201735838 방준석

Open Source PHW

import numpy as np

import pandas as pd

from typing import List

from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler, RobustScaler

from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import GridSearchCV

def FindBestModel(df, target, scalers = None, encoders = None, models = None) :

    """

    Find the best combination of scaler, encoder, fitting algoritm

    print best score and best combination

    Parameters

    --------------------------------

    df : preproessed Dataset

    target : target column

    scalers: list of sclaer

            None: [StandardScaler(), MinMaxScaler(), MaxAbsScaler(), RobustScaler()]

            if you want to scale other ways, then put the sclaer in list

    encoders: list of encoder

        None: [OrdinalEncoder(), OneHotEncoder()]

        if you want to use only one, put a encoder in list

    models: list of encoder

        None: [DecisionTreeClassifier(entropy), DecisionTreeClassifier(gini), SVC(), LogisticRegression()]

        if you want to fitting other ways, then put in list

    """

    ##################################################

    """

    X: DataFrame to scaled

    y: DataFrame to encoding

    X\_category : columns to encode

    X\_num : columns to scaled

    df\_cateEmpty is True if There is no category feature

    df\_num is True if There is no numerical feature

    """

    X = df.drop(target, axis=1)

    y = df[target]

    X\_category = X.select\_dtypes(include = 'object')

    df\_cateEmpty = X\_category.empty

    X\_num = X.select\_dtypes(exclude = 'object')

    df\_numEmpty = X\_num.empty

    if scalers == None:

        scale = [StandardScaler(), MinMaxScaler(), MaxAbsScaler(), RobustScaler()]

    else: scale = scalers

    if encoders == None:

         encode = [OrdinalEncoder(), OneHotEncoder()]

    else: encode = encoders

    if models == None:

        DecisionTreeEntropy = DecisionTreeClassifier()

        DecisionTreeGini = DecisionTreeClassifier()

        SupportVC = SVC()

        LoR = LogisticRegression()

        classifier = [DecisionTreeEntropy, DecisionTreeGini, SupportVC, LoR]

    else: classifier = models

    # to Compare performance (accuracy) of the following classification

    # models against the same dataset.

    """

    best\_score model name : best score

    best\_param model name : best parameter

    best scaler\_encoder : best scaler and encoder

    """

    best\_score\_DT\_entropy = 0

    best\_DT\_entropy\_param = []

    best\_scaler\_encoder\_entropy = []

    best\_score\_DT\_gini = 0

    best\_DT\_gini\_param = []

    best\_scaler\_encoder\_gini = []

    best\_score\_SVC = 0

    best\_param\_SVC = []

    best\_scaler\_encoder\_SVC = []

    best\_score\_LR = 0

    best\_param\_LR = []

    best\_scaler\_encoder\_LR = []

    # hyperparameters for GridSearchCV

    DTEparameters = {'criterion':["entropy"],'max\_depth':[1,2,3], 'min\_samples\_split':[2,3]}

    DTGparameters = {'criterion':["gini"],'max\_depth':[1,2,3], 'min\_samples\_split':[2,3]}

    LRparameters = {'C': [0.1, 1, 10],'penalty': ['l2']}

    SVCparameters = { 'C': [0.001, 0.01, 0.1, 1, 10, 100],'gamma': [0.001, 0.01, 0.1, 1, 10, 100]}

    for i in scale :

        for j in encode:

            if df\_numEmpty is False :

                scaler = i

                scaler = pd.DataFrame(scaler.fit\_transform(X\_num))

            if j == OrdinalEncoder() and df\_cateEmpty is False:

                enc = j

                enc = enc.fit\_transform(X\_category)

                new\_df = pd.concat([scaler,enc], axis=1)

            elif j == OneHotEncoder() and df\_cateEmpty is False:

                dum = pd.DataFrame(pd.get\_dummies(X\_category))

                new\_df = pd.concat([scaler, dum], axis=1)

            else:

                new\_df = scaler

            for model in classifier:

                X\_train, X\_test, y\_train, y\_test = train\_test\_split(new\_df,y,test\_size=0.3, shuffle=True, random\_state=34)

                # GridSearchCV of each model is used

                if model == DecisionTreeEntropy:

                    grid\_dt\_E = GridSearchCV(model, param\_grid=DTEparameters, cv=3)

                    grid\_dt\_E.fit(X\_train, y\_train)

                    score = grid\_dt\_E.score(X\_test, y\_test)

                    if score > best\_score\_DT\_entropy:

                        best\_score\_DT\_entropy = score

                        best\_DT\_entropy\_param = grid\_dt\_E.best\_params\_

                        best\_scaler\_encoder\_entropy = [i,j]

                elif model == DecisionTreeGini:

                    grid\_dt\_G = GridSearchCV(model, param\_grid=DTGparameters, cv=3)

                    grid\_dt\_G.fit(X\_train, y\_train)

                    score = grid\_dt\_G.score(X\_test, y\_test)

                    if score > best\_score\_DT\_gini:

                        best\_score\_DT\_gini = score

                        best\_DT\_gini\_param = grid\_dt\_G.best\_params\_

                        best\_scaler\_encoder\_gini = [i,j]

                elif model == SupportVC:

                    grid\_SVC = GridSearchCV(model, param\_grid=SVCparameters, cv=3)

                    grid\_SVC.fit(X\_train, y\_train)

                    score = grid\_SVC.score(X\_test, y\_test)

                    if score > best\_score\_SVC:

                        best\_score\_SVC = score

                        best\_param\_SVC = grid\_SVC.best\_params\_

                        best\_scaler\_encoder\_SVC = [i,j]

                else:

                    grid\_LR = GridSearchCV(model, param\_grid=LRparameters, cv=3)

                    grid\_LR.fit(X\_train, y\_train)

                    score = grid\_LR.score(X\_test, y\_test)

                    if score > best\_score\_LR:

                        best\_score\_LR = score

                        best\_param\_LR = grid\_LR.best\_params\_

                        best\_scaler\_encoder\_LR = [i,j]

    # print best score and hyperparameters of each model.

    print('Best score for Decision Tree (Entropy) :', best\_score\_DT\_entropy)

    print('Model parameters : ', best\_DT\_entropy\_param)

    print('Scaling and Encoding Method :', best\_scaler\_encoder\_entropy)

    print()

    print('Best score for Decision Tree (GINI) :', best\_score\_DT\_gini)

    print('Model parameters : ', best\_DT\_gini\_param)

    print('Scaling and Encoding Method :', best\_scaler\_encoder\_gini)

    print()

    print('Best score for Support Vector Machine :', best\_score\_SVC)

    print('Model parameters : ', best\_param\_SVC)

    print('Scaling and Encoding Method :', best\_scaler\_encoder\_SVC)

    print()

    print('Best score for Logistic Regression :', best\_score\_LR)

    print('Model parameters : ', best\_param\_LR)

    print('Scaling and Encoding Method :', best\_scaler\_encoder\_LR)

    return

df = pd.read\_csv('ML\\breast-cancer-wisconsin.data')

df.columns = ['id','thickness','size\_uniformity','shape\_uniformity','adhesion','epithelial\_size',

              'bare\_nucleoli','bland\_chromatin','normal\_nucleoli','mitoses','class']

# describe dataset

print('Original dataset Row length : ',len(df))

# preprocessing

df['bare\_nucleoli'] = df['bare\_nucleoli'].astype('category')

idx = df[df['bare\_nucleoli'] == '?'].index

df\_temp = df.drop(idx)

print('preprocessing dataset Row length : ', len(df\_temp))

# drop id column

df\_temp = df\_temp.drop(['id'], axis = 1)

FindBestModel(df\_temp, 'class')

Examine Dataset



Output

텍스트이(가) 표시된 사진

자동 생성된 설명