Assignment 6: AutoML

2016314364 박수헌

다섯 개의 dataset을 pandas.read_csv를 통해 읽어들인 후, trainval set에 500개의 instance, test set에 나머지 instance가 들어갈 수 있게 train_test_split을 이용해 split 하였다. Pipeline의 내용으로는 preprocessor, regressor를 설정했다. 각 model들의 hyper parameter에 대한 search space는 아래와 같이 설정했다. Gridsearch의 scoring은 'neg_mean_squared_error'로 설정하였으므로 마지막 test set의 RMSE를 확인하는 부분에서는 음수값을 취했다.

{'regressor' : [LinearRegression()], 'preprocessing' : [StandardScaler(), MinMaxScaler(), None]},

{'regressor' : [Ridge()], 'preprocessing' : [StandardScaler(), MinMaxScaler(), None], 'regressor__alpha' : [0, 0.01, 1, 10, 100]},

{'regressor' : [Lasso()], 'preprocessing' : [StandardScaler(), MinMaxScaler(), None], 'regressor__alpha' : [0.0001, 0.001, 0.01, 0.1, 1, 10], 'regressor__tol':[0.00001, 0.0001,0.001]},

{'regressor' : [RandomForestRegressor()], 'preprocessing' : [None], 'regressor__max_features' : ['auto', 'sqrt', 'log2']}, {'regressor' : [SVR()], 'preprocessing' : [StandardScaler(), MinMaxScaler(), None], 'regressor__epsilon' : [0.001, 0.01, 0.1], 'regressor__gamma' : [0.01, 0.1], 'regressor__C': [1, 100]},

{'regressor' : [MLPRegressor()], 'preprocessing' : [StandardScaler(), MinMaxScaler(), None], 'regressor__max_iter' : [5000,10000], 'regressor__activation' : ['tanh', 'relu'], 'regressor__solver' : ['lbfgs', 'sgd', 'adam'], 'regressor__hidden_layer_sizes': [(10,),(20,),(50,),(100,)]}]

결과는 아래와 같았다.

<abalone> best hyperparam : {'preprocessing': None, 'regressor': MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(50,), learning_rate='constant', learning_rate_init=0.001, max_fun=15000, max_iter=10000, momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5, random_state=None, shuffle=True, solver='sgd', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False), 'regressor__activation': 'relu', 'regressor__hidden_layer_sizes': (50,), 'regressor__max_iter': 10000, 'regressor__solver': 'sgd'}

best cross-validation score : 5.13 / test-set score : 4.61

<concrectecs> best hyperparam : {'preprocessing': StandardScaler(copy=True, with_mean=True, with_std=True), 'regressor': MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(100,), learning_rate='constant', learning_rate_init=0.001, max_fun=15000, max_iter=10000, momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5, random_state=None, shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False), 'regressor__activation': 'relu', 'regressor__hidden_layer_sizes': (100,), 'regressor__max_iter': 10000, 'regressor__solver': 'adam'}

best cross-validation score : 30.88 / test-set score : 38.61

<parkinsons> best hyperparam : {'preprocessing': None, 'regressor': RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse', max_depth=None, max_features='auto', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None, oob_score=False, random_state=None, verbose=0, warm_start=False), 'regressor__max_features': 'auto'}

best cross-validation score : 81.62 / test-set score : 90.43

<skillcraft> best hyperparam : {'preprocessing': MinMaxScaler(copy=True, feature_range=(0, 1)), 'regressor': Ridge(alpha=1, copy_X=True, fit_intercept=True, max_iter=None, normalize=False, random_state=None, solver='auto', tol=0.001), 'regressor__alpha': 1}

best cross-validation score : 0.00 / test-set score : 0.00

<winequality-white>best hyperparam : {'preprocessing': None, 'regressor': RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse', max_depth=None, max_features='log2', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None, oob_score=False, random_state=None, verbose=0, warm_start=False), 'regressor__max_features': 'log2'}

