**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\_split(abalone, abalone\_y, train\_size = 500, random\_state = 1016)

concretecs\_trainval, concretecs\_test, concretecs\_y\_trainval, concretecs\_y\_test = train\_test\_split(concretecs, concretecs\_y, train\_size = 500, random\_state = 1016)

parkinsons\_trainval, parkinsons\_test, parkinsons\_y\_trainval, parkinsons\_y\_test = train\_test\_split(parkinsons, parkinsons\_y, train\_size = 500, random\_state = 1016)

skillcraft\_trainval, skillcraft\_test, skillcraft\_y\_trainval, skillcraft\_y\_test = train\_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, wine\_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(), 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,)]}]

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\_test],

        [parkinsons\_trainval,parkinsons\_y\_trainval,parkinsons\_test,parkinsons\_y\_test],

        [skillcraft\_trainval,skillcraft\_y\_trainval,skillcraft\_test,skillcraft\_y\_test],

        [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])))