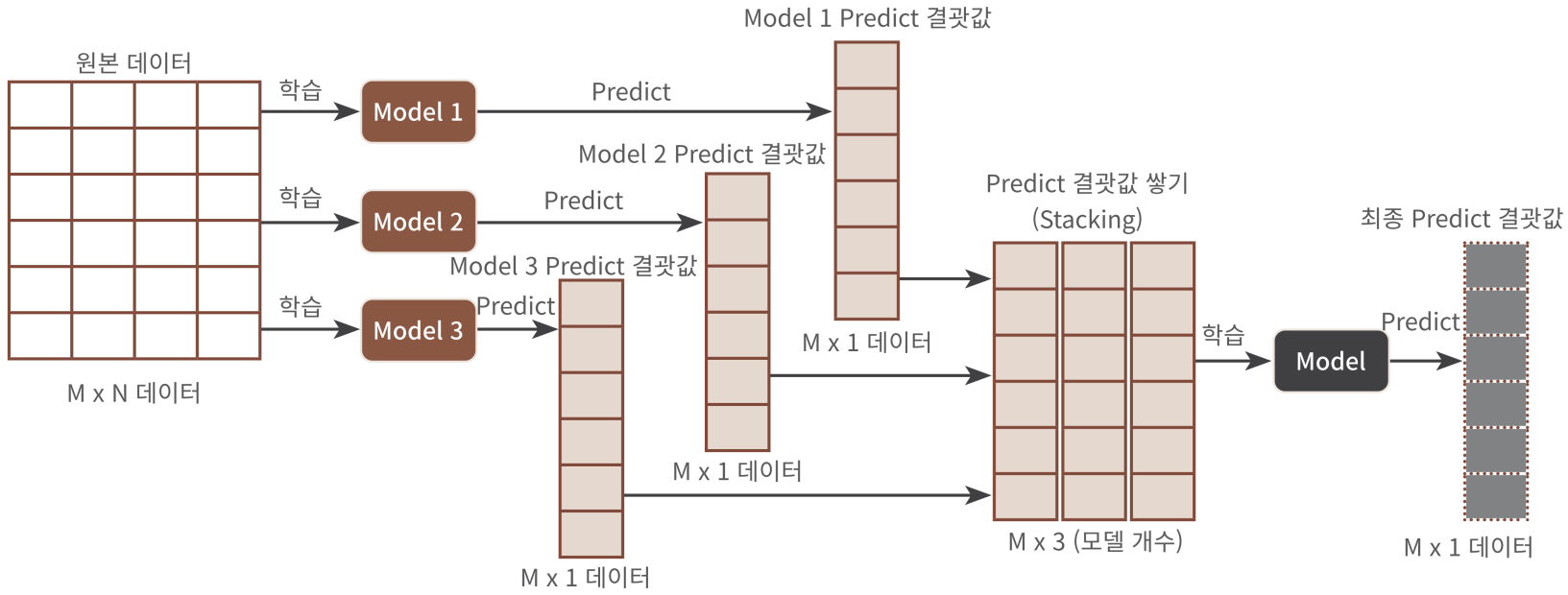
스태킹 앙상블(Stacking Ensemble)

스태킹 앙상블(Stacking Ensemble)

개별 모델의 예측된 데이터 세트를 다시 기반으로 하여 학습하고 예측하는 메타 모델을 말함.(블렌딩 모델이라고도 함)

개별 알고리즘의 예측 결과 데이터 세트를 각각 스태킹 형태로 결합하여 최종 메타 모델의 피처 데이터 세트와 피처 테스트 데이터 세트로 만들고 난 후, 별도의 ML 알고리즘으로 최종 학습을 수행하고 테스트 데이터를 기반으로 다시 최종 예측을 수행하는 방식. (두 종류의 학습 모델이 필요함)





※ 출처: 파이썬 머신러닝 완벽가이드, p279, 위키북스

[실습1] 위스콘신 암 데이터 세트 스태킹 모델 구축

- 필요한 모듈 임포트

```
import numpy as np

from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

- 데이터셋 준비 및 분할

```
cancer_data = load_breast_cancer()

X_data = cancer_data.data
y_label = cancer_data.target

X_train , X_test , y_train , y_test = ₩
    train_test_split(X_data , y_label , test_size=0.2 , random_state=0)
```

- 개별 모델 및 최종 스태킹 모델 객체 생성