best cross-validation score : 0.51 / test-set score : 0.51

parkinsons 데이터셋의 RMSE 값이 너무 높게 나왔다. 따라서, 해당 데이터셋만 따로 가지고 각 모델들에 대해 heat map을 확인해보며 feature extraction이나 feature engineering을 통해 개선해야할 것으로 보인다.

```
import pandas as pd
from sklearn.model_selection import train_test_split

abalone = pd.read_csv("A6_datasets/abalone.csv")
concretecs = pd.read_csv("A6_datasets/concretecs.csv")
parkinsons = pd.read_csv("A6_datasets/parkinsons.csv")
skillcraft = pd.read_csv("A6_datasets/skillcraft.csv")
wine = pd.read_csv("A6_datasets/winequality-white.csv")

abalone_y = abalone.pop('rings')
concretecs_y = concretecs.pop('Concrete compressive strength')
parkinsons_y = parkinsons.pop('total_UPDRS')
skillcraft_y = skillcraft.pop('ComplexAbilitiesUsed')
wine_y = wine.pop('quality')

abalone_trainval, abalone_test, abalone_y_trainval, abalone_y_test = train_test_spl
it(abalone, abalone_y, train_size = 500, random_state = 1016)
concretecs_trainval, concretecs_test, concretecs_y_trainval, concretecs_y_test = tr
ain_test_split(concretecs, concretecs_y, train_size = 500, random_state = 1016)
parkinsons_trainval, parkinsons_test, parkinsons_y_trainval, parkinsons_y_test = tr
ain_test_split(parkinsons, parkinsons_y, train_size = 500, random_state = 1016)
skillcraft_trainval, skillcraft_test, skillcraft_y_trainval, skillcraft_y_test = tr
ain_test_split(skillcraft, skillcraft_y, train_size = 500, random_state = 1016)
wine_trainval, wine_test, wine_y_trainval, wine_y_test = train_test_split(wine, win
e_y, train_size = 500, random_state = 1016)
import warnings
warnings.filterwarnings(action='ignore')

from sklearn.pipeline import Pipeline
from sklearn.model_selection import StratifiedKFold, GridSearchCV, KFold
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor

pipe = Pipeline([('preprocessing', None), ('regressor', LinearRegression())])
hyperparam_grid = [
    {'regressor' : [LinearRegression()], 'preprocessing' : [StandardScaler(), MinMa
xScaler(), None]},
    {'regressor' : [Ridge()], 'preprocessing' : [StandardScaler(), MinMaxScaler(),
None],
    'regressor__alpha' : [0, 0.01, 1, 10, 100]},
    {'regressor' : [Lasso()], 'preprocessing' : [StandardScaler(), MinMaxScaler(),
None],
    'regressor__alpha' : [0.0001, 0.001, 0.01, 0.1, 1, 10], 'regressor__tol':[0.000
01, 0.0001,0.001]},
    {'regressor' : [RandomForestRegressor()], 'preprocessing' : [None],
    'regressor__max_features' : ['auto', 'sqrt', 'log2']},
    {'regressor' : [SVR()], 'preprocessing' : [StandardScaler(), MinMaxScaler(), No
ne],
```

```
    'regressor__epsilon' : [0.001, 0.01, 0.1], 'regressor__gamma' : [0.01, 0.1], 'r
egressor__C': [1, 100]},
    {'regressor' : [MLPRegressor()], 'preprocessing' : [StandardScaler(), MinMaxSca
ler(), None],
    'regressor__max_iter' : [5000,10000], 'regressor__activation' : ['tanh', 'relu'
],
    'regressor__solver' : ['lbfgs', 'sgd', 'adam'], 'regressor__hidden_layer_sizes'
: [(10,),(20,),(50,),(100,)]}]

kfold = KFold(5, shuffle=True, random_state=1016)
grid = GridSearchCV(pipe, hyperparam_grid, scoring='neg_mean_squared_error', refit=
True, cv = kfold)

data = [[abalone_trainval,abalone_y_trainval,abalone_test,abalone_y_test],
        [concretecs_trainval,concretecs_y_trainval,concretecs_test,concretecs_y_tes
t],
        [parkinsons_trainval,parkinsons_y_trainval,parkinsons_test,parkinsons_y_tes
t],
        [skillcraft_trainval,skillcraft_y_trainval,skillcraft_test,skillcraft_y_tes
t],
        [wine_trainval,wine_y_trainval,wine_test,wine_y_test],]

for i in data:
    grid.fit(i[0], i[1])
    print("best hyperparam : \n{}".format(grid.best_params_))
    print("best cross-validation score : {:.2f}".format(-grid.best_score_))
    print("test-set score : {:.2f}".format(-grid.score(i[2], i[3])))
```