위스콘신 유방암 데이터의 자동 시스템 만들기

학습 목표

- 자동 시스템을 구현해 본다.

목차

01 라이브러리 불러오기

목차로 이동하기

In [1]:

```
import pandas as pd
from sklearn.model_selection import ParameterGrid, KFold

from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier

from sklearn.metrics import *
# from functools import partial
import numpy as np
import warnings
warnings.filterwarnings(action='ignore')
```

02 automl 클래스 만들기

목차로 이동하기

In [2]:

```
class MyAutoML1:
    ## 생성자 - 처음 모델이 만들어질때
    def __init__(self,
        exclude_models=[],   # 제외 모델
        seed=None,
        cv=5,                # 교차검증 폴더수
        scoring="accuracy",  # 평가지표 지정
        summarize_scoring="mean",
        early_stopping=False,       # 조기 종료
        early_stopping_criteria=0.1, # 조기 종료 기준
    ):
        # self.exclude_models 정의
        model_set = {"KNN", "DT", "RF"}       # 현재 모델
        self.exclude_models = exclude_models

        # self.seed 정의
        self.seed = seed

        # self.cv 정의 - 교차검증 폴더 수
        self.cv = cv

        # self.scoring 정의 - 평가지표 지정
        scoring_dict = {
            "accuracy": accuracy_score,
            "precision": precision_score,
            "recall": recall_score,
            "f1": f1_score
        }
        self.scoring = scoring_dict[scoring]

        # self.summarize_scoring 정의
        summarize_scoring_dict = {"mean": np.mean, "max": np.max, "min": np.min}
        self.summarize_scoring = summarize_scoring_dict[summarize_scoring]

        # 조기 종료 지정
        self.early_stopping = early_stopping

        # 조기 종료 조건 지정
        self.early_stopping_criteria = early_stopping_criteria

    ## 학습 메서드 - fit
    def fit(self, X, y):
        # 자료형을 확인 후, X, y의 값을 지정
        if isinstance(X, pd.DataFrame):
            X = X.values
        elif isinstance(X, list) or isinstance(X, tuple):
            X = np.array(X)
        if isinstance(y, pd.Series):
            y = y.values
        elif isinstance(y, list) or isinstance(y, tuple):
            y = np.array(y)

        # K최근접 이웃 모델 파라미터 정의
        kNN_grid = ParameterGrid(
            {"n_neighbors": [3, 5, 7, 9, 11], "metric": ["euclidean", "manhattan"]}
        )
        # 의사 결정 트리 모델 - 파라미터 정의
        DT_grid = ParameterGrid(
            {"max_depth": [3, 5, 7, 9], "min_samples_split": [2, 5, 10]}
```

```
    )

    # 랜덤 포레스트 모델 - 파라미터 정의
    RFR_grid = ParameterGrid(
        {
            "n_estimators": [50, 100, 200],
            "max_depth": [2, 3, 4],
            "max_features": [0.2, 0.4, 0.6, 0.8, 1.0],
        }
    )

    # 전체 파라미터 정의
    grid = {
        KNeighborsClassifier: kNN_grid,
        DecisionTreeClassifier: DT_grid,
        RandomForestClassifier: RFR_grid
    }

    # 그리드 서치 시작
    best_score = 0
    self.leaderboard = []

    # 모델별 반복
    for model_func in grid.keys():

        # 모델이 제외 모델이 있으면 다음, 없으면 해당 모델의 파라미터 반복
        if model_func in self.exclude_models:
            continue
        for params in grid[model_func]:
            if model_func != KNeighborsClassifier:
                params["random_state"] = self.seed
            kf = KFold(n_splits=self.cv, shuffle=True, random_state=self.seed)
            fold_score_list = []

            # 조기 종료를 하는 경우
            if self.early_stopping:
                for train_index, test_index in kf.split(X):
                    X_train, X_test = X[train_index], X[test_index]
                    y_train, y_test = y[train_index], y[test_index]
                    model = model_func(**params).fit(X_train, y_train)
                    y_pred = model.predict(X_test)
                    fold_score = self.scoring(y_test, y_pred)
                    fold_score_list.append(fold_score)
                    # 조기 종료 조건 추가
                    if fold_score < best_score * (1 - self.early_stopping_criteria):
                        break

            # 조기 종료를 하지 않는 경우
            else:
                for train_index, test_index in kf.split(X):
                    X_train, X_test = X[train_index], X[test_index]
                    y_train, y_test = y[train_index], y[test_index]
                    model = model_func(**params).fit(X_train, y_train)
                    y_pred = model.predict(X_test)
                    fold_score = self.scoring(y_test, y_pred)
                    fold_score_list.append(fold_score)

            # 현재까지 찾은 최고의 답안 및 리더보드 업데이트
            score = self.summarize_scoring(fold_score_list)
            if score > best_score:
                best_score = score
```

```
                    best_model_func = model_func
                    best_params = params
                self.leaderboard.append([model_func, params, score])

        self.model = best_model_func(**best_params).fit(X, y)

        # 모델별 리더보드에 저장
        self.leaderboard = pd.DataFrame(self.leaderboard,
                                        columns=["모델", "파라미터", "점수"])

    ## 예측 - predict 메서드
    def predict(self, X):
        return self.model.predict(X)

    ## show_leaderboard 메서드
    def show_leaderboard(self):
        return self.leaderboard
```

