# 위스콘신 유방암 데이터의 자동 시스템 만들기

## 학습 목표

- 자동 시스템을 구현해 본다.

## 라이브러리 불러오기

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
```

## 클래스 만들기

In [3]:

```
class MyAutoML1:
    ## 생성자
    def __init__(
        self,
        exclude_models=[],  # 제외 모델
        seed=None,
        cv=5,
        scoring="accuracy",
        summarize_scoring="mean",
        early_stopping=False,
        early_stopping_criteria=0.1,
    ):
        # self.exclude_models 정의
        model_set = {"KNN", "DT", "RF"}
        self.exclude_models = exclude_models

        # self.seed 정의
        self.seed = seed

        # self.cv 정의
        self.cv = cv

        # self.scoring 정의
        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]

        # self.early_stopping 정의
        self.early_stopping = early_stopping

        # early_stopping_criteria 정의
        self.early_stopping_criteria = early_stopping_criteria

    ## fit 메서드
    def fit(self, X, y):
        # X, y 포맷 변경
        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 그리드 정의
        kNN_grid = ParameterGrid(
            {"n_neighbors": [3, 5, 7, 9, 11], "metric": ["euclidean", "manhattan"]}
        )
        # 결정 나무 그리드 정의
        DT_grid = ParameterGrid(
            {"max_depth": [3, 5, 7, 9], "min_samples_split": [2, 5, 10]}
        )

        # 랜덤 포레스트 그리드 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 그리드 정의
        grid = {
            KNeighborsClassifier: kNN_grid,
            DecisionTreeClassifier: DT_grid,
            RandomForestClassifier: RFR_grid
        }

        # 그리드 서치 시작
        best_score = 0
        self.leaderboard = []
        for model_func in grid.keys():
            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)
                        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 해 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)
        self.leaderboard = pd.DataFrame(self.leaderboard,
                                        columns=["모델", "파라미터", "점수"])

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

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

## 적용

In [5]:

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

In [7]:

```
cancer = load_breast_cancer()

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

Out[7]:

| mean<br>ncave<br>oints | mean symmetry | mean fractal dimension | ... | worst texture | worst perimeter | worst area | worst smoothness | worst compactness | worst concavity |
|---|---|---|---|---|---|---|---|---|---|
| 14710 | 0.2419 | 0.07871 | ... | 17.33 | 184.60 | 2019.0 | 0.1622 | 0.6656 | 0.7119 |
| 07017 | 0.1812 | 0.05667 | ... | 23.41 | 158.80 | 1956.0 | 0.1238 | 0.1866 | 0.2416 |
| 12790 | 0.2069 | 0.05999 | ... | 25.53 | 152.50 | 1709.0 | 0.1444 | 0.4245 | 0.4504 |
| 10520 | 0.2597 | 0.09744 | ... | 26.50 | 98.87 | 567.7 | 0.2098 | 0.8663 | 0.6869 |
| 10430 | 0.1809 | 0.05883 | ... | 16.67 | 152.20 | 1575.0 | 0.1374 | 0.2050 | 0.4000 |

In [8]:

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

In [9]:

```
aml = MyAutoML1()
aml.fit(X, y)
result = aml.show_leaderboard()
display(result.sort_values(by = "점수", ascending = False))
```

| | 모델 | 파라미터 | 점수 |
|---|---|---|---|
| **66** | <class 'sklearn.ensemble._forest.RandomForestC... | {'max_depth': 4, 'max_features': 1.0, 'n_estim... | 0.959603 |
| **57** | <class 'sklearn.ensemble._forest.RandomForestC... | {'max_depth': 4, 'max_features': 0.4, 'n_estim... | 0.959603 |
| **40** | <class 'sklearn.ensemble._forest.RandomForestC... | {'max_depth': 3, 'max_features': 0.4, 'n_estim... | 0.959571 |
| **63** | <class 'sklearn.ensemble._forest.RandomForestC... | {'max_depth': 4, 'max_features': 0.8, 'n_estim... | 0.959571 |
| **64** | <class 'sklearn.ensemble._forest.RandomForestC... | {'max_depth': 4, 'max_features': 1.0, 'n_estim... | 0.959525 |
| **...** | ... | ... | ... |
| **11** | <class 'sklearn.tree._classes.DecisionTreeClas... | {'max_depth': 3, 'min_samples_split': 5, 'rand... | 0.926129 |
| **12** | <class 'sklearn.tree._classes.DecisionTreeClas... | {'max_depth': 3, 'min_samples_split': 10, 'ran... | 0.924437 |
| **21** | <class 'sklearn.tree._classes.DecisionTreeClas... | {'max_depth': 9, 'min_samples_split': 10, 'ran... | 0.922683 |
| **17** | <class 'sklearn.tree._classes.DecisionTreeClas... | {'max_depth': 7, 'min_samples_split': 5, 'rand... | 0.920882 |
| **15** | <class 'sklearn.tree._classes.DecisionTreeClas... | {'max_depth': 5, 'min_samples_split': 10, 'ran... | 0.913880 |

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 [11]:

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

Out[11]:

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

In [12]:

```
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 [14]:

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

## 실습

- 추가 모델을 추가하여 확인해 보자.