```
# 개별 ML 모델을 위한 Classifier 생성.
knn_clf  = KNeighborsClassifier(n_neighbors=4)
rf_clf = RandomForestClassifier(n_estimators=100, random_state=0)
dt_clf = DecisionTreeClassifier()
ada_clf = AdaBoostClassifier(n_estimators=100)

# 최종 Stacking 모델을 위한 Classifier생성.
lr_final = LogisticRegression(C=10)

```

- 개별 모델 학습

```
# 개별 모델들을 학습.
knn_clf.fit(X_train, y_train)
rf_clf.fit(X_train , y_train)
dt_clf.fit(X_train , y_train)
ada_clf.fit(X_train, y_train)
```

- 개별 모델 예측 데이터셋 생성 및 성능 측정

```
# 학습된 개별 모델의 예측 데이터 셋을 생성하고 개별 모델의 정확도 측정.
knn_pred = knn_clf.predict(X_test)
rf_pred = rf_clf.predict(X_test)
dt_pred = dt_clf.predict(X_test)
ada_pred = ada_clf.predict(X_test)

print('KNN 정확도: {0:.4f}'.format(accuracy_score(y_test, knn_pred)))
print('랜덤 포레스트 정확도: {0:.4f}'.format(accuracy_score(y_test, rf_pred)))
print('결정 트리 정확도: {0:.4f}'.format(accuracy_score(y_test, dt_pred)))
print('에이다부스트 정확도: {0:.4f} :'.format(accuracy_score(y_test, ada_pred)))
```

- 메타 모델을 위한 학습 데이터셋 생성

```
pred = np.array([knn_pred, rf_pred, dt_pred, ada_pred])
print(pred.shape)