03 automl 클래스를 활용한 학습 및 결과 확인

목차로 이동하기

In [3]:

```
from sklearn.datasets import load_breast_cancer
```

In [4]:

```
cancer = load_breast_cancer()

cancer_df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
cancer_df['y'] = cancer.target
cancer_df.head()
```

Out[4]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mear symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.241
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.181
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.206
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.259
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.180

5 rows × 31 columns

In [5]:

```
# 데이터 불러오기
X = cancer_df.drop('y', axis = 1)
y = cancer_df['y']
```

In [6]:

```
%%time

aml = MyAutoML1()
aml.fit(X, y)
```

```
Wall time: 48.4 s
```

In [7]:

```
result = aml.show_leaderboard()
display(result.sort_values(by = "점수", ascending = False))
```

	모델	파라미터	점수
54	<class 'sklearn.ensemble._forest.RandomForestC...	{'max_depth': 4, 'max_features': 0.2, 'n_estim...	0.961372
45	<class 'sklearn.ensemble._forest.RandomForestC...	{'max_depth': 3, 'max_features': 0.6, 'n_estim...	0.961341
56	<class 'sklearn.ensemble._forest.RandomForestC...	{'max_depth': 4, 'max_features': 0.4, 'n_estim...	0.961310
55	<class 'sklearn.ensemble._forest.RandomForestC...	{'max_depth': 4, 'max_features': 0.4, 'n_estim...	0.959556
57	<class 'sklearn.ensemble._forest.RandomForestC...	{'max_depth': 4, 'max_features': 0.4, 'n_estim...	0.959540
...	...	...	...
12	<class 'sklearn.tree._classes.DecisionTreeClas...	{'max_depth': 3, 'min_samples_split': 10, 'ran...	0.922652
18	<class 'sklearn.tree._classes.DecisionTreeClas...	{'max_depth': 7, 'min_samples_split': 10, 'ran...	0.919190
17	<class 'sklearn.tree._classes.DecisionTreeClas...	{'max_depth': 7, 'min_samples_split': 5, 'rand...	0.915588
21	<class 'sklearn.tree._classes.DecisionTreeClas...	{'max_depth': 9, 'min_samples_split': 10, 'ran...	0.912172
20	<class 'sklearn.tree._classes.DecisionTreeClas...	{'max_depth': 9, 'min_samples_split': 5, 'rand...	0.906816

67 rows × 3 columns

```
    def __init__(
        self,
        exclude_models=[],  # 제외 모델
        seed=None,
        cv=5,
        scoring="accuracy",
```

```
        summarize_scoring="mean",
        early_stopping=False,
        early_stopping_criteria=0.1,
    ):
```

평가기준과 조기종료 가능하도록, seed값을 설정해 둔 이후에 확인.

In [8]:

```
aml = MyAutoML1(scoring='f1', early_stopping=True, seed=2022)
aml.fit(X, y)
result = aml.show_leaderboard()
result.columns
```

Out[8]:

```
Index(['모델', '파라미터', '점수'], dtype='object')
```

In [9]:

```
display(result.sort_values(by = "점수", ascending = False))
```

	모델	파라미터	점수
58	<class 'sklearn.ensemble._forest.RandomForestC...	{'max_depth': 4, 'max_features': 0.6, 'n_estim...	0.966958
55	<class 'sklearn.ensemble._forest.RandomForestC...	{'max_depth': 4, 'max_features': 0.4, 'n_estim...	0.965332
57	<class 'sklearn.ensemble._forest.RandomForestC...	{'max_depth': 4, 'max_features': 0.4, 'n_estim...	0.965193
61	<class 'sklearn.ensemble._forest.RandomForestC...	{'max_depth': 4, 'max_features': 0.8, 'n_estim...	0.964033
59	<class 'sklearn.ensemble._forest.RandomForestC...	{'max_depth': 4, 'max_features': 0.6, 'n_estim...	0.963952
...	...	...	...
15	<class 'sklearn.tree._classes.DecisionTreeClas...	{'max_depth': 5, 'min_samples_split': 10, 'ran...	0.939285
17	<class 'sklearn.tree._classes.DecisionTreeClas...	{'max_depth': 7, 'min_samples_split': 5, 'rand...	0.936190
21	<class 'sklearn.tree._classes.DecisionTreeClas...	{'max_depth': 9, 'min_samples_split': 10, 'ran...	0.936190
19	<class 'sklearn.tree._classes.DecisionTreeClas...	{'max_depth': 9, 'min_samples_split': 2, 'rand...	0.935019
20	<class 'sklearn.tree._classes.DecisionTreeClas...	{'max_depth': 9, 'min_samples_split': 5, 'rand...	0.934753

67 rows × 3 columns

In [10]:

```
result.to_excel("result_f1_score.xlsx", index=False)
```

실습

- 추가 모델을 추가하여 확인해 보자.