데이터 과학 프로그래밍 과제 6
2015313254 노인호

1. problem
- 총 12개의 csv파일로 만들어져 있는 dataset을 automl 프레임 워크를 구현하여 학습시키는 만든 모델로 새로운 데이터를 예측했을 때 얼마나 잘 예측했는지 모델을 평가하는 문제이다. 각 csv파일은 피쳐가 모두 다르고 데이터의 개수도 모두 다르다. 단 마지막 피쳐가 regression문제에 적용할 타겟 즉 종속변수가 된다.

2. data preprocessing
- 모든 파일들을 반복문을 이용해서 한번에 학습시키기 위해 dataset이라는 폴더 안에 넣어두었다. 또한 getdata.py를 이용해 마지막 피쳐를 타겟 즉 종속변수 y로, 나머지 피쳐를 독립변수 x로 설정하였다. x는 data라는 변수에 저장되고 y는 target이라는 변수에 저장된다. 그리고 train_test_split을 이용해서 데이터를 train set과 test set으로 8:2로 분리하였다. 후에 stratifiedKFold 즉 교차 검증방식을 쓸 것이기 때문에 valid set은 따로 나누어 주지 않았다. 또한 정규화와 같은 처리는 pipeline에서 따로 처리할 수 있기 때문에 다음 항목에 설명하도록 하겠다.

3. Learning Algorithm
-기본적으로 Pipeline과 GridSearchCV를 이용하였다. 처음에는 기본 pipeline에는 selectFromModel을 이용하여 feature selection을 진행하였지만(RandomForestRegressor 사용), 결과에 큰 차이가 없어 빼고 모델을 추가하는 방향으로 갔다. parma_grid 딕셔너리에 RandomForestRegressor()을 위해서 스케일링은 없음, n_estimator=100, max_depth=15 (데이터가 2^15=32768개를 넘는게 없어서), min_samples_leaf=2, max_leaf_nodes=None으로 설정하였다. RandomForestRegressor()모델이 시간이 오래 걸리는게 단점이지만, 스케일링도 필요없고 오버피팅도 발생할 염려가 거의 없기 때문에 이 모델을 선택해서 hyperparameter를 변경하기로 하였다. 그 다음 신경망 모델 MLPRegressor을 추가해 주었다. solver=lbfgs로 preprocessing은 정규화 스케일링과 없음 하나씩, activation=relu, logistic, tanh 3개, hidden layer size=(2,),(3,),(5,),(7,),(9,),(13,),(17,),(23,),(29,)를 추가해 주었다. StratifiedKFold 는 스플릿 수를 5개로 설정하였다. 학습은 구글 콜랩에서 진행하였다.

4. Conclusion
각 데이터 파일의 best hyperparameter과 cross-validation score, test set score를 구했다. GridSearchCV에서의 scoring은 rmse가 없어서 neg_mean_squared_error를 이용한다음 -1과 1/2승을 해서 최종 rmse를 구했다. 결과를 모두 한 장으로 담기에는 지면이 부족해서 코드를 첨부한 링크 안에서 결과물을 확인할 수 있다. 모두 cross validation score로 나온 rmse와 test set score의 rmse가 비슷하여 오버피팅 없이 학습된 모델을 만들 수 있었다. appliancesenergy.csv의 rmse가 엄청 큰 것을 제외하고는 모두 비교적 오버피팅 없이 잘 학습이 되었음을 알 수 있었다.

웹사이트 https://nbviewer.jupyter.org/에서 다음의 gist주소를 입력하면
https://gist.github.com/nosy0411/a8661b8a42fd294587a5e0033b69b56d
안깨진 코드와 결과물을 볼 수 있다. google colab에서 학습시켜 결과값만 jupyter notebook 에 붙여넣기 하였다.

```
import csv
import os
from getdata import dataset
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectFromModel

from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import GridSearchCV, StratifiedKFold

from sklearn.metrics import mean_squared_error

path = "./dataset"
# print(os.listdir(path))
# print(os.path.join(path,'*.csv'))

for input_file in os.listdir(path):
    data, target = dataset(input_file)
    X_train, X_test, y_train, y_test = train_test_split(
        data, target, test_size=0.2, random_state=0)
    print("====="+input_file+"==========")
    print("X_train:", X_train.shape)
    print("y_train:", y_train.shape)
    print("X_test shape:", X_test.shape)
    print("y_test shape:", y_test.shape)
    print("============================")

    # pipe=Pipeline([('preprocessing', None), ('feature_selection', SelectFromModel(RandomForestRegressor(n_estimators=64,max_features=X_train.shape[1]))), ('model',RandomForestRegressor())], memory="cache_folder")
    pipe=Pipeline([('preprocessing', None), ('model',RandomForestRegressor())], memory="cache_folder")
    param_grid = [
        {'preprocessing': [None],
```

```
            # 'feature_selection__threshold':["0.5*mean","mean"],
            'model__n_estimators': [100],
            'model__max_depth': [15],
            'model__min_samples_leaf': [2],
            'model__max_leaf_nodes': [None],},
            {'model': [MLPRegressor(solver='lbfgs')],
             'preprocessing' : [StandardScaler(),None],
             'model__activation' : ['relu','logistic','tanh'],
             'model__hidden_layer_sizes':[(2,),(3,),(5,),(7,),(9,),(13,),(17,),(23,),(29,)],
            }]

    kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
    grid = GridSearchCV(pipe, param_grid=param_grid, scoring='neg_mean_squared_error', cv=5, n_jobs=-1)
    grid.fit(X_train, y_train)

    print("Best params:\n{}\n".format(grid.best_params_))
    print("Best cross-validation score(root mean squared error): {:.2f}".format(((grid.best_score_)*(-1))**0.5))
    print("Test-set score(root mean squared error): {:.2f}".format((grid.score(X_test, y_test)*(-1))**0.5))

print("\n================================================================================")

print("================================================================================\n\n")
```