# transpose를 이용해 행과 열의 위치 교환.
# 컬럼 레벨로 각 알고리즘의 예측 결과를 피처로 만듦.
pred = np.transpose(pred)
print(pred.shape)
```

```
(4, 114)
(114, 4)
```

- 최종 메타모델(로지스틱 회귀) 학습 및 성능 평가

```
1  lr_final.fit(pred, y_test)
2  final = lr_final.predict(pred)
3
4  print('최종 메타 모델의 예측 정확도: {0:.4f}'.₩
5            format(accuracy_score(y_test , final)))
```

최종 메타 모델의 예측 정확도: 0.9737

※ 메타 모델이 학습할 때 학습용 레이블 데이터셋이 아닌 테스트용 레이블 데이터셋을 기반으로 학습함을 유의할 것
- 과적합 발생 가능성 있음

[실습2] CV 세트 기반 - 위스콘신 암 데이터 세트 스태킹 모델 구축

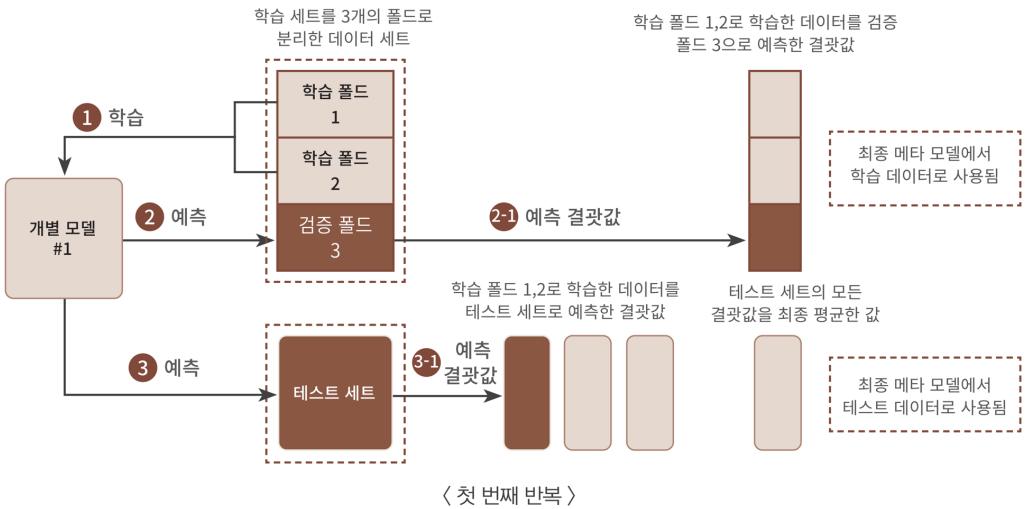
과적합을 개선하기 위해 최종 메타 모델을 위한 데이터 셋을 만들 때 교차 검증 기반으로 예측된 결과 데이터 셋을 활용하는 방법

