안건하-전소영-고효진-오예진

Github: <https://github.com/Hyojinko/2021_datascience.git>

#OpenSource module

import warnings

warnings.filterwarnings("ignore")

import pandas as pd

from sklearn import preprocessing

from sklearn.neighbors import KNeighborsRegressor

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

from sklearn.ensemble import RandomForestRegressor

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

def process\_module(df, targetName):

# Split the dataset

y = df[targetName]

X = df.drop([targetName], 1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

# Normalization with 4 Scaling methods

maxAbsScaler = preprocessing.MaxAbsScaler()

minmaxScaler = preprocessing.MinMaxScaler()

robustScaler = preprocessing.RobustScaler()

standardScaler = preprocessing.StandardScaler()

df\_maxAbs\_scaled\_train = maxAbsScaler.fit\_transform(X\_train)

df\_maxAbs\_scaled\_train = pd.DataFrame(df\_maxAbs\_scaled\_train, columns=X\_train.columns)

df\_maxAbs\_scaled\_test = maxAbsScaler.fit\_transform(X\_test)

df\_maxAbs\_scaled\_test = pd.DataFrame(df\_maxAbs\_scaled\_test, columns=X\_test.columns)

df\_minMax\_scaled\_train = minmaxScaler.fit\_transform(X\_train)

df\_minMax\_scaled\_train = pd.DataFrame(df\_minMax\_scaled\_train, columns=X\_train.columns)

df\_minMax\_scaled\_test = minmaxScaler.fit\_transform(X\_test)

df\_minMax\_scaled\_test = pd.DataFrame(df\_minMax\_scaled\_test, columns=X\_test.columns)

df\_robust\_scaled\_train = robustScaler.fit\_transform(X\_train)

df\_robust\_scaled\_train = pd.DataFrame(df\_robust\_scaled\_train, columns=X\_train.columns)

df\_robust\_scaled\_test = robustScaler.fit\_transform(X\_test)

df\_robust\_scaled\_test = pd.DataFrame(df\_robust\_scaled\_test, columns=X\_test.columns)

df\_standard\_scaled\_train = standardScaler.fit\_transform(X\_train)

df\_standard\_scaled\_train = pd.DataFrame(df\_standard\_scaled\_train, columns=X\_train.columns)

df\_standard\_scaled\_test = standardScaler.fit\_transform(X\_test)

df\_standard\_scaled\_test = pd.DataFrame(df\_standard\_scaled\_test, columns=X\_test.columns)

# Alogrithm

print("\n------------------------- Using maxAbs scaled dataset -------------------------")

max\_score\_maxAbs = algorithm\_module(df\_maxAbs\_scaled\_train, df\_maxAbs\_scaled\_test, y\_train, y\_test)

print("\n------------------------- Using minMax scaled dataset -------------------------")

max\_score\_minMax = algorithm\_module(df\_minMax\_scaled\_train, df\_minMax\_scaled\_test, y\_train, y\_test)

print("\n------------------------- Using robust scaled dataset -------------------------")

max\_score\_robust = algorithm\_module(df\_robust\_scaled\_train, df\_robust\_scaled\_test, y\_train, y\_test)

print("\n------------------------- Using standard scaled dataset -------------------------")

max\_score\_standard = algorithm\_module(df\_standard\_scaled\_train, df\_standard\_scaled\_test, y\_train, y\_test)

# Result

max\_score\_result = max(max\_score\_maxAbs, max\_score\_minMax, max\_score\_robust, max\_score\_standard)

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

print("Final maximum score: %.6f" % max\_score\_result)

def algorithm\_module(X\_train, X\_test, y\_train, y\_test):

# Linear Regression

line\_reg = LinearRegression()

line\_reg.fit(X\_train, y\_train)

y\_prec\_linear = line\_reg.predict(X\_test)

score\_linear = line\_reg.score(X\_test, y\_test)

print("\ny\_predict\_linear: \n", y\_prec\_linear[0:50])

print("Score: %.6f" % score\_linear)

# Polynomial Regression

poly\_reg = PolynomialFeatures(degree=2)

X\_poly\_train = poly\_reg.fit\_transform(X\_train)

X\_poly\_test = poly\_reg.fit\_transform(X\_test)

pol\_reg = LinearRegression()

pol\_reg.fit(X\_poly\_train, y\_train)

y\_prec\_poly = line\_reg.predict(X\_test)

score\_poly = pol\_reg.score(X\_poly\_test, y\_test)

print("\ny\_predict\_poly: \n", y\_prec\_poly[0:50])

print("Score: %.6f" % score\_poly)

# KNN algorithm

knn = KNeighborsRegressor(n\_neighbors=5)

knn.fit(X\_train, y\_train)

y\_prec\_knn = knn.predict(X\_test)

score\_knn = knn.score(X\_test, y\_test)

print("\ny\_predict\_KNN: \n", y\_prec\_knn[0:50])

print("Score: %.6f" % score\_knn)

# Random Forest

random\_forest = RandomForestRegressor(max\_depth=4, random\_state=0)

random\_forest.fit(X\_train, y\_train)

y\_predict\_rf = random\_forest.predict(X\_test)

score\_rf = random\_forest.score(X\_test, y\_test)

print("\ny\_predict\_RF: \n", y\_predict\_rf[0:50])

print("Score: %.6f" % score\_rf)

max\_score = max(score\_linear, score\_poly, score\_knn, score\_rf)

return max\_score

#Module Result

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명