개별 모델은 각각 교차 검증으로 메타 모델을 위한 학습용 스태킹 데이터 및 예측용 테스트 스태킹 데이터 셋을 생성하고 메타 모델은 이를 기반으로 학습과 예측을 수행함.

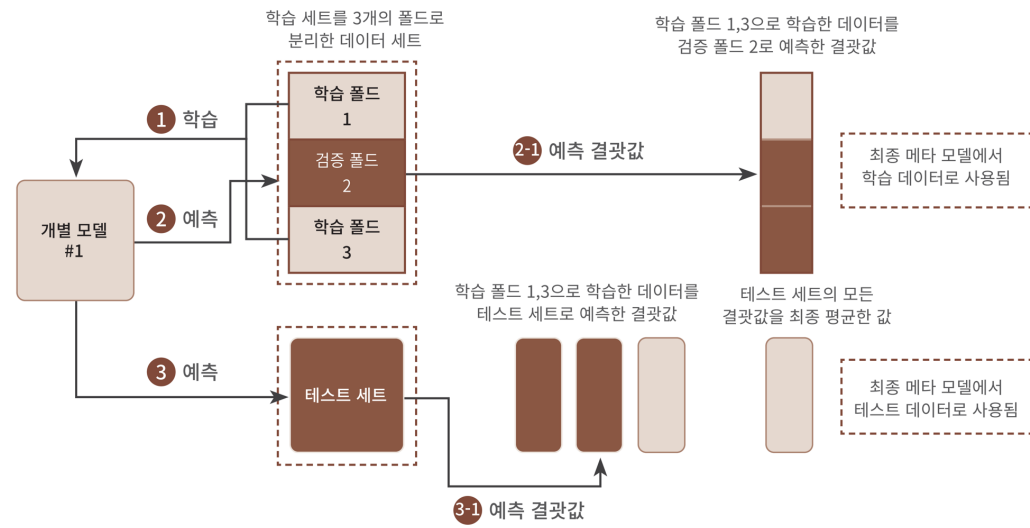
Step1 : 각 모델은 원본 학습/테스트 데이터를 예측한 결과 값을 기반으로 메타 모델을 위한 학습용/테스트용 데이터셋 생성(개별 모델 레벨에서 수행되며 이러한 로직은 여러 개의 개별 모델에서 동일하게 수행됨)

Step2: 개별 모델이 생성한 학습용(테스트용) 데이터를 모두 스태킹 형태로 합쳐서 메타 모델이 학습할 최종 학습용(테스트용) 데이터 셋 생성.

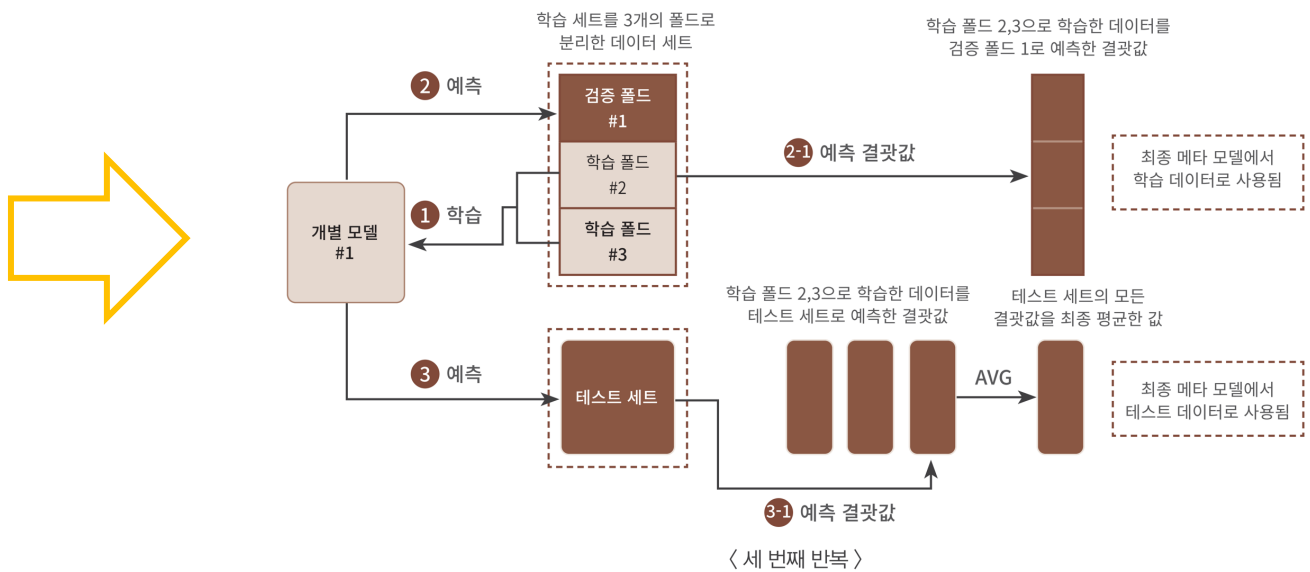
Step3: 메타 모델은 최종적으로 생성된 학습 데이터셋과 원본 학습 데이터의 레이블 데이터를 기반으로 학습한 뒤, 최종적으로 생성된 테스트 데이터셋을 사용하여 예측하고, 원본 테스트 데이터셋의 레이블 데이터를 기반으로 평가함



〈 첫 번째 반복 〉

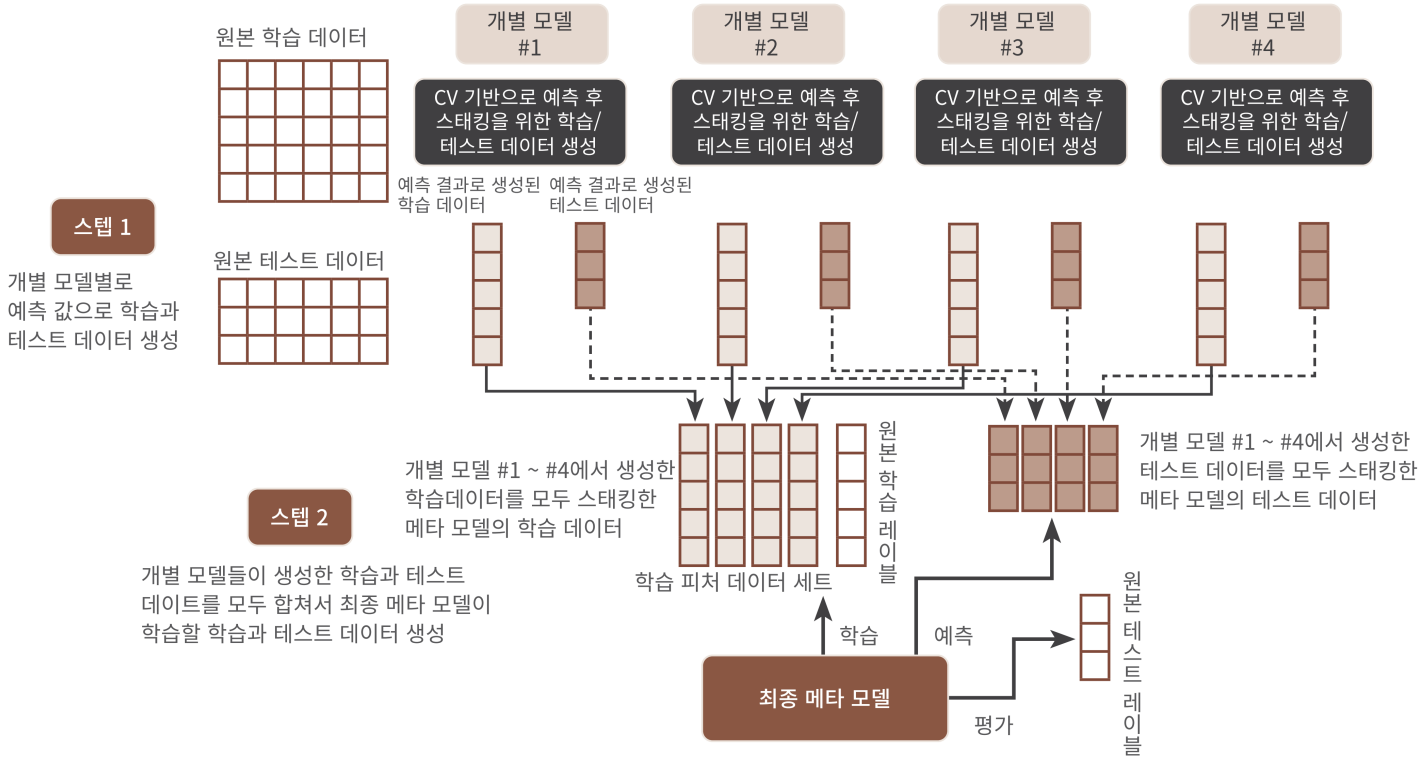


〈 두 번째 반복 〉



〈 세 번째 반복 〉

※ 출처: 파이썬 머신러닝 완벽가이드, p286, 위키북스



※ 출처: 파이썬 머신러닝 완벽가이드, p286, 위키북스

```python
from sklearn.model_selection import KFold
from sklearn.metrics import mean_absolute_error

# 개별 기반 모델에서 최종 메타 모델이 사용할 학습 및 테스트용 데이터를 생성하기 위한 함수.
def get_stacking_base_datasets(model, X_train_n, y_train_n, X_test_n, n_folds ):
    # 지정된 n_folds값으로 KFold 생성.
    kf = KFold(n_splits=n_folds, shuffle=False, random_state=0)
    #추후에 메타 모델이 사용할 학습 데이터 반환을 위한 넘파이 배열 초기화
    train_fold_pred = np.zeros((X_train_n.shape[0] ,1 ))
    test_pred = np.zeros((X_test_n.shape[0],n_folds))
    print(model.__class__.__name__ , ' model 시작 ')

    for folder_counter , (train_index, valid_index) in enumerate(kf.split(X_train_n)):
        #입력된 학습 데이터에서 기반 모델이 학습/예측할 폴드 데이터 셋 추출
        print('₩t 폴드 세트: ',folder_counter,' 시작 ')
        X_tr = X_train_n[train_index]
        y_tr = y_train_n[train_index]
        X_te = X_train_n[valid_index]

        #폴드 세트 내부에서 다시 만들어진 학습 데이터로 기반 모델의 학습 수행.
        model.fit(X_tr , y_tr)
        #폴드 세트 내부에서 다시 만들어진 검증 데이터로 기반 모델 예측 후 데이터 저장.
        train_fold_pred[valid_index, :] = model.predict(X_te).reshape(-1,1)
        #입력된 원본 테스트 데이터를 폴드 세트내 학습된 기반 모델에서 예측 후 데이터 저장.
        test_pred[:, folder_counter] = model.predict(X_test_n)

    # 폴드 세트 내에서 원본 테스트 데이터를 예측한 데이터를 평균하여 테스트 데이터로 생성
    test_pred_mean = np.mean(test_pred, axis=1).reshape(-1,1)

    #train_fold_pred는 최종 메타 모델이 사용하는 학습 데이터, test_pred_mean은 테스트 데이터
    return train_fold_pred , test_pred_mean
```

- 메타모델의 학습용/테스트용 데이터셋 준비

```
Stack_final_X_train = np.concatenate((knn_train, rf_train, dt_train, ada_train), axis=1)
Stack_final_X_test = np.concatenate((knn_test, rf_test, dt_test, ada_test), axis=1)
print('원본 학습 피처 데이터 Shape:',X_train.shape, '원본 테스트 피처 Shape:',X_test.shape)
print('스태킹 학습 피처 데이터 Shape:', Stack_final_X_train.shape,
      '스태킹 테스트 피처 데이터 Shape:',Stack_final_X_test.shape)
```

```
원본 학습 피처 데이터 Shape: (455, 30) 원본 테스트 피처 Shape: (114, 30)
스태킹 학습 피처 데이터 Shape: (455, 4) 스태킹 테스트 피처 데이터 Shape: (114, 4)
```

- 메타모델 학습, 예측 및 성능 평가

```
lr_final.fit(Stack_final_X_train, y_train)
stack_final = lr_final.predict(Stack_final_X_test)

print('최종 메타 모델의 예측 정확도: {0:.4f}'.format(accuracy_score(y_test, stack_final)))
```

```
최종 메타 모델의 예측 정확도: 0.9